AN EVOLUTIONARY ALGORITHM FOR GENERAL SYMBOL SEGMENTATION

A Thesis

Presented to

The Faculty of Graduate Studies

of

The University of Guelph

by

STEPHEN PEARCE

In partial fulfilment of requirements

for the degree of

Master of Science

August, 2003

0-612-85085-4

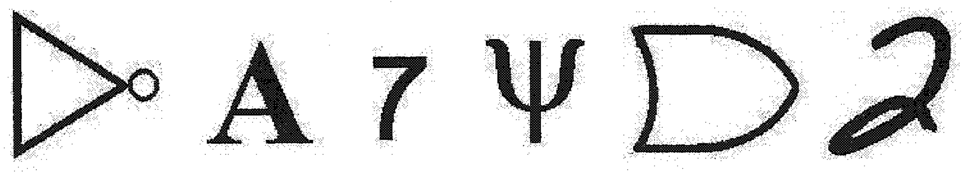
ABSTRACT

AN EVOLUTIONARY ALGORITHM FOR GENERAL SYMBOL SEGMENTATION

Stephen Pearce
University of Guelph , 2003

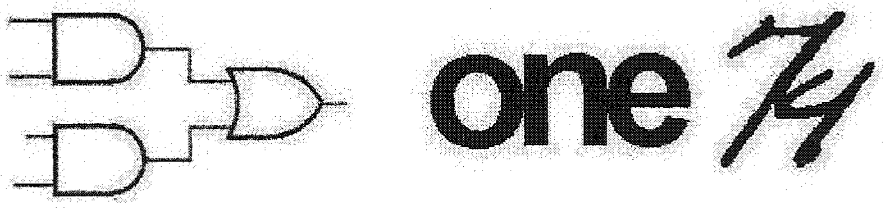
Advisor:
Dr. M. Ahmed

One of the most common forms of visual communication is that of written words and line diagrams. These words and diagrams take many forms ranging from written script in English, Arabic, or other languages to symbolic diagrams like electrical circuit schematics and process flow diagrams. These scripts and diagrams can be easily digitized using a scanner, or camera, but only exist in image format. It would be of a great advantage to be able to convert these images to a more useful knowledge based format, but automatic processes to recognize and understand the meaning of them face many difficulties. Many researchers have developed techniques to accurately recognize clean isolated symbols such as individual characters or symbols in a line diagram.



Clean isolated symbols.

However, all these systems face difficulty when two or more symbols are connected.



Touching and connected symbols.

Due to the range of possible scripts, diagrams, or combination thereof, it is a daunting task to develop approaches to the problem. One approach, that will work for segmenting a variety of scripts and diagram types into clean isolated symbols is presented here.

This technique is based on a genetic, or evolutionary algorithms, and uses an independent and replaceable system to verify valid symbols in any connected string of characters or line diagrams. The input to the system is a scanned image of the connected string or diagram, that is first converted to black and white, then thinned to a single pixel width line. The line image is then represented by a powerful graph structure that is used to interchange between individuals in the evolutionary algorithm, and binary image form. The population is initialized using a special splitting algorithm based on depth first searching of the graph, and is evolved until a number of sub graphs that represent the symbols contained in the connected string are isolated. These sub graphs can then be drawn individually, together on one image, or used as input to some higher level understanding system.

The new system is capable of isolating symbols in any domain that can be described as "touching or connected symbols in a line diagram". This definition includes touching characters, logic circuit diagrams, and many other types of line diagrams. Experiments were performed, and results were presented on a number of data sets that varied both in the number of symbols, and the types of symbols present in the diagram.

Acknowledgements

Looking back over the years it took to create this work, it is impossible to compile a complete list of the influence and encouragement I have received from my friends and mentors. The words here can not express the gratitude I have for all of the support I have received, and will always have in my heart.

First, I would like to acknowledge Dr. Maher Ahmed. Without his encouragement before I even considered this path in life, and his wise counsel after, I would never had the confidence or courage to do this.

There have been many other mentors at both WLU and UoG, whose support both got me started, and kept me going. Dr. Kathleen Cameron from WLU has been particularly supportive, and I thank her for keeping me excited about education. Dr. Wirth and Dr. Grewal were the first to give me a sense of comfort and belonging, thanks for talking to me on the level.

Trying to list all of my friends that gave me the push to keep going, is an undertaking I will not attempt. The list would never be complete, and I dare not miss one because they were all important. Instead, I will show my gratitued in life, and the lasting friendships we all share. I would however like to thank Estelle Arthur in particular, for setting and example of how to balance living and learning, when one threatens to consume the other.

And lastly, I could not have survived in academia without the unquestionable love and support from my parents. No matter how little I understood about where I was headed, they were always behind me, offering more than I ever wanted to accept.

Notes to the Reader

The work in this thesis has spawned at least one research paper of the same title. The paper was accepted to *IEEE International Conference on Document Analysis and Recognition (ICDAR'03)*. The research will be presented in Edinburgh, Scotland, August 3 - 6, 2003.

Table of Contents

Table of Figures

Chapter 1: Introduction

1.1 Motivation

In today's modern world computers has become more of a necessity than a tool. It is becoming increasingly common to have a web-enabled computer in any home or office. Until recently interaction with these increasingly common machines, was handled by a select few that knew the computer's language, and could understand it's responses. A new and welcome change in philosophy is taking place, where more and more people are attempting to have computers understand the more natural communication between humans.

Efforts are underway to mimic all lower and higher level functions that humans use to communicate efficiently. The lowest level of information exchange is the actual transmission of information in both audible and visual communication. Audible communication consists of speech and is easily input using a microphone. Visual communication is made up of gesturing, writing, and drawing. The higher level functions necessary to use the information transmitted in this way are varied and difficult to define. At one stage, this information must be analyzed and recognized, so that the ideas and concepts it expresses can be identified. At another stage, the information can be stored for later retrieval or otherwise used by in a useful way to aid with human understanding, or accomplish goals of a computer agent.

One of the most abundant sources of information is printed documents and diagrams. Volumes of literature have been produced for recreation, instruction, and countless other reasons. The advantages of having a computer that could accurately and tirelessly sift through this volume of data are boundless. It would be possible to digitize and search historical documents, interpret a students hand-drawn class notes, or train intelligent systems from existing literature.

However this task involves more than it may seem. Children are not taught to read by recognizing entire words, but by looking at each character in a word. Only when a child learns the characters and the spelling of a few words can they be recognized. Similarly, before a the meaning of a diagram can be understood, the individual symbols must be examined. The problem of identifying isolated characters and symbols has been addressed by many researchers [4, 27, 31, 40, 48]. One flaw of all of these systems is that they assume that the input symbol is clean, isolated, and complete as in figure 1, which rarely occurs when dealing with hand-writing and hand-drawn diagrams.

To satisfy this assumption it is necessary to develop a system that is capable of locating and separating valid symbols from a connected string. Different systems have been developed for handwriting [3, 8, 41, 42, 45, 51, 54,] and other line diagrams [11, 35, 47, 56, 58]. Every segmentation system however has been built using algorithms and techniques that are highly specialized to the domain that the source images come from, making it necessary to implement a different system for each type of diagram or handwriting.

For a system that would be able to interpret both line images and script, it would be an advantage to be able to handle all inputs in a common way. The system presented here meets this demand for flexibility. It may be used to segment line diagrams and handwriting, provided that there is a symbol recognition system implemented for each domain.

1.2 Problem Statement

Good character and symbol recognition systems have been developed by [31] and others. These recognition systems rely on the same assumption however, that the input symbol is a single, clean image of a character as in figure 1. Although necessary for development of the ideas, this assumption is impractical in the real world.

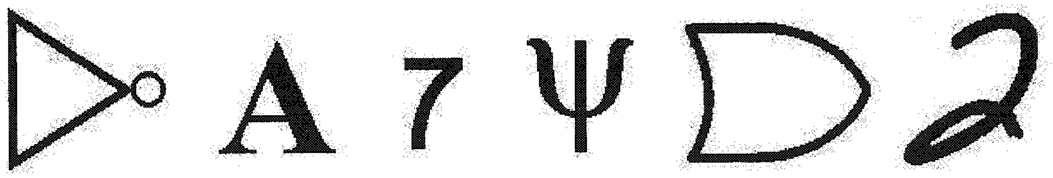


Figure 1. Clean isolated symbols.

The problem of separating unknown symbols takes many forms. The unknown symbols could have a related meaning in any context imaginable. Most commonly these symbols are adjacent characters in a hand written string as in figure 2, but could also be components in a circuit diagram, or glyphs in a mural as in figure 3. There exist many different types of segmentation problems, but they may easily be broken down into domains based on possible layouts. For handwriting, all symbols are arranged horizontally on some baseline. For line images, the problem is more complex since symbols may be arranged both horizontally and vertically, and are not necessarily separated by a fixed distance such as in digital circuit diagrams.

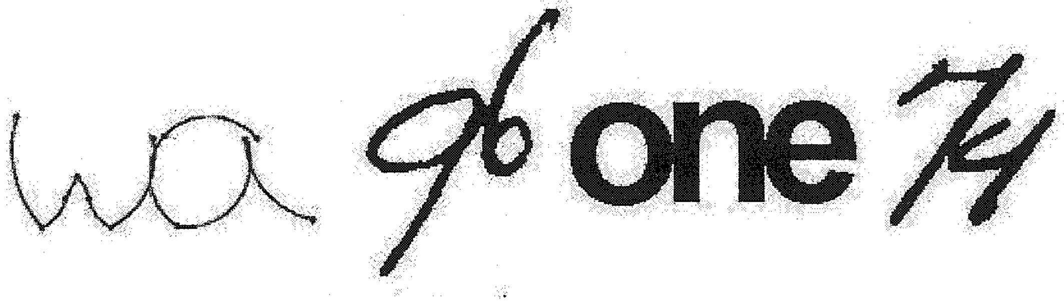


Figure 2. Touching character and digit symbols.

In reality, most handwriting runs together, contains broken characters, or is slurred by other noise on the paper or lines. The effects of noise can be minimized though image processing techniques [29, 53], but characters running together presents a paradox. The paradox is that the individual symbols can not be recognized using [31] until they are separated. However, until they are recognized there is no reliable information that would help to perform the segmentation. Information that would help includes what the symbols are, or even how many are adjacent.

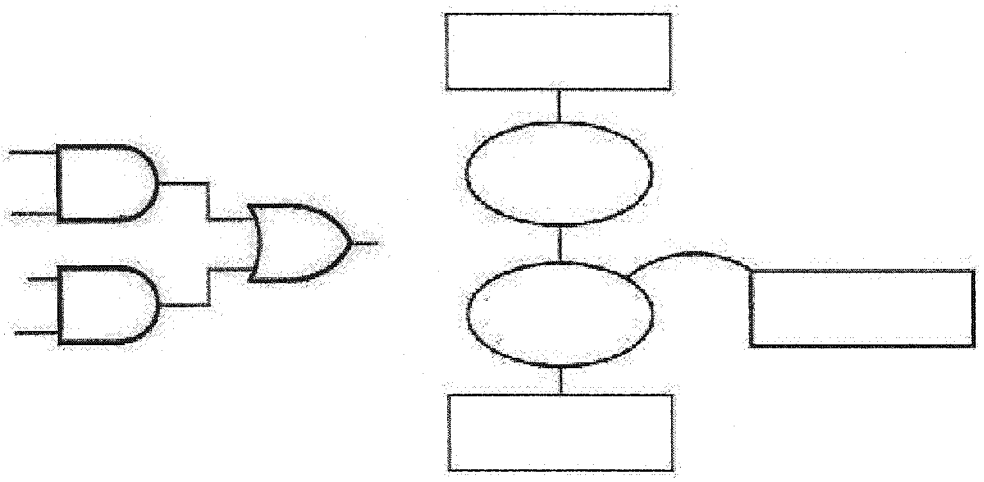


Figure 3. Connected line diagrams.

A solution to the paradox of segmenting two or more adjacent characters will be presented in the following thesis. In the following section, the problem is examined as both a recognition problem, and also as an optimal searching problem.

1.3 About this Document

The remainder of this chapter will describe the nature of the segmentation problem as seen by the author. This discussion will provide the reader with an introduction to the strengths of the proposed system. The remainder of the thesis will be divided into five chapters and two appendices. The next Chapter, number Two, will discuss the background knowledge that is required to understand the operation of the proposed system. Chapter Three will discuss previous approaches to the segmentation problem for characters and line diagrams, although no previous work on a general system was found. Chapter Four will introduce the reader to the proposed system in great detail. The results of the numerous experiments performed will be discussed in Chapter Five. This goal of this discussion is to show a few examples of segmentation in characters and line diagrams, and then to summarize all of the results in a common way. The last Chapter, number Six, will draw some conclusions about the performance of the algorithm as a whole, and identify a few areas for further improvements.

Two appendices included at the end of this thesis. The first appendix will contain pseudocode of some important algorithms. The pseudocode will help to clarify the description of the key algorithms. The second appendix will contain a more detailed summary of the experimental results. The form of this appendix will be discussed in greater detail in Chapter Five.

1.4 Angles of Approach

Before attempting to find a solution for this problem, it is necessary to discuss it's nature and the applicable techniques. The segmentation issue examined here can be viewed in a number of different ways.

Segmentation as an Algorithmic Problem

A "good" algorithm to solve a problem is guaranteed to find a solution in polynomial time, for every possible case. The nature of the problem excludes this approach, since the variety of inputs will be nearly infinite. With current input methods, and the variety of writing instruments it is impossible to identify or define all forms of handwriting and diagrams.

Segmentation as an (AI) Recognition Problem

To classify something as a recognition problem, there would have to be a complex relationship between a set of input states and the output. This relationship can either be learned, or rule-based, and strives to identify a single possible solution or response for any input case. It is quite common that the input domain is not completely deterministic, but is well enough known that generalizations can be made so that inputs fall into a smaller range of values or sets of data.

Many systems have been developed that strive to identify the core of each symbol and work from there [11]. Other systems sometimes use grammars or symbol matching [8, 35] to determine where characters with a known beginning would have an end. In either method, the same operations are repeated a number of times until the process terminates or a result is found.

Methods that view segmentation as an artificial intelligence problem are often completely dependant on the domain of symbols they are segmenting. A system that segments based on handwriting primitives may not be adaptable to any other domain. Similarly they may not present enough flexibility in their target domain, if the string being analyzed does not conform to the primitives.

A system that segments based on artificial intelligence methods may also handle exceptions very poorly. Before recognition occurs it is necessary to do both a separation of complete characters, and a unification of broken characters. This coupled with the reality that sometimes there are meaningless lines or information attached to a character symbol can cause insurmountable obstacles if the special case is not previously identified.

Segmentation as Optimal Searching

On the other hand, segmentation may be viewed as an optimal searching problem The goal of an optimal search is to find a good solution in a reasonable amount of time. This view has the advantage that it accepts the reality that some handwriting is gibberish, or illegible to any reader.

A good optimizer will work along a variety of different paths in order to try different solutions. It will not fail when an input belongs to a special case that was not predicted or enumerated. Instead, it will make efforts to extract what information it can, using all or part of any potential symbol, and employs some randomness that offsets the unpredictability of human handwriting.

Viewing segmentation as an optimization problem also adds to the versatility of the system. Optimization techniques generally employ an evaluation or fitness function to compare candidate solutions. This evaluation can be adapted to control the segmentation in a variety of ways. The evaluation is the only part of the segmentation system that needs any knowledge of the target domain, so most of the impact of retargeting will be in the evaluation. Also, the evaluation can be tuned to prefer certain symbols, or arrangements of them so that higher level information can easily be used when available to help with the segmentation.

1.5 The Proposed System

This thesis will develop and show the feasibility of a system that performs segmentations on script and line diagrams. The developed system views the goal of separating unknown symbols as one of optimal searching, attempting to identify regions of ink on the page that represent symbols when they do not appear in clean isolated form. The scope of the thesis starts in the area above noise removal and identifying the regions of ink, and ends just below gaining any high level knowledge of the meaning of the symbols.

Input to the system will be a image of a connected string of characters, or connected components in a line diagram that is free of noise from scanning, similar to Figure 2 and Figure 3. The output will be a set of noise free images of valid symbols. When necessary for higher level processing, the information on relative location of the symbols is readily available, but otherwise no information from a higher level source is necessary, thus no interpretation of the symbol meaning is done.

For verifying the output symbols, as well as aiding with evaluation, the OCR system in [31] will be used to give an accept or reject rating to connected regions of the input string. Input to the symbol verification will be limited to the equivalent of a function call with an image of a potential symbol as a parameter, and a boolean return value describing if the symbol is valid. Interaction will be limited to this so that the symbol verification can be easily replaced if a better system is found, since symbol recognition is necessary but beyond the scope of this work. The symbol verification system will require patterns that represent valid symbols, and these are provided for the symbols that appear in the input strings.

The proposed algorithm is based on a genetic algorithm framework, but for reasons discussed in Chapter 2, the terminology of an 'evolutionary algorithm' is preferred. The algorithm consists of four phases that will be discussed in detail in Chapter 4. A brief summary is presented here as an overview to show the place of each operation in the whole algorithm.

The first step is to skeletonize a thick input line image.

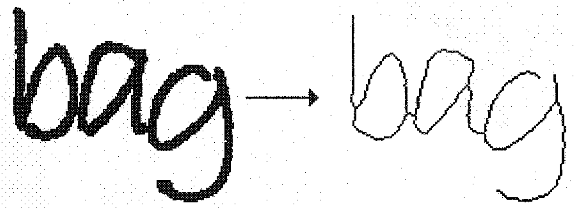


Figure 4. A thick line image has been thinned into a single pixel width line image.

The second step is to represent the thin line image with a graph data structure.

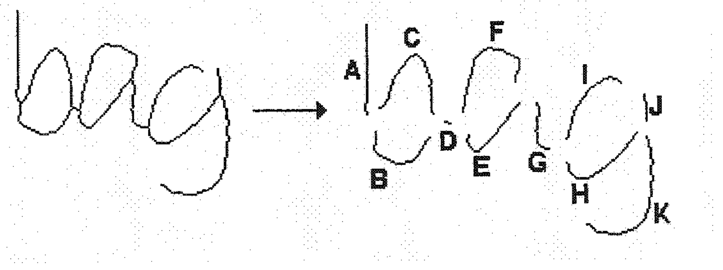


Figure 5. The thin line image has been graphed, with edges labelled A to K.

The third step partitions the graph into the individuals in the population. Some effort is made to ensure a diverse population.

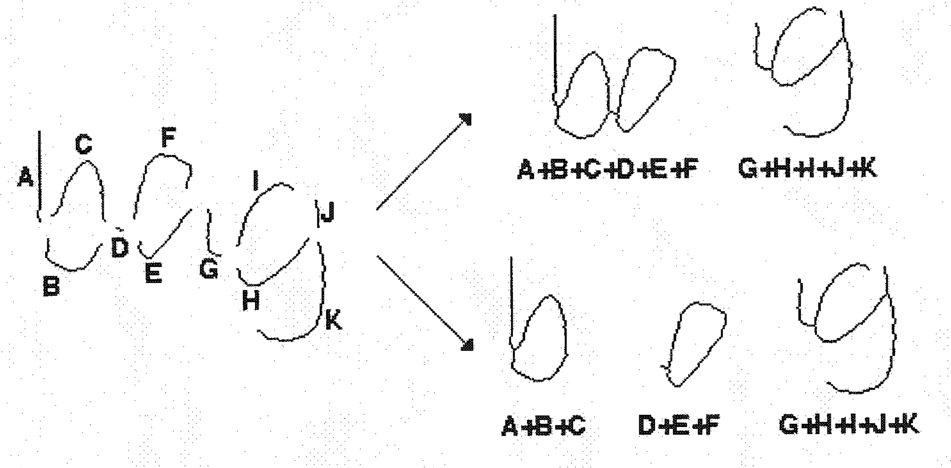


Figure 6. The graph has been divided into two distinct partitions. There will a partition for each individual in the population.

The fourth step involves two different evolution operators, mutation and crossover. These operators will work together to evolve the individual solution hypotheses into partitions that represent isolated symbols. Evaluation of the individuals and termination of the algorithm are discussed in detail in Section 4.6.

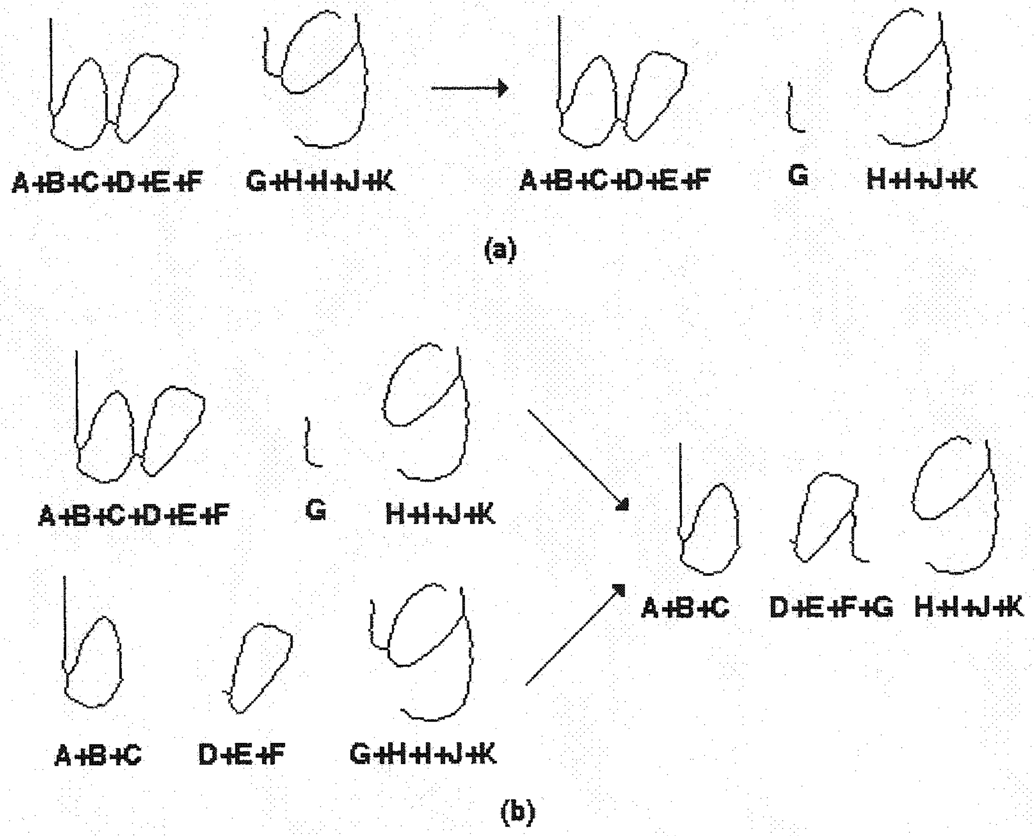


Figure 7. (a) A mutation operation moves an edge from one part to another, possibly new one. (b) A crossover operation preserves matched symbols in two different individuals.

The final result, is a partition that represents all of the isolated symbols. It is possible, and in many cases likely, that there will be parts in the partition that do not represent symbols. the goal is to have all of the symbols isolated in one part, and the number of unmatched parts does not matter.

Details and Data Sources

The proposed system has been implemented in Java, and experiments performed on jpeg images. Since the proposed system relies on accurate symbol verification and this is not a topic covered here, effort will be made to ensure the best performance of the system in

[31]. For this reason, sources of experimental data have been strictly controlled, so that standard symbols appear in various connected forms. Sources of data include:

- Manufactured images of thick touching characters.
- Manufactured images of thinned overlapping characters.
- Scanned images of circuit diagrams from [55] after manual noise removal. These circuit diagrams were all of a common typeset, with equal symbol sizes, and scanned at the same resolution.

It is generally not accepted to rely on manufactured data for such a system, but this was done for two reasons:

- Improve performance. The system will fail if a symbol in the string cannot be recognized in isolated form. Symbol recognition is beyond the scope of this thesis, so failures of this must be kept minimal.
- Lack of available data. No database of suitable information was encountered for showing the feasibility of this system. The requirements as stated above were closest met by the CEDAR database of handwritten zip codes. This database has a large number of noisy images, and not all images contain connected strings of characters. Due to the handwritten nature of the database, the recognition system in [31] would have required a complete set of patterns for handwritten digits, which was not available.

The impacts of using strictly controlled and manufactured must also be considered. The only upgrade necessary to extend the capabilities from touching typed characters to handwriting is a more complete symbol verification system. An attempt has been made to provide testing data with a variety both in the way the characters are touching, as well as variety within the characters themselves.

The scanned diagrams provide the closest to real world data of all experiments. Reasons for using typeset diagrams are similar to those suggested for touching typed characters. That is, the only upgrade necessary to extend the capabilities is in the symbol verification. If this were improved to recognize handwritten symbols of various sizes, the system would be applicable to hand-drawn diagrams. Noise removal is beyond the scope of this thesis so all operations were performed manually. Large areas of noise were removed so that each image contained a single connected line image, with one or more symbols and some attention was given to smooth the edges of lines. A simple threshold was used to binarize the images into black ink and white background.

Chapter 2: Background Knowledge

This chapter describes some of the background knowledge that is to understand the system being presented. The discussions are intended to bring the novice reader into basic technical knowledge of the graph structure and searching, as well as enlighten everyone on how OCR, graphs, and searching strategies can be used for segmentation.

2.1 OCR

Optical Character Recognition, or OCR, refers to the act of identifying a given character or symbol in an image. OCR systems have been developed that perform well [4, 27, 31, 40, 48] but operate under the same assumption that the symbol is clean and isolated. Complete exploration of OCR research is beyond the scope of this thesis. It is sufficient to define how the proposed system is related to the concept of OCR, and how the proposed system can be used as an early phase in a complete document recognition system.

It was briefly mentioned in the first chapter that an OCR system will be employed to accept or reject valid symbols in the proposed system. Any good OCR system is usable, provided that it can recognize symbols in a skeletonized form. The system in [31] is sufficient and was implemented to function as an exterior module of the segmentation system. It is also necessary to switch terminology, to use the term symbol verification as opposed to OCR, since the segmentation system is not interested in the meaning of a recognized symbol, only that one is confirmed as valid.

The symbol verification system is exterior and independent of the symbol segmentation system. This allows for flexibility of domains, so that a general system like [31], or a specialized system can be used when desired. Any chosen system must allow for the single interface between the symbol verification and segmentation. The verification system must be able to examine an image, and either accept or reject it as a valid symbol.

The proposed segmentation system employs the framework of an evolutionary algorithm, which will be discussed in greater detail in the following sections. As part of this framework, the symbol verification is used in the evaluation of candidate solutions. A candidate solution consists of a number of potential symbols, each of which must be tested to see if it is a valid symbol. By maintaining the separation between the symbol verification and the symbol segmentation, it is possible to implement a system that is both powerful, flexible, and easily adaptable to new techniques and technology.

2.2 Graphs

A graph, as it is used here, is a data structure that contains two types of elements, nodes and edges. A node can be used to represent an object, point, state, or any thing else, while an edge represents the relationship between two nodes. Nodes are said to have a degree which corresponds to the number of adjacent edges. Graphs have been used in many applications for storing large amounts of complex data, such as cities and roads on a map or terminals and communications in a network.

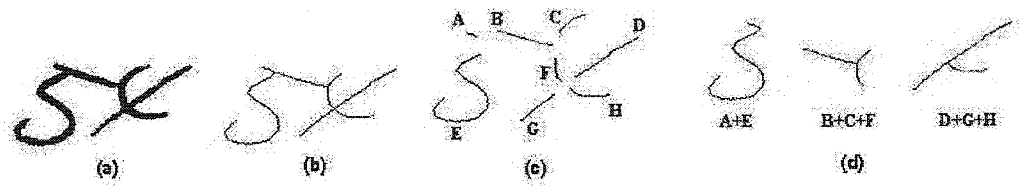


Figure 8 (a) Sample input image from CEDAR. (b) A thinned version of (a). (c) A graphed version of (b) using the*traceGraph*algorithm. (d) A partition of the graph in (c) with 3 parts.

A partition of a graph is a division of the graph into a number of different sections or parts. Each node must be assigned to at least one part and each part may contain any number of nodes. The part contains any edges connecting two internal nodes and the parts are separated by any edges that have one end in either part.

In this system, a graph is used as an efficient representation of the connected string input. An example of such a graph is shown in Figure 8(c) which is formed using the *traceGraph* algorithm in Appendix A. Nodes in the graph represent the end points of lines in the image, and exist at the location of any pixel that has more or less than two neighbouring pixels. Thus nodes will exist at the end points and intersection of any strokes in the image. The nodes contain attributes that record the position of the node in the image. Nodes may be considered equivalent only if they are placed at the exact same position. Nodes that are adjacent neighbours will be considered as separate, and joined by an edge with a length of 0.

The edges represent the actual lines of the image. An edge connects two nodes, and contains a chain code that records the steps used to trace the ink line between the nodes. Edges may be considered equivalent if they represent the same line, thus an edge E from n1 to n2 that heads due east, is equivalent to ~E from n2 to n1 that heads due west. However, edge E1 from n1 to n2 is not equivalent to edge E2 from n1 to n2 if they contain different chain codes. The weight or area of an edge is defined here to be the number of steps in the chain code for the edge.

Graphs can be represented in a number of different ways, such as a list of adjacent nodes, or a list of edges and their end points. There are even more complex options involving connection matrices or more complicated data structures. The internal representation chosen for implementation may affect the performance of the algorithm, but it is irrelevant for discussion of the theory. More will be discussed about the structure chosen in Chapter 4.

Before it may be graphed, the image must first be reduced to a line image with a single pixel width using an algorithm that produces results similar to [30]. Then the *traceGraph* algorithm shown in Appendix A may be used to initialize the graph representation. Figure 8 shows how this algorithm forms the graph from a particular input image. The input image, Figure 8 (a) is thinned into Figure 8 (b). Nine nodes are quickly placed, with six at degree one stroke endpoints, two at a degree three intersections, and one at a degree four intersection. These nodes are then connected by eight edges, labelled A though H in Figure 8 (c). Figure 8 (d) shows a simple partition of the graph, with edges A and E belonging to the first part, edges B, C, and F belonging to the second part, and the remaining edges D, G, and H compose the third part. In this instance, each edge belongs to one, and only one part, however it is possible to have a single edge belong to multiple parts, but each edge must be included in at least one part.

2.3 Searching in Large Spaces

Given a specific problem, and a list of possible solutions (or search space), it is possible to find the best solution to the problem by examining the entire search space. There are a number of problems that are relevant to some researchers in computer science that are considered to be unsolvable by an algorithmic approach. These problems are intractable because they are so complex, that even with the most powerful computers available the entire search space can not be examined in a single lifetime. Even for problems with a reasonable search space, it may be impractical to perform a complete search because of limits placed on time and resources. The act of attempting to find the best solution, or a reasonable one that satisfies some constraints is a broad and active research area [5, 6, 9, 22, 44].

As with most areas of research, it is very difficult to say that there is one best searching strategy. Instead, an appropriate strategy must be chosen for the problem at hand. For problems that can be considered "distributed" in nature, like finding a fastest route through a computer network, the most appropriate searching methods may be distributed searching strategies that each examine a small region of the search space. Other problems may be more easily solved using techniques like evolutionary algorithms, ant colony optimization, and simulated annealing that focus on mimicking natural behaviour and have proved to be robust and easily understood.

2.3.1 Evolutionary Algorithms

The term "evolutionary algorithm" will be used throughout the rest of this paper to describe an algorithm that "evolves" a solution over time. The "evolution" of a solution is a progression from some solution to a more suitable one that satisfies the constraints of the problem. The proposed algorithm is based on what is commonly known as a "genetic" algorithm, but the author prefers to avoid the use of the word "genetic" since it provides too many allusions to the natural process. As the term is used here, an evolutionary algorithm is a heavily modified version of a genetic algorithm.

Evolutionary algorithms are a recent approach to finding optimal or near optimal solutions to some problems that have a very large search space [1, 23, 39]. Evolutionary algorithms take advantage of a population of candidate solutions, and specialized operators to tweak and combine these solutions. Over time the population evolves and characteristics of the better solutions are shared amongst the others in the population, forming new candidate solutions and hopefully an optimal one.

Every evolutionary algorithm may be defined by a number of factors. The fundamental aspects being:

- *problem* - The input and problem to solve. For instance, given a computer network layout, with information on the time it takes to send packets between each machine, one could be asked to find the shortest route through a particular set of terminals. In this thesis the problem is to identify all of the symbols in a connected string.

- *goals* - The criteria that the algorithm aims to satisfy. For this thesis the universal goal is to identify an unknown number of symbols in a connected string or line image. Since the number of symbols is unknown, and the number of assumptions regarding the layout of the input is kept to a minimum, it is impossible to define a heuristic that can differentiate between a partition that isolates all of the symbols, or only a subset of them. It is also possible to identify two or more valid partitions from the same input, such as a cursive character 'w' being segmented into the cursive characters 'u' and 'i'. Without high level knowledge, the correct partition can not be selected from the set of valid ones.

When the problem and goals have been defined, the algorithm must be tuned in a number of ways. The most important considerations are the representation of the individual (or solution to the problem) and the goals to which it aspires. The individuals must have an internal structure that is organized in such a way that partial information from two or more parents can be combined to form a new individual of the species. It is also necessary to be able to evaluate the fitness of an individual relative to some goal or ideal individual.

Some popular choices for the structure of individuals are a bit string, or a permutation of integers. When considering a representation for candidate solutions to a particular problem one must consider the following four factors [6, 7, 9]:

- All possible solutions can be represented. It would not be possible to find an optimal solution to the problem if it were impossible to describe the solution with the given representation.
- All solutions are represented equally. The algorithm should not have a natural preference for solutions in a small region of the search space.

- It should be easy to evaluate a solution. This will help with the practical implementation, since conversion of the solution from one form to another can be an expensive operation.
- It should possess locality, so that a small change in the individual will produce a small change in the associated solution.

For example, in the "travelling sales-person problem", the solution is conveniently represented as a permutation that indicates the order which the nodes are visited ensuring all orderings are possible and equally likely. Obviously, if an element of the permutation were moved to a different location only two changes would occur in the path ensuring locality. The first path change is that the elements preceeding and proceeding the nodes original position would now be joined, as well as the elements preceeding and proceeding it's new position would be disconnected, and joined to the newly inserted element. All other portions of the path are left unchanged.

In general, an evolutionary algorithm will employ the following five processes or operators throughout it's lifecycle.

- *initialization*
- *mutation*
- *crossover*
- *evaluation*
- *aspiration*

Initialization consists of creating an initial population of individuals. In some cases these individuals may be created at random, but it is quite common that the goals may be satisfied sooner, and with better quality individuals, if the population is initialized near good solutions. Often times an algorithm that approximates a reasonable solution to the problem is used to initialize individuals.

Mutation introduces new evolutionary material to the population by spontaneously changing an individual. This operation may take place at random, or be directed [31] to create a better individual only. Random mutations may sometimes create an individual that is infeasible as a solution to the problem, or one that was worse than the original. In some cases this is undesirable, in others it is necessary to be able to find new solutions. For the system presented in this thesis, a hybrid mutation will be used. The hybrid mutation randomly chooses a mutation from a reduced set of possible mutations. The reduction of the number of mutations provides the directed mutation performance. A more detailed discussion of this mutation will be provided in chapter 4.

Crossover is an operation that mimics procreation in living beings. Two (or more) individuals are selected as parents, so that some material from one may be combined with material from another to produce a new individual that shares characteristics of all of it's parents. Crossover operators have been defined to work in many ways, from the "one point crossover" [44] which exchanges sub-strings of the parents to the "uniform crossover" which randomly takes material from either parent. The ideal crossover operator depends on both the problem at hand, and the selected chromosone representation. The crossover used in the proposed system preserves matched parts of the partition, but otherwise behaves as a uniform crossover.

Evaluation involves comparing the solution presented in an individual, to the ultimate goals of evolution. The fitness of an individual is a measure of this comparison, and can be used to determine which individuals are better candidate solutions, or even when the goals of evolution are satisfied. The evaluation of an individual is meant to provide some means of ranking them, and to quickly and easily determine the status in relation to the goals. Most often, the evaluation will assign a single value representing a fitness, but the system presented here uses a more complicated vector evaluation. More details of the vector evaluations advantages and uses are presented in chapter 4.

Aspiration is a special case of mutation, that occurs when an individual is evaluated to be close to a goal. If the individual examined differs from the goal in a known or determinable way, then the individual may be mutated into the goal solution. This mutation does not occur at random, and is directed to achieve a goal, not only towards it.
The ability of an individual to undergo an aspiration operation is usually noticed as a side effect of evaluation and is not relied upon as a means to develop solutions. Instead, aspiration should be considered an enhancement, since it only helps to improve performance by not losing useful information when it is discovered by accident.

Evolution takes place in an evolutionary algorithm as the operators are allowed to function on the population. There are a variety of evolutionary strategies, and parent selection methods that differ only in details. In all cases, individuals in the population are replaced by newer, better individuals, until some stopping condition is reached. The stopping condition can take any form such as a target fitness for an individual, slow or no evolution over a number of epochs, or a set time limit. When evolution ends, the population should all be good or reasonable solutions. This algorithm can be adapted for optimization if the best individual encountered, or in the population is considered an optimal solution.

Evolutionary algorithms as described here, also have the advantage that they are naturally parallel [1, 23, 39, 48] and can be implemented in distributed machines. There are many possible parallel implementations such as the island model, where small populations evolve separately and share information over some interval of time. Another option is to implement some expensive operations such as evaluation, mutation, or crossover on a dedicated machine. All of these operators are independent of one another, meaning the action of one will not affect any individuals other than the one or two involved.

2.3.2 Other Search Strategies

Random searching would be an appropriate technique for solving the graph partitioning problem as it is presented here. Partitions of the graph could be generated at random and examined in the method described in Section 4.6. This method can not be expected to produce reasonable results however, because of the low likelihood of randomly picking an appropriate partition out of the available search space.

Taboo search [18] is modelled after the human ability to remember where one is going and where one has been. It is an improvement on random searching, where a candidate solution is tested and changed if not suitable. The changes made are chosen at random, but controlled by a taboo list, so that the same change is not made and un-made resulting in cycling about the solution space. This technique however requires an unclouded understanding of what makes a solution, and how it can be changed. With consideration to the proposed problem, taboo search is inappropriate because of the changing nature of the input information. It is not expected that the input will be the same size, have the same solution changes available. In addition, the goal system will be capable of working in both text and line diagrams, which have some effect on the preferred solution changes and the taboo list.

Some techniques are modelled after observations of other living creatures. *Ant colony optimization* [37] relies on a population of "ants" to co-operate on finding an efficient route between two points. The ants all set off on different paths with the same goals, and a method of marking the world around them. The marking behaves like a pheromone that the other ants can detect, that wears off over time. When pheromones are detected, the ants have some preference for the path with the strongest scent, and will regularly choose that one. As time progresses the shorter paths become more heavily laden with pheromones, while the longer paths become unmarked and unused. Since it is primarily a shortest path strategy, this technique is not applicable to the problem at hand. However, this method may be appropriate if the ants could be placed at random and made to trace symbols. In this way they would not be able to leave the symbol that they exist on, and the groups of ants would have identified symbols. The ant-colony algorithm is only proposed, and not explored by the author due to the complexity of the ants tracing symbols requirement.

Some optimization techniques, such as *simulated annealing* [49] are modelled after real world physical processes. Simulated annealing attempts to emulate the behaviour of the formation of a crystal lattice into the optimal structure. In simulated annealing, solution elements are placed where they form a system with the lowest energy[1]. These elements can then either be removed from their current place or sealed in by new elements. This process continues, attempting to form a structure with the lowest energy. In terms of optimization, energy is simply a heuristic that evaluates the placement of a solution element. In terms of the problem presented here, a stroke that is near to, or connected to a character would have a lower energy when joined than a stroke that is far away, although there may be other factors to consider. Simulated annealing would be difficult to implement as a segmentation optimization since the energy evaluation would be uncertain when only partial characters have been found. No further consideration was given to this technique by the author because of the perceived problems with energy evaluation.

A few conclusions can be drawn about the search strategies discussed here:

- Each technique uses randomness in addition to searching short cuts or common sense.
- Undirected random searches are not expected to perform well.
- Taboo Search, and Simulated Annealing maintain a single candidate solution and use expensive operations to improve it.

[1] Energy is used as a heuristic evaluation.

- Evolutionary algorithms and ant colony optimization maintain multiple candidate solutions that can share useful information. Operations for improvement are usually less expensive since they are generally trial and error as opposed to carefully selected.

2.3.3 Why Choose Evolutionary Algorithms?

Based on the discussions above and in chapter 1, evolutionary algorithms offer a few advantages over other searching strategies:

- Form of the solution and the ability to use partial solutions. In different instances of the segmentation problem it may be easier to isolate the left-most, right-most, or any symbol in the middle first. The crossover operator will naturally preserve and collect the isolated symbols into more appropriate candidate solutions.
- Population of individuals. As mentioned above, some individuals may isolate different characters at the same time. It is also possible that some individual may isolate a symbol that is only part of a larger symbol, such as the AND gate that makes up the top of the character 'P'. Without high level knowledge, it is impossible to determine which is the correct symbol if they are both encountered. Maintaining a population of candidate solutions instead of a single one, allows for some means to consider both options when ambiguity is encountered as well as the ability to isolate distinct symbols simultaneously in different individuals.
- Modularity for Retargetability. The primary goal of this thesis is to introduce a flexible and retargetable approach to symbol segmentation. To achieve this, it was necessary to efficiently use domain specific information like symbol verification and established technologies when applicable. The independent symbol verification system can be easily replaced without affecting the operation of any aspect of the segmentation. If it was desired to implement some specialization of the system in a particular domain, any operator could be replaced with one that is directed by specific knowledge.

- Modularity for Maintenance. The evolutionary algorithm framework provides an efficient way to select, replace, or modify any operators or the knowledge contained in them. Any other operation such as initialization, mutation, or crossover can be modified with a minimum impact to the overall operation of the algorithm. Other techniques like Taboo Search, and Ant-hill optimization are inherently monolithic where a small change in the algorithm constitutes a large change in the operation of it.
- Parallel implementations. It is possible to implement most evolutionary algorithms, including this one, on a distributed machine, or network of computers. Small populations can be evolved under different constraints and share information based on some distribution strategy, such as an "island model", or whenever a new symbol is found. This option was not explored, because such a distributed system was not available to the author.

The combination of all of these factors has lead to the selection of evolutionary algorithms as the preferred approach.

2.4 Summary

The reader should now be familiar with the graph structure and OCR, as well as how they are to be used in this thesis. It is also assumed that the reader is familiar with the theory of an evolutionary algorithm, and the processes contained in one. More relevant background knowledge will be discussed in Chapter 3 regarding previous approaches to segmentation of both handwritten text and line images. These two chapters will provide a sufficient body of knowledge for a thorough understanding of the system introduced in chapter 4.

Chapter 3: Literature Review

This chapter is intended to provide the reader with some knowledge of the requirements of a document recognition system, as well as a background of previous segmentation approaches. Information will be presented in an ever narrowing view, beginning with Section 3.1 which is an attempt to identify the niche that segmentation occupies in a recognition system. When the role of segmentation has been clearly identified (Section 3.2), the major philosophies and boundaries between techniques will be defined (Section 3.3). Significant works are described for both character and line image segmentation in Section 3.4, followed by a short discussion on the applicability of past work to the development of a general approach.

3.1 Document Recognition.

The goal of any document analysis system is to interpret, understand, or catalogue the information contained on a printed page. This must be done at a number of levels, and in a bottom up approach. At the most basic, and deterministic level, a sentence can not be understood until each word is correctly identified, and similarly a word can not be identified until each letter in the word has been recognized[1]. This breakdown of the tasks for document understanding defines a sort of ladder, where the steps are must be accomplished in order for the goals to be realized.

[1] It is possible to recognize entire words as one symbol, but this is inefficient due to the number of words in a given language. Humans may do this quite regularly for short popular words such as "at" or "the" but it becomes more difficult with longer words like "conversation" and "conversion". Without considering the spelling, these words are almost indistinguishable in the way they appear on paper. The same can be said for keywords in a sentence, since changing the a single word, or the ordering of words in the sentence can change the meaning significantly.

At the bottom of the ladder is the digitization of the document image. This can be done accurately and reliably in a number of ways. Digital scanners and cameras may be used to collect off-line data representing the ink on the paper. Another choice is an electronic tablet that a user may write on with a special stylus to produce on-line data representing the ink as well as the ordering of all the strokes used. It is possible for most people to have access to these technologies in their home or office, yet it is utilized more often for pleasure than anything else.

For off-line systems, once an image of the document in question has been captured, the image can easily be processed with standard techniques such as thresholding or gray-scale segmentation [29]. This will produce a clean binary image, where the areas with information or ink are easily located. Processing of this nature is not necessary for an on-line document because the data tablet will introduce less noise and a sharp binary image, unlike a scanner or camera. Knowing where the ink is on the page, is far from knowing what information the document contains.

At this point, the information has been located, but not identified or understood. Typically, an OCR (Optical Character Recognition) system[2] would be used to interpret any areas of text, and a diagram interpreter used to process any non-text areas. This branching naturally introduces specialization into the general document recognition system. Any system must know before hand, what the document is expected to contain, whether it is typed text, handwritten text, in a specific language, or line images from a

[2] In this case, it is assumed the OCR system is capable of recognizing all text, not only clean isolated characters. Segmentation is a necessary step in OCR when characters are not clean and isolated.

specific domain. The system presented in chapter 4 provides a tool that can be used for both text and line images, improving the generality of the system as a whole. More will be discussed about recognition of text and diagrams in the following sections.

When each letter or symbol can be identified [2, 4, 27, 31, 40, 48, 50], the next step is to understand the meaning behind the ink. This step depends on the desired output of the system. In most cases, it is sufficient to store the information on the page directly to a database in some searchable form such as ASCII text, or object-oriented models. Converting digital information to a higher level, such as natural language interpretation, is a vast and open research topic, and no discussion will be presented here. For more information the reader is referred to [10].

3.2 The Role of Segmentation in Recognition Systems.

The term segmentation as applied to document recognition can describe operations at three different levels.

- Segmenting foreground text from background noise in an image. This can sometimes be done by thresholding, but often requires more sophisticated algorithms. The authors in [29] use multiple resolutions to distinguish ink from background marks and noise, and the method presented in [53] uses a thinning algorithm to locate regions of white space surrounding blocks of text.
- Segmenting text from images, and locating areas of related information on a page. Examples of where this would be necessary are bank statements, or directories of information. In both cases information with related content is grouped together, either in rows or columns. The system in [12] uses multi-resolution techniques to segment blocks of text that represent entries in a table. A method is proposed in

[58] to separate information that is useful at different levels in combined media images, such as distinguishing between diagrams and text.

- Segmenting information to help in understanding, such as segmenting characters in a string or symbols in a diagram, which is the focus of the remainder of this section.

In line images, such as script and some diagrams, a great deal of information is hidden inside the shapes of the curves, and the pattern they form. It is useful to be able to interpret the meaning of this information without first knowing what is expected to be in the image.

Traditionally, recognition of text and diagrams is handled in fundamentally different ways. Text on a page conforms to strict rules dictating it's organization and the placement of characters for a given language and alphabet. Segmentation of text may sometimes involve adjusting the baseline the text was written on [26] or normalizing other variables related to the information in the image. Diagrams on the other hand are drawn with all sorts of different notations and styles. They follow loose rules dictating layout, in such a way that they are most aesthetically pleasing, but the relative position of symbols may also contain information [47]. The remainder of the chapter will be focused on identifying the areas where we can unite approaches to the segmentation and recognition of text and line images.

Although traditionally treated as separate problems, both areas of recognition can be thought of as interpreting symbols that are connected. The properties of the connections between symbols are only important for the high-level interpretation of the meaning of the document. Thus, the role of segmentation in any recognition system is to isolate the symbols from one another and from noise in the image. This isolation may either be a division of the ink pixels into symbol and non-symbol groups, or locating the position and some information about the symbol that will help to identify it. To develop a general system that can perform segmentations of both character strings and line diagrams, it is necessary to examine both areas of research, and identify common useful approaches.

3.3 Factors to be Considered

Throughout the following case studies, systems will be described as either on-line or offline, and either straight segmentation or segmentation recognition [46]. The meanings of these and other terms are described below accompanied by a short discussion of their potential implications.

3.3.1 Segmenting Types of Information and Automatic Domain Identification

Some research is going on in the area of automatically segmenting images into useful areas. One application is separating text from the line images [58] so that they can be processed by separate systems. Another is identifying what language or a region of text belongs to [16]. Part of that work focuses on identifying whether a symbol is from an Asian or Latin alphabet and the rest of the system determines which language (English, French, Spanish, etc.) any Latin text is in. The system proposed in this thesis aims to eliminate this consideration, since text of any language, and diagrams from any domain will all be treated in exactly the same way, with the exception of a symbol classifier that will recognize symbols from a given domain.

3.3.2 On-line vs. Off-line

Data suitable for recognition has been examined in both the on-line and off-line forms. In on-line recognition, the writing is done on an electronic tablet that records both the x,y co-ordinates of the stroke, but also time information that indicates the stroke order. With off-line recognition, the only information that is available is the x,y co-ordinates of the stroke, and any temporal order must be inferred by other means [27]. On-line recognition is in general easier [27] if one assumes that a symbol is completely drawn before the next one is begun.

Since off-line data is essentially a sub-set of the data available on-line, any solutions to the problem in the off-line world are also applicable to the on-line one. This makes off-line recognition much more useful, since there would be no advantage to using on-line recognition for interpreting historical records and data when only an aged handwritten document is available. Off-line text segmentation is far more useful because a good system would allow automatic interpretation of all kinds of printed data.

3.3.3 Straight Segmentation vs. Segmentation Recognition

There is another, more fundamental choice that distinguishes segmentation techniques. That is the difference between straight segmentation, and segmentation-recognition. In straight segmentation, a connected string is surveyed and broken into regions with no knowledge of what the string contains, and there is minimal feedback and resegmentation initiated by higher levels of the recognition. With segmentation-recognition, the segmentation is often goal based, attempting to identify particular elements as a starting point, and working from there. Segmentation-recognition techniques normally have either a character matching system built in, or a high level of communication with the character matcher and document interpreter.

Straight segmentation is obviously much more general, since no high level knowledge is required. To perform straight segmentation ideally, one would not even classify the symbols before the action is complete. On the other hand, segmentation-recognition techniques often use information on the meaning of some parts of the image to help with the segmentation of others. This feedback can be anything from location of symbols and primitives that make up dashed lines [58] to the complex symbols found and expected size and position of the next one in mathematical formulae [17].

3.3.4 The Ideal System

Based on the above factors of consideration, the author proposes that the ideal system be:

- Capable of segmenting symbols from symbols given a clean binary image.
- Capable of segmenting both text and diagrams so that there is no need for an intermediate identification step, or the implementation of several different approaches in a single system.
- Off-line so that it can handle paper based documents of a historical nature.
- As close to straight segmentation as possible, to maximize the available generality. It is harder to define high level knowledge than low level knowledge, so the less domain and context information employed, the better.

Such a system is proposed in this thesis and is intended for use as a pre-processing step before high level recognition of a document.

3.4 Techniques for Isolating Symbols.

From this point on, segmentation should be thought of as separating symbols, as opposed to specifically separating characters or diagram elements. Literature will be presented that focuses specifically on either domain, and it will be shown that some methods for one are useful for the other.

3.4.1 Character Strings

No Segmentation

It is possible to perform some recognition without segmentation. The authors in [19, 20, 32, 33] claim to perform word recognition with their approach for reading ZIP Codes [19, 32], addresses [33], and words in a dictionary [20]. Entire blocks of text are first described as primitives, whether connected or not. The primitives are then represented by a vector. There is a similar vector for all valid words in a specific dictionary for the domain, and the entry that is measured to be closest to the input is selected as the word. One major drawback of this approach is that when it is applied to handwriting, different writing styles may produce different vectors for the same word.

A similar approach was used in [15] for recognition of typed Arabic script. The Arabic script is cursive in nature, and the typeset insures that all characters are well formed. The system then finds arrangements of primitives that could represent characters, and searches for calculates a probability that it represents each word in the dictionary. The word with the highest probability is selected as output, but no segmentation of the word is ever presented.

The off-line system in [34] extracts control points from unspecified features in a handwritten word. In a supervised training phase, each character in the alphabet is mapped with the control point representation, and placed in a database. This step is necessary to ensure that the isolated symbols will be recognized. In the process of recognizing a word, control points are first extracted from the word based on features in the line image. When the matching process begins, a probability matrix is constructed that holds some matching measures between the control points in the word, and those from the alphabet. The probability matrix allows the system to accurately identify enough characters in a word, so that the entire word can be known with the help of a dictionary. As described here, the system has a built in character recognizer, which would cause it to be classified as a segmentation-recognition system. This classification is not accepted because high level information is not used to direct any segmentation, and in fact no specific segmentation is ever performed. The pattern of control points is merely matched to previously stored patterns in a database.

Cutting the String (Straight Segmentation)

By far the most popular method for segmenting text, this technique is also quite intuitive. It can be summarized as the goal of finding a set of breaking points on a connected line diagram or string see Figure 9.

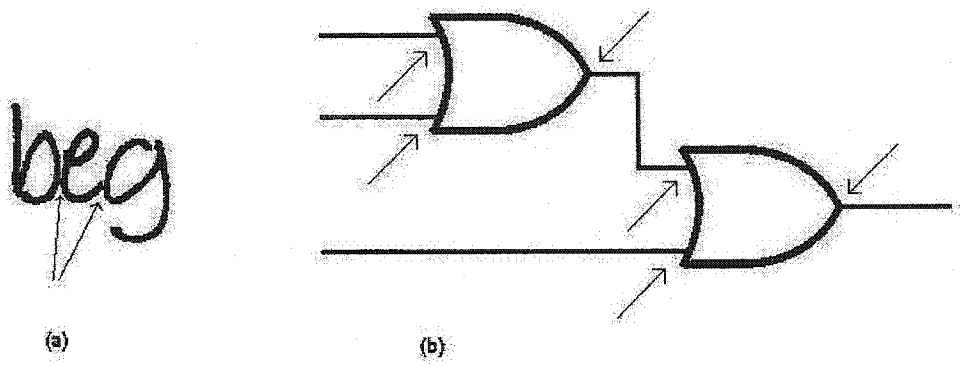


Figure 9 (a) Connected characters with 2 appropriate segmentation points indicated. (b) Connected logic symbols with 6 segmentation points indicated.

In terms of segmenting text, this normally involves finding a vertical line that separates two characters, where a segmentation point would be the intersection of the imaginary vertical line with any ink, Figure 9(a). For diagrams, one line will not be sufficient Figure 9(b), so this is could be considered very similar to *region-finding* methods which will be discussed later. The following systems can all be characterized as straight segmentation, since they have a low degree of contextual feedback aiding with segmentation. In some cases, the symbols are not even identified before they are isolated. Also, no article presented as *cutting the string* has a specific discussion of thinning or skeletonization, so it is assumed all techniques do not employ this phase.

Systems for segmenting text were presented in [3, 8, 21, 25, 41, 42, 45, 51, 52, 54, 57] that all focus on the same goal of locating and optimizing a set of segmentation points which surround valid characters. In all articles, some assumptions were made about segmentation points and the connected handwritten strings, such as:

- Segmentation points exist on horizontal lines, because text is a string of horizontally arranged symbols.

- Segmentation points are regularly spaced because all characters in an alphabet have roughly the same width.
- There was not a significant amount of extra non-symbol ink in the image
- Any two adjacent characters were only joined by one segmentation point (which is not always the case, however the harder *doubly-connected* case was never specifically discussed)

An optimum set of segmentation points can not be detected directly. Instead it is recommended to generate a set that represents an over-segmentation of the string. The over-segmentation contains points that may possibly segment a character into one or more regions. A final phase can then select the optimal sub-set from all possible points. Potential segmentation points can be located in a number of intuitive ways including:

- Directly from features in the image like contours [41, 42, 52], ANNs [45] and gray-scale information [61] or complex strategies like hit and deflect [57] and drop fall [54]. These methods are straight forward and generally the easiest to comprehend.
- Lexicon and grammar based approaches [8, 11, 35] that first describe the string as a set of primitives, and use a rule based approach to locate the segmentation points. This method is closely related to many popular character recognition techniques.
- Mathematical morphology and multi-resolution techniques [36, 51] that use complex algorithms to enhance features in the image that correspond to segmentation points. These techniques tend to search for features specific to the writer, such as lighter lines where the writer may have intended to lift the pen off the page.

Approaches that fall under the first bullet, using segmentation points located at features in the string are originally reserved for typeset text but are now being adapted to handwriting. There is a great deal of variability of the hand-drawn form, where histograms and rigidly defined features may change between different writers. The off-line system in [3] used this types of approach for segmenting handwritten zip codes. In that system, the number of digits in the string was known, and the character set was limited to only the digits zero through nine. Because of these restrictions, more specific features, and a fixed number of segmentation points could be defined adding efficiency. Segmentations are refined in a primitive way, where each sub string between each pair of segmentation points is submitted to a classifier. If the correct number of digits can be extracted the segmentation is correct, otherwise it fails.

In [41, 42] a special heuristic evaluation is used to determine the segmentation points on the connected string. This heuristic examines the shape of the curve, histogram profiles, neighbourhood, nearby segmentation points, and whether the current point is in a "hole" as in the lower edge of characters like "c" and "a". The heuristic is designed to add extra segmentation points, so that an optimal set can be extracted. The hypothesis is then optimized by using a specially trained ANN to select an appropriate set of segmentation points.

The off-line system in [52] uses the contours of the lines, in conjunction with observations of the Roman alphabet. The author notes that there are five different formations for the left edge of a character in either lower or upper case. They are straight left edge ("B"), slanted left edge ("A"), left concavity ("X"), left convex ("C"), and protruding left horizontal stroke ("t"). Based on these classifications, all possible curved cut lines are identified to the left of a character. A dynamic algorithm is proposed, that will compute a cost of any given segmentation. To compute this cost, first the centroid of each potential character is found, and from that the minimum cost segmentation lines for that character are found. The algorithm searches for a globally minimal cost of all potential segmentations and returns that as the correct result.

The hit and deflect [57] Figure 10(a), and drop fall [54] Figure 10(b) approaches are quite similar. They both operate off-line to separate numerals or characters in the Latin alphabet. With hit and deflect, a "projectile" or vector is initialized at some point above or below the connected string. The projectile is then "launched" towards the string, and is deflected when it hits ink pixels at an obtuse angle. The "projectile" then continues to deflect through the string, until it eventually is forced though to the other side. The path it formed will cross the string at some potential segmentation point. Drop-fall techniques operate in a more passive way. Similarly, the "projectile" or "drop" is initialized above or below the string and allowed to fall (either up or down, depending on where it is initialized) towards it. However, instead of deflecting when it encounters ink, the drop rolls along the contour until it falls into a local minima, where it proceeds though the ink. Performing this from different positions above and below the string will result in different segmentation points being generated.

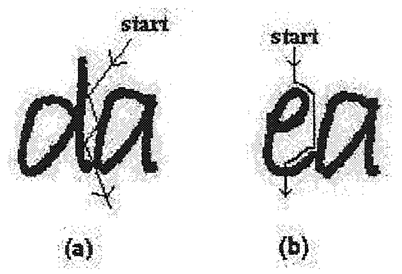


Figure 10 (a) The hit and deflect strategy. (b) The drop fall strategy.

Lexicon based approaches first describe a connected string in terms of symbol primitives, then group the primitives into valid characters as in [8, 20, 32, 33, 43]. In [8] a handwritten string is recorded on-line as parametric functions *x(t)* and *y(t)*. Inflections and extrema in these functions are located and their surrounding primitives are identified. A grammar is defined that helps in grouping the primitives into characters using a rule-based approach. A similar system is described in [17] that decodes the primitives directly to the valid characters, but further discussion of this technique is more appropriate as a *segmentation-recognition* technique.

Mathematical morphology has also been employed as a tool for segmentation. The work in [51] describes an off-line system for segmentation of cursive Arabic script which uses an opening (removes pixels, opening small gaps) operation and other image additions and subtractions to localize segmentation points. This method depends heavily on the features of the Arabic alphabet and the way characters appear when written by hand, such as vertical strokes being thicker than horizontal ones. The opening operation is used to remove all of the thinnest lines and dots. The remaining ink in the image represents *singularities* which occur at the thickest vertical lines in the image. These *singularities* are then subtracted from the original image to locate *regularities* in the image, which correspond to horizontal lines. The *regularities* are examined, and a number of rules are applied to locate segmentation points on them.

3.4.2 Line Diagrams

Some effort has been made to address line diagram segmentation with similar techniques. However, none of the assumptions made about segmentation points for text are valid when discussing diagrams. In fact, the opposite of nearly every assumption is true:

- Segmentation points can be on any side of a symbol, and any symbol may be joined to one, two, three, or more other symbols.
- Segmentation points are distributed in clusters around symbols, areas of the connected string may be devoid of symbols and segmentation points.
- A significant portion of the ink represents as non-symbol lines.
- Often there are two or more valid segmentation points between any two symbols, since there is at least one at each end of a connecting line.

Since these assumptions are all nearly the opposite, the goal of the system becomes somewhat different as well. In line diagrams, symbols are separated by lines, not points, hence bullet four. In the case of line diagrams, it is easier to think of a segmentation point as the point where symbol ink meets non-symbol ink.

An example of how string cutting must be modified for diagrams is given in [59, 60] for interpreting circuit and engineering diagrams where groups of lines, and not specific points represent the segmentation. It must be noted that one additional assumption was made in these two systems, that symbols are all made up of closed loops, as in a logical AND gate. The first step is to break the line diagram down into small lines, where each end is either an intersection with another line, or a free end surrounded by white-space. Each line is then labelled as either a *ps (probable symbol line)* or *pc (probable connection line),* and groups are formed from the *ps* lines, in hopes that they form some loops. If a *ps* group is near forming a loop, a *pc* or *ps* line may be relabelled and the groups reformed. When appropriate groups of *ps* lines are found, the shape they form is submitted to a symbol classifier. The process of relabeling lines continues until all the *ps* line groups are accepted as symbols. Any remaining unused lines are considered as connections between the symbols and are decoded as such.

Region Finding (Straight Segmentation)

Region finding is very similar to the above straight segmentation. The above techniques attempt to divide the ink on the page into small sections that represent symbols, and region finding attempts to divide the page or image into smaller images that represent symbols. Region finding involves placing a box onto the image in such a way that the box outlines a symbol. This technique is very difficult when symbols overlap or invade each others bounding boxes in other ways, which is often the case for handwritten text.

The work in [14] shows a system that could also be considered as a segmentation-recognition technique, but uses very little feedback on what's already been identified. In this system, a bounding box is initialized over an area of the string. An ANN (Artificial Neural Network) is trained as a detector to verify whether the string inside the bounding box is or is not a character. A second ANN is trained as a locator, that will translate and resize the bounding rectangle until it completely surrounds a known character. This system may be implemented to favour some symbols, and locate these first giving it some segmentation-recognition behaviour.

Feedback Segmentation (Segmentation Recognition)

It is sometimes impossible to determine a proper segmentation without contextual feedback on the image being segmented. This feedback could take many forms, ranging from an expected number of symbols, to a specific symbol to look for. This classification also describes the bulk of the line image segmentation techniques [17, 47, 56, 58], since most take advantage of specific aspects of the image layout and context to help form symbols.

An on-line system is presented in [24] for reading mathematical formulae. The stroke input sequence is used to generate a symbol hypotheses net (SHN). As they are input, each stroke is classified as either *primitive, standard,* or *complex.* The stroke sequence Figure 11(b) is then processed so that they can be confidently grouped with three or less temporally adjacent strokes to form a symbol, Figure 11(c).

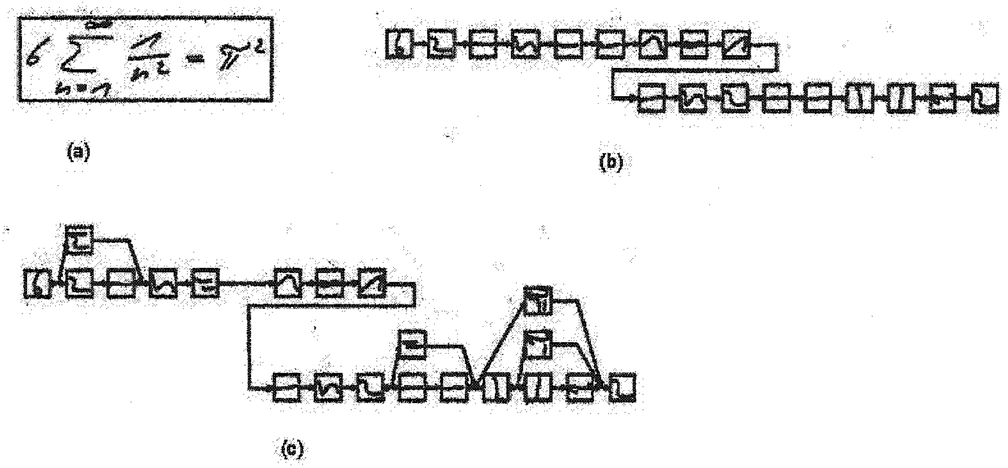


Figure 11 (a) A mathematical equation in off-line form. (b) The on-line stroke sequence that produced the formula in (a). (c) The regrouped SHN.

The stroke processing occurs at two levels. The first checks the validity of a symbol formed by a combination of the classifications above. Some groups may be rejected if the combination of classes within the group is not allowed. The second level of processing compares each stroke to those near it in the sequence. The comparison involves the distance between the strokes, the degree to which bounding boxes for the strokes overlap, and the relationship between the end points of both strokes.

The results from both levels of stroke processing are used to group the strokes into symbols. Some pre-recognition is used on strokes that meet special criteria so that some simple symbols such as "dots" and "minus" do not get grouped where they do not belong. This system is considered segmentation-recognition because it has the ability to go back and reorganize the groupings at a later stage.

A method for off-line reading of line diagrams like blueprints and bar-graphs that is based on grammar recognition is presented in [56]. The system first converts the image into a set of primitives associated with a position, and uses a specialized domain grammar to collect the primitives into sets of related items. The grammar dictates what types of primitives are supposed to belong to a particular group. For instance, if a group of primitives is intended to represent the scale marks on a bar graph, which are all short horizontal strokes, only strokes that are aligned vertically will be allowed into the group. In this method, no dashes from a dotted line or areas of text will be grouped as a scale mark. This system is retargetable by rewriting the domain grammar to allow and disallow sets of symbol primitives. As part of the grouping process, equivalence relations and other procedures are used to ensure the sets are maximal and as close to complete as possible. An interesting aspect of this system is that primitives are allowed to belong to multiple sets simultaneously allowing for situations such as in a blueprint where walls are a single line, but the line belongs to rooms on both sides.

Cutting the string methods, as described above may also have some weak segmentation-recognition behaviour. The system in [61] examines gray-level information, topographic features, and projection profiles to form theories of segmentation points. The topography, such as peaks and valleys give important clues about segmentation, which are either re-enforced or disregarded based on changing intensity levels. A graph search is then employed to further refine the segmentation theory. Finally the theory is tested with a recognition system. High level information is not used to define goals for particular segmentation regions. However, the author's claim that it is a recognition based segmentation method is justified because there is a stage where a tested theory can be redefined based on which symbols were recognized.

The term segmentation would also apply to the process of separating line images and text in the same image, such as labels on a map or drawing. This is the goal of the system presented in [58] which uses a run-graph representation of the image. The run-graph is a loose approximation of a thinned version of the image, and allows for easy identification of connected components and some symbol primitives like dots and dashes. The easily identified primitives can then be grouped into likely associations and removed from the graph, and the process can repeat using various levels of complexity in the primitives and groupings. In this way the symbols in the image are pulled away layer by layer of increasingly complex sets. These sets could represent symbol domains, since characters are complicated and would not appear in the same layer as a dashed or solid line that is part of a diagram. Recognition of the symbols was not discussed, however this is considered segmentation-recognition because of grouping of similarly complex symbols. This is equivalent to searching for a particular set of symbols first, and the order chosen may affect the final segmentation result.

3.5 Discussion of Previous Work

As discussed above, there have been a number of varied attempts to implement useful segmentation. Some techniques [11, 17, 35, 52] claim to function with over 95% accuracy in their intended domains. There is still some desire however to have a general system, since "such research would eventually result in a recognition system that is reconfigurable for various diagrammatic notations" [47]. As discussed in section 3.2, research is being done to automatically determine text and diagram boundaries, but this could be eliminated if the reconfigurable system could also operate as a text recognition system.

Segmentation-free recognition is highly specialized for domains of text with a strict vocabulary. One advantage is that the output word is more often correct since small errors in one segmenting a single character will not result in an erroneous output. These methods are expensive to run, since in general comparisons can be made with every entry in the dictionary. This approach is not applicable to hand-drawn diagrams because of their complexity and the fact it is unlikely that the same artist will produce identical diagrams each time.

It has been shown that segmentation techniques for text and line diagrams are sometimes related. In some ways, the separation of characters in a text string is a sub-problem of the separation of symbols in a line diagram, where the layout of the symbols is one dimensional and the non-symbol lines are not present. Straight segmentation techniques such as the *cutting the string* methods discussed are a useful starting point when looking for generality. Advanced techniques designed for modularity such as the heuristic function for placing segmentation points [41, 42] and lexicon based approaches [8] have potential to be retargetable, provided the heuristics and rules used for grouping were further developed. The system in [52] is highly specialized for the Roman alphabet, and as a result is not directly applicable for other alphabets or domains. Writing style may also have a great impact on the performance of this system since there are only a five ways characters can be divided.

Straight region finding techniques are often bound to fail from the start. For handwritten strings, in some cases it is impossible to draw vertical lines that separate characters, or a rectangle that surrounds a single character that does not contain parts of another. Positive recognition of such cases can not be done unless the intruding ink is removed. A system that was a hybrid of a region finding and ink removing approaches would no longer be considered region finding however, since the region inside the box has changed. In the case of line diagrams, if symbols are not square in nature, the same difficulties will be encountered.

Segmentation-recognition techniques [17] also claim good results, but are mostly for diagrams. However, they depend a great deal on the domain in which they are operating, and often take a long time and intensive development of rule systems. They will function extremely poorly if at all, when applied to other problems, because of the high level knowledge they use to direct segmentation. Since these techniques are primarily for diagrams, and the segmentation of text problem can be described as a sub-set of this, it would be useful to derive a general system from some of the strengths and abilities of these approaches.

Chapter 4: An Evolutionary Algorithm

4.1 The new system.

The system presented here is suitable for finding one or more valid segmentations of a connected string of characters or symbols. It must be possible to represent each symbol as a binary line image. Some suitable symbols include the Latin, Arabic, or Chinese alphabet, digits, digital and analogue circuit symbols, and elements on a map. Such symbols are easily recognizable by many symbol recognition methods including [31]. This system will take the form of an evolutionary algorithm as described in Chapter 2. With regards to the other segmentation systems presented in Chapter 3, this one is a close relative of the *straight-segmentation cutting the string* approach. Some *segmentation-recognition* behaviour is also employed, however no high level knowledge is used to direct segmentation, only an accept or reject rating on potential symbols.

The remainder of this chapter will be divided into six Sections describing the elements of the algorithm:

1. Individuals (Section 4.2)
2. Population Initialization (Section 4.3)
3. Mutation (Section 4.4)
4. Crossover (Section 4.5)
5. Evaluation, Termination and Aspiration (Section 4.6)
6. Evolution Strategies (Section 4.7)

When the algorithm is started, a population of *individuals* are created from the image of an input string. The details of this creation are controlled by the *population initialization* phase. Each individual is then *evaluated*, and if the *termination* criteria is satisfied, the algorithm is stopped and results returned. It is expected that the first *evaluation* will show that no useful symbols have been isolated, at which point the *evolution strategy* is employed to evolve the population. When no valid symbols have been found, *mutation* is the only option to change the features of the population. When two individuals have been *evaluated* and found to have distinct symbols isolated, *crossover* may be applied to create a new individual with two symbols isolated. This process will continue as directed by the *evolution strategy* until the *termination* criteria has been satisfied, and the algorithm stops. At any point in the evolution of a population, it is possible for the *evaluation* to accidentally stumble upon a new symbol that the individual has not isolated. To speed up the evolution, this accidental find is not forgotten, but instead it is made a part of the individual through *aspiration*. This *aspiration* sometimes allows the evolutionary algorithm to make large jumps towards a solution in the search space, that would have otherwise taken many epochs to evolve.

4.2 Individuals

One of the most important parts of an evolutionary algorithm is the form the individuals will take. Any suitable representation will have to meet the following criteria that were briefly described in Chapter 2:

1. The representation should be capable of representing all possible solution hypotheses, both feasible and infeasible. In the segmentation problem we should not rule out any possible symbols before they have the chance to be isolated, therefore all solutions are feasible.

2. The representation should be unbiased in the sense that all possible solutions are equally represented. This means that the representation should not have a higher

probability of representing one specific solution than it does at representing any other specific solution.

3. The representation should be easily exchangeable between the chromosone form and a form suitable for evaluation. It should not be an expensive operation to look at the hypothesis that the chromosone represents.

4. The representation should possess locality, so that a small difference between chromosomes reflects a small difference in the underlying solution hypotheses.

One representation that satisfies all of these criteria is a partition of an abstract graph (Figure 12). As shown in Section 2.2, a line image can be easily converted to and from a graph. The partition of the graph that will be used here is a grouping of the edges, so that each edge must belong to at least one group, and potentially more than one. It is necessary to allow the edges to belong to multiple parts to satisfy criteria 1 above, since in some cases two adjacent symbols may share one or more edges.

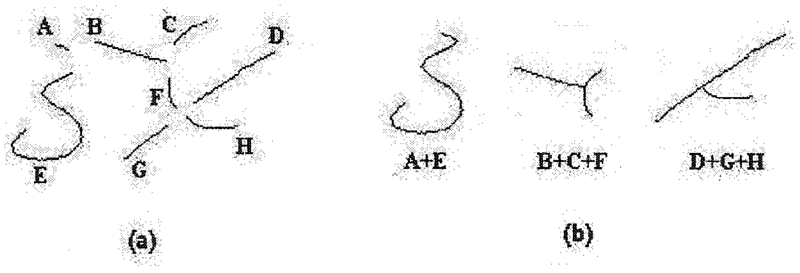


Figure 12 (a) The graph from Figure 8(c). (b) A partition of the graph in (a) with 3 parts.

In general, symbols will exist as single connected regions. This implies that any part that is not connected will be rejected as a symbol. Therefore any individual that contains a part that is not connected must be repacked see (Figure 13 (a)) so that all parts are single components (Figure 13(b)). Repacking an individual consists of locating any unconnected parts and removing one of the regions. This region will then be added to the partition as a new part, and is treated the same as any other potential symbol.

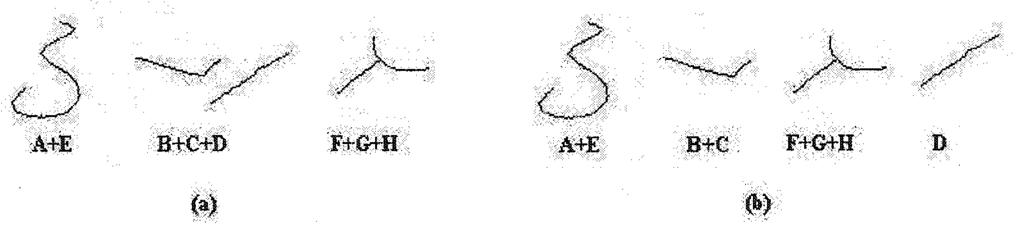


Figure 13 (a) An individual that requires repacking. (b) A repacked version of the individual shown in (a).

Information Loss:

Some information will be lost when converting the input image into a graph and then into individuals. A necessary step in the graphing procedure is the thresholding and thinning of the input image into a binary line image. The thresholding will remove all colour information, and the thinning may remove some fine details, including relative line thickness.

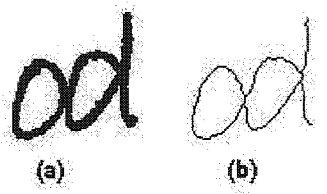


Figure 14. (a) Two thick touching characters, the line in the exact centre is twice as thick as the others . (b) A thinned version of the characters, where they now share a common line.

The graph representation still contains enough information to interpret the meanings of most symbols, but many times two thick lines that are touching or overlapping will be thinned into the same line as shown in Figure 14, or extra pixels may be left where they are not needed. The algorithm presented here is designed to overcome this loss of information, provided the symbol verification system can recognize the symbols in thinned form.

4.3 Population Initialization

The goal of the initialization is to create individuals that may evolve into feasible, high quality solutions to a problem. Therefore, to properly design an initialization routine we should look ahead at the behaviour of the operators, and also at any properties of the input that may be helpful.

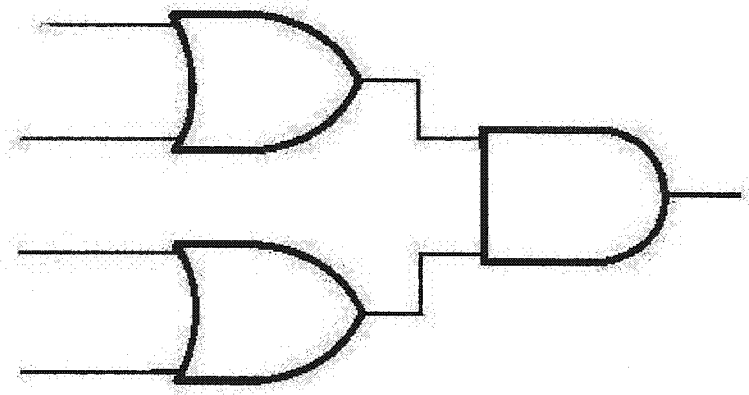
The mutation operator described in Section 4.4 is very good at shrinking large components. For this reason, it is ideal to have a complete symbol plus a few extra edges in one part of an individual. More accurately, it would be ideal to have each symbol appear at least once, with a few extra edges in part in at least one individual. This ideal case however can never be guaranteed since no knowledge of the image or it's source domain can be assumed before segmentation begins.

The *initializeIndividual* algorithm in Appendix A, is also designed to work on both character strings and flow diagrams. Characters are generally complex symbols arranged horizontally, while flow diagrams are generally simple symbols arranged horizontally and vertically. The biggest commonality between the two domains is the horizontal arrangement of symbols. For this reason, it was decided the initialization should try to break the string horizontally first, while letting the other operators handle any problems this introduces.

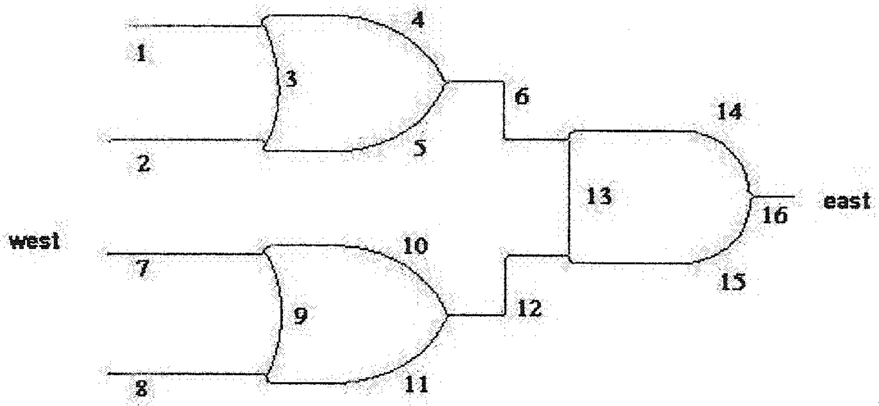
It is also important to have diversity in any population. To ensure diversity, it must be guaranteed that there will be a number of unique individuals. With the application of an algorithm for initialization, this guarantee becomes more difficult to uphold, and the algorithm must be strictly controlled. In this case, the algorithm is given a balance parameter *(p)* that will be explained soon, and uses Equation (1), where *popSize* is the number of individuals in the population, and *indNum* is the current individual (between 1 and *popSize*) that is being initialized.

$$p = indNum \,/\, popSize \qquad (1)$$

The first step in the initialization algorithm is to identify the eastern (N_{east}) and western-most (N_{west}) degree one nodes, see Figure 15. In a typical character string these will usually be at either end, as parts of different symbols. In some cases, the connected string may not contain any degree one nodes, so the eastern or western-most node will be used instead. In the case where only one degree one node exists, this represents both the eastern and western-most nodes.



(a)



(b)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
east	5	5	5	4	4	3	5	5	5	4	4	3	3	2	2	1
west	7	7	7	6	6	5	1	3	2	2	3	3	4	5	4	5

(c)

Figure 15 (a) An original circuit image. (b) A thinned and graphed version of (a) with the eastern and western-most nodes marked, and all of the edges are numbered. (c) The results of BFS from both the east and west nodes. Here the maximum depth from the east is 5 and from the west is 7.

After the nodes are identified, a breadth first search is performed on the graph, starting from N_{east}. The depth of each edge encountered is recorded as a list of integers *(D_{east})*, that is as long as the number of edges in the original graph *(G)*. The same is performed to find D_{west} starting from N_{west}. The maximum depth in D_{east} is recorded as d_{east}, and the maximum depth in D_{west} is d_{west}. If the edges are numbered, the results can be stored in a vector as shown in Figure 15 (c).

The parameter p was used to calculate the cut off depth d_{cut} for one segment in the individual.

$$d_{cut} = pd_{west} \qquad (2)$$

$$d_{cut} = (1-p)d_{east} \qquad (3)$$

If the parameter was below one half, then the west segment would form first using Equation (2), if it was above one half, the east individual would form first using Equation (3). In either case, all edges found with a depth less than or equal to d_{cut} were added directly to the appropriate segment. The other edges are assigned based on their position in the search order, with edges being assigned to the part that they are closer to, and a third made up out of the any edges equally close to either end.

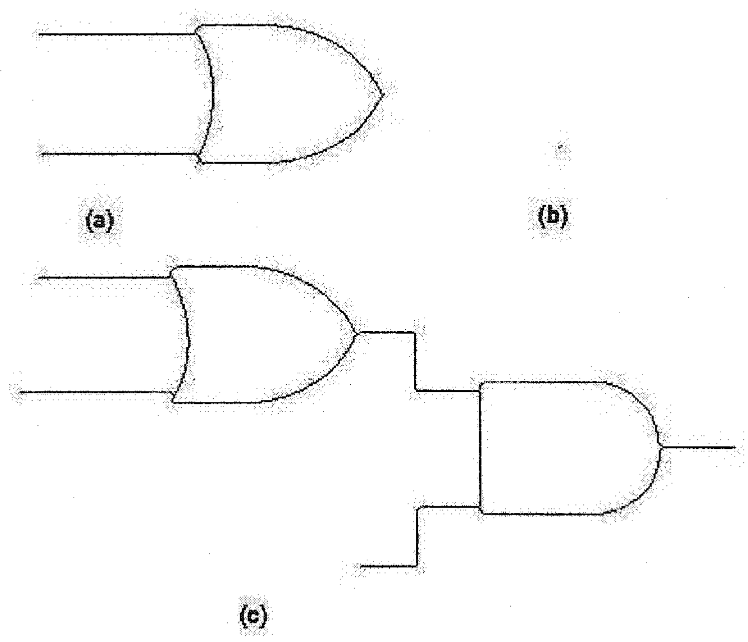


Figure 16 An initial hypothesis with 3 segments, (a), (b), and (c). Formed with balance = 0.45.

The individual in Figure 16 was formed using a balance parameter of p = 0.45. Thus Equation (1) is employed to produce a cut-off of $d_{cut}=\lfloor 0.45 \times 7 \rfloor = 3$ from the western-most node. The segment shown in Figure 16(a) forms first, representing the western portion, followed by the segment shown in Figure 16(c) which represents the eastern portion. The segment shown in Figure 16(b) contains a single edge between two nodes that are neighbouring pixels.

Similarly the individual shown in Figure 17 was formed using a balance parameter of p = 0.55. This individual forms from the eastern side with a cut-off depth of $d_{cut}=\lfloor (1-0.55) \times 5 \rfloor = 2$ from the eastern-most nodes. The segment in Figure 17(b) forms first, and includes the eastern-most node. The remainder of the graph is connected, and thus is collected into a single segment or part.

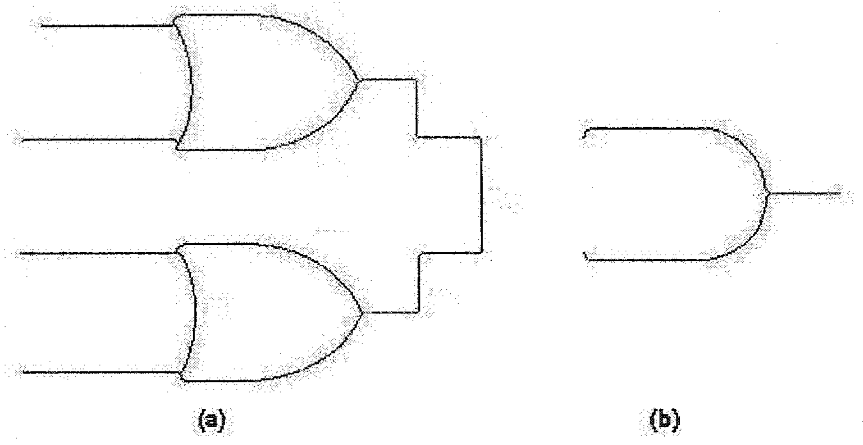


Figure 17 An initial hypothesis 2 segments, formed with balance = 0.55. (a) The western-most segment. (b) the eastern-most segment.

Since the p is a fractional number, and the depth of each edge must be an integer, it is possible that different balance parameters can produce identical individuals. The number of identical individuals will vary depending on the function that controls the value of p. Equation (1) produces a range of p values who's consecutive differences (p_{dif}) are shown by Equation (4).

$$p_{dif} = 1 / popSize \qquad (4)$$

Therefore, to ensure that all individuals generated from the same end are unique, we require that for individuals t and $t+i$ the minimum difference in d_{cut} values is 1, as shown in Equation (5).

$$d_{cut}(t+1) - d_{cut}(t) = 1 \qquad (5)$$

Substituting Equation 4.2.1(2):

$$(p + p_{dif})d_{west} - pd_{west} = 1 \qquad (6)$$

Therefore:

$$p_{dif}d_{west} = 1 \qquad (7)$$

Rearranging the terms we get:

$$p_{dif} = 1/d_{west} \qquad (8)$$

Now if we return to Equation (5) and substitute Equation (3) instead, we get:

$$(1-(p+p_{dif}))d_{east} - (1-p)d_{east} = 1 \qquad (9)$$

Which reduces to:

$$p_{dif} = 1/d_{east} \qquad (10)$$

Comparing Equations (4) and (7) reveals that to maintain unique individuals from the west we need:

$$popSize = d_{east} \qquad (11)$$

And similarly comparing Equations (4) and (10):

$$popSize = d_{west} \qquad (12)$$

This seems like quite a valuable result, but in reality is more like a guideline for choosing parameters than a strict rule. Since there is no way to predict the layout of the graph, or to ensure that there is no short path between the eastern and western-most nodes, there is no valuable relationship between d_{east} (or d_{west}) and the number of edges in the graph *(E)*. It can not even be assumed that d_{east} is equal to d_{west}. Also, in this scheme the formation flips from one end to the other at the half way point which introduces more chance for identical individuals to be formed in different ways.

A population requires some diversity, but evolution will still happen if there are some identical individuals. Therefore the only assistance Equations (11) and (12) can give us is to help predict the degree to which individuals will be unique. If *popSize* is greater than both d_{east} and d_{west}, from Equation (4) we would expect p_{dif} to be small, resulting in many unique individuals. Conversely, if *popSize* was smaller than both d_{east} and d_{west}

then there is a low chance that identical individuals will be produced. However, diversity will be lost and evolution made difficult if *popSize* is too small.

4.4 Mutation

In this evolutionary algorithm mutation is the primary operator, and can be performed on a single individual at any time. The mutation discussed here exists in two levels, however the second is only appropriate for cases when an individual has isolated one or more symbols. An algorithm called *mutation* that performs this operation is described in Appendix A.

The first level of mutation is rather simple, and aims to reduce the size of a connected part of a single segment in a single individual. The individual to be mutated *(Ind)* is generally chosen at random, but this decision is not part of the operator. The first step in a mutation is to choose a segment at random (S_{shrink}) that is not already matched. This segment is then surveyed to identify the set D_1 of nodes with degree one. If D_1 is non-empty then an edge (E_r) is chosen at random that joins a node from D_1 to the remainder of S_{shrink}. If D_1 is empty then E_r is chosen from S_{shrink} with no constraints. The edge E_r is then removed from the segment S_{shrink} and added to S_{grow}, an unmatched segment that is not S_{shrink} and has the highest index. This movement of an edge from one part to another is shown in Figure 18.

If E_r is adjacent to a node in D_1, the segment S_{shrink} will still be a single connected component. Conversely, if D_1 was empty, then it is possible that S_{shrink} has been broken into two components. In both cases there is no guarantee that S_{grow} will remain as a single component. Since we are only interested in symbols that are a single connected component (Section 1.2) it is necessary to repack the individual *Ind* so that all segments are single, connected sub-graphs. This repacking must be done after every mutation

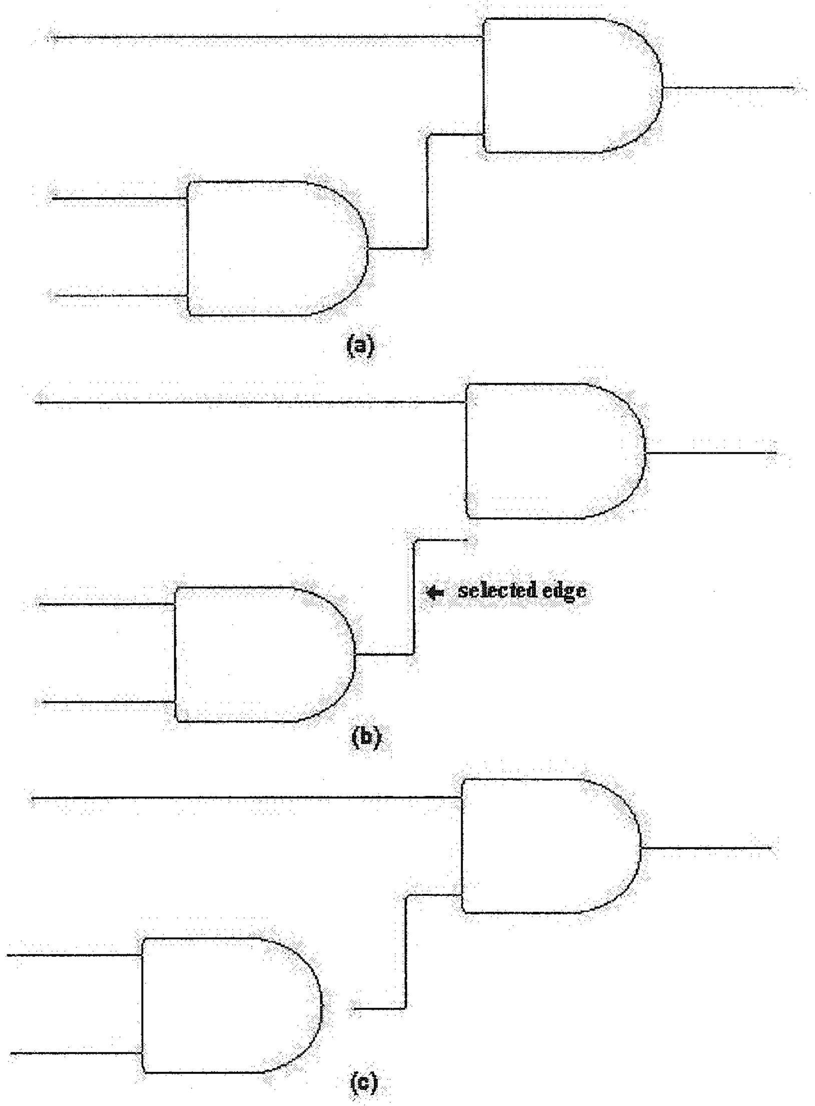


Figure 18 (a) A thinned circuit image. (b) A segmentation hypothesis on the circuit in (a) with the edge E_r marked. (c) A new individual after mutation using the edge selected in (b).

since there is no guarantee that the edge S_{grow} will be connected.

The simple mutation described above is suitable in all applications of this evolutionary algorithm. There are no restrictions placed on the choice of segments or edges involved, except those described above. This operator also uses no information from the evaluation of *Ind* or any other individual in the population. This operator works blindly to shrink segments down until a symbol is found somewhere within, and thus performs better when a single symbol is contained in a segment. However there is no way to determine or control the number of symbols contained in any single segment in an individual, so a modification is described below to improve performance in some cases.

The enhancement to the simple mutation, dubbed as "second level mutation", operates on the assumption that symbols belonging to a specific domain occupy the same area, or more specifically are made up of the same number of pixels. Since this assumption is not completely accurate, a parameter (p_{close}) is defined to act as an uncertainty in size, that is restricted to the range between zero and one and is determined experimentally. Using this parameter, Equation (13) describes equality of area between two symbols with areas a_1 and a_2.

$$a_1 = a_2 \Leftrightarrow a_2 - (p_{close} * a_2) < a_1 < a_2 + (p_{close} * a_2) \qquad (13)$$

A second level mutation may only be applied to individuals that already have at least one segment matched. In this case the average area of all matched symbols $(a_{matched})$ is calculated. At this point, one of two branches is chosen at random. The first branch employs a shrinking mutation similar to the one described above, provided that S_{shrink} has an area (a_{shrink}) and (14) is true.

$$a_{shrink} > a_{matched} - (a_{matched} * p_{close}) \qquad (14)$$

The second branch of the second level mutation combines two adjacent segments into a single larger one. This branch behaves similar to a small crossover operator, since suitable parents must be selected before a child segment may be created. Suitable parents (p_1 *and* p_2) are chosen so that (14), (15), and (16) are all true.

$$p_1 \textit{ has area} < a_{matched} + (a_{matched} * p_{close}) \qquad (14)$$

$$p_2 \textit{ has area} < a_{matched} + (a_{matched} * p_{close}) \qquad (15)$$

$$p_1 \cup p_2 \textit{ has one component} \qquad (16)$$

The segments p_1 and p_2 are then removed from *Ind* and the new segment, $p_1 \cup p_2$ is added. There new segment is expected to have a greater area then $a_{matched}$ but there is no guarantee. In either case, whether it is smaller or larger then the average symbol, mutations will still continue as normal on this segment.

4.5 Crossover

In this system the crossover is an operator that is used to share information between individuals in the population, and is described in Appendix A, as the *crossover* algorithm. A crossover may only be applied to two individuals *(Ind_1 and Ind_2)* that are compatible. Equations (17), (18), and (19) define compatibility given that M_1 is the set of all edges from Ind_1 that belong to any matched segment. Similarly M_2 is established from all edges matched in Ind_2.

$$M_1 \neq \emptyset \qquad (17)$$

$$M_2 \neq \emptyset \qquad (18)$$

$$M_1 \cup M_2 - M_1 \cap M_2 \neq \emptyset \qquad (19)$$

The action of the crossover operator is simple compared to the mutation operator. The goal of a crossover, as described above, is to share information. To do this, a new individual is created *(Ind_{child})*, initially with no segments. All matched segments from Ind_1 and Ind_2 are copied directly into Ind_{child}.

After the matched segments are copied directly, a list of the unassigned edges is made *($M_{unassigned}$)*. Each edge in this list is then assigned to a new segment *($S_{unassigned}$)* in Ind_{child}, see Figure 19. Since there is no guarantee that all of the edges in $S_{unassigned}$ will form a single connected component, the new individual, Ind_{child} must then be repacked.

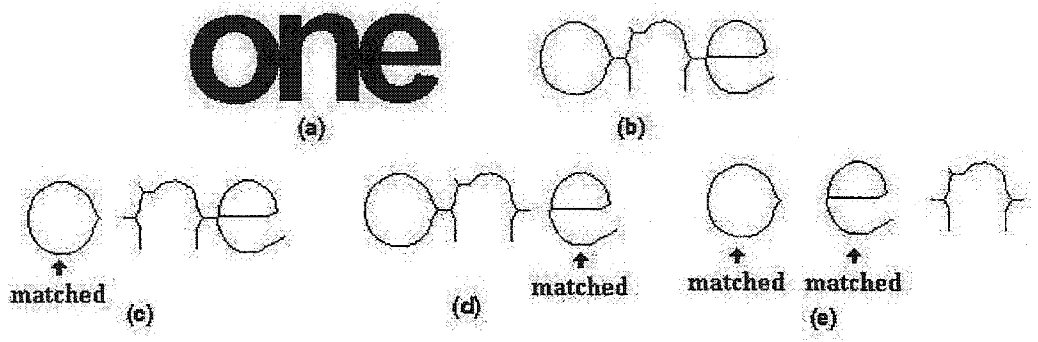


Figure 19 (a) An original connected string. (b) A thinned version of (a). (c) An individual who has matched the character 'o'. (d) A different individual whom has matched the character 'e'. (e) A new individual resulting from the crossover of the individuals in (c) and (d).

The new individual Ind_{child} now has been completely formed. This individual is guaranteed to have more edges included in matched symbols since the parents would not be compatible if this were not true. It is possible however, that the one of the new symbols is merely a sub graph of the other. In this case, the symbols are left alone, since it is possible to gain multiple valid segmentations from the same connected string, and there is no way to accurately determine which is the more correct symbol.

4.6 Evaluation, Termination, and Aspiration

4.6.1 Evaluation

In most evolutionary algorithms an individual will be given a one dimensional evaluation or rating. This is appropriate when individuals aspire towards a single identifiable goal, and can be ranked based on their distance fom it. The segmentation problem examined here is much more complex, and it is difficult to say that any one individual is absolutely better than another one. This is because different individuals may isolate different symbols first, and this can not be accurately represented with a simple ranking.

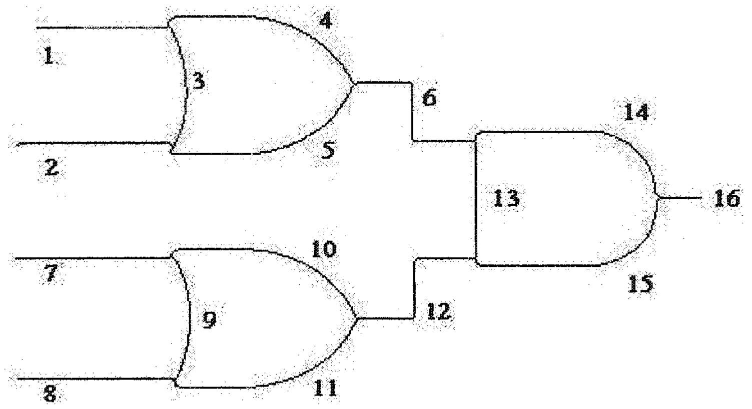
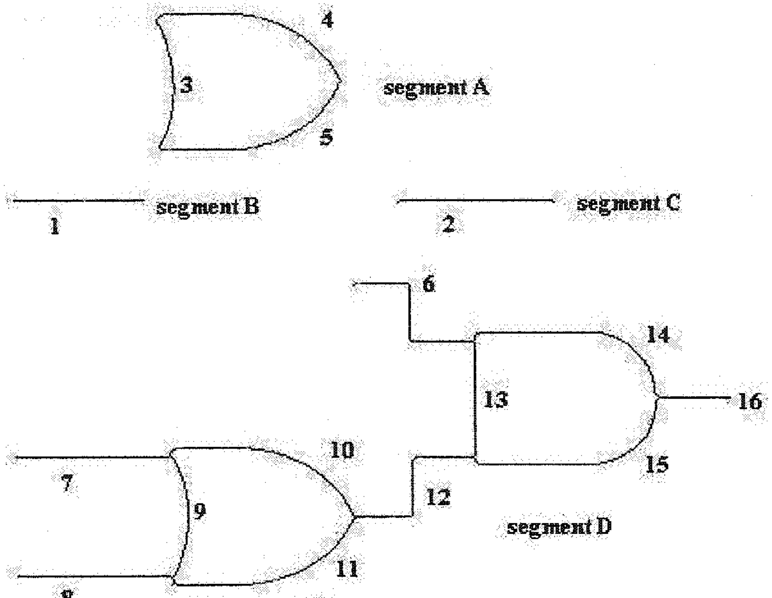


Figure 20. A graph of a circuit image with the edges numbered.

In this evolutionary algorithm, a binary vector ranking is used. First, the edges in the graph are numbered (Figure 20) and the vector is created with a bit for each of the edges in the graphed representation of the connected string. In an individual that has no symbols matched, this binary vector contains a false in all bits. When an individual succeeds in isolating a symbol, the bits associated with each edge involved in that symbol becomes true. In this way, all edges used in at least one symbol are given a true value in the binary vector.



(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
segment A			T	T	T											
segment B	T															
segment C		T														
segment D						T	T	T	T	T	T	T	T	T	T	T

(b)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		T	T	T											

(c)

Figure 21 (a) An individual with 4 segments, segment A is a matched symbol. (b) The binary representation vector of the individual in (a), false values are shown as blanks. (c) The binary evaluation vector of the individual in (a), false values are shown as blanks.

This binary vector can then be used to quickly identify all individuals with at least one symbol matched and give some information about that matching. The evaluation vector can also be used to quickly test for compatibility as required for the crossover operator. The vectors from the two potential parents are input to an inclusive OR function, and a resultant vector produced. If the resultant vector contains more true bits than either parent individually, an offspring produced by crossover should contain either larger, or more numerous symbols than the parents.

4.6.2 Termination

An evolutionary algorithm may terminate for any number of reasons. A popular default termination condition is a maximum number of evolutionary epochs or running time. This will ensure that the algorithm will always finish, regardless of the solutions found. It would be preferable to have the algorithm terminate when every symbol in the string has been found, but this is difficult to detect in such a general system. If the number of symbols were known, as in a postal code or telephone number, the number of symbols would be quite useful. In this system however, no knowledge of the input is assumed, except that the symbols can be recognized. It is also impossible to determine what portion of the edges, or connected string area will be used in the final solution.

There are any number of possible termination criteria that could be defined. It is intuitive to think there is some relationship between the number of edges and the number of symbols, but this is not the case. In some instances symbols can be made up of as few as one or two edges, and in other cases they can be very complicated and made up of many lines. Such variance may appear in a single image, based on the amount of noise or the domain of the symbols present.

If the symbols are generally connected by extra lines, a termination condition similar to the second level mutation can be defined. The non-matched segments are merged into maximal connected segments, and the areas of the matched symbols and the largest non-matched segment are be compared. If the non-matched area is smaller than the symbols, it is unlikely a new symbol will be found, and the algorithm can terminate. This termination criteria is not absolutely reliable however, since there is no distinction made between cases with touching symbols, and cases were symbols are joined by extra lines. Symbols that are touching will tend to cause the maximal non-matched segments to be fragmented whenever a shared edge connects two portions of an unmatched symbol.

4.6.3 Aspiration

When performing the evaluation new edge groups are formed temporarily. These groups represent the largest, non-matched connected regions. The possible termination criteria described above examines the relative areas of these sub-graphs, but no consideration is given to the symbols these regions may potentially form. When performing the evaluation and checking the termination condition, the new potential symbols are submitted to the symbol verification system. If a positive result is ever obtained, indicating the maximal connected region represents a symbol, the individual being examined undergoes an aspiration mutation, storing a segment representing the newly found symbol.

This aspiration mutation is more likely to take place in domains where the symbols are close or touching like character and digit strings. In line diagrams, the symbols are generally accompanied by extra line segments that can not be classified as part of any symbol and the positive result will never be returned.

4.7 Evolution Strategies

There are a number of population replacement strategies that are possible for evolutionary algorithms. The choice between these has a greater impact when a choice is necessary to maintain a strict population size. In the system presented here, individuals are evolved primarily through mutation which requires only a simple strategy, the mutated individual will replace the original.

When crossover is applied to create a single individual from two parents, it becomes necessary to eliminate one individual from the population. Since crossover maintains all useful information from both parents, nothing will be lost if either parent is eliminated. It is expected that both parents will become obsolete once they undergo a crossover, since there is now an individual in the population that has identified the same segment as them, plus additional segments. In fact, if individuals are allowed to identify the same segments as each other, crossover will happen a few times without any mutations to spread the new symbols throughout the population. In this way, any individual that has identified a symbol independently, with only mutations on it's original form, will be given all of the symbols that the rest of the population has found.

It is always desirable to avoid premature convergence, or to limit the number of individuals that identify a particular symbol. As described above, with no restrictions, any individual that identifies a single symbol, will quickly be evolved to identify the same symbols as the rest of the population. This will cause the evolution to converge on a set of symbols, that may or may not be the correct one. In many cases, one symbol may appear as a part of another, such as a the digit '1' inside the character 't'. The convergence described above may force all individuals to locate only the '1' and none will ever locate the 't'. This is a very expensive operation to implement, since it would require comparing all of the matched symbols from each individual to count the number of repetitions.

Chapter 5: Results

5.1 Data Sources

The current system differs from previous research in the number of functional domains. Previous systems focused on small, well defined sets of data input such as mailing addresses in the US postal service, or logic dagrams consisting of certain gates. In such systems, large databases of available input can be collected, such as the CEDAR postal database. In general development of these systems was focused equally on segmentation of touching symbols and recognition of isolated ones.

In the current system, it is assumed that the symbol verification system is complete, and capable of recognizing any isolated symbol in appropriate domains. In reality, the system employed relies on a set of pattern vectors to identify symbols, with more patterns, more symbols can be recognized. To ensure maximum performance of the segmentation system, the recognition system must make as few inappropriate rejections as possible since such a rejection would cause the segmentation to fail in that case. For this reason, data sets are limited to standard symbols, such as characters from a font, or graphic symbols from a typeset, that are arranged and connected in various ways. Segmentation focuses on breaking the connections, and it is of little consequence what the symbols are or represent.

Data from public sources such as the CEDAR database contain too many variations of symbols, which would require time and effort to ensure the symbol verification system could recognize all symbols it will encounter. Test data will be collected manually into two distinct sets:

- Thick, touching characters. Due to the connected characters, thinning will produce minor variations of the skeleton patterns. Samples include two, three, and four touching characters from the 72 point Bradley Hand ITC font generated using techniques described in [57]. See figure 20, figure 21, and figure 22.
- Scanned logic circuit images from [55]. These circuit diagrams are all of a common typeset, with various numbers of gates and layouts. Effort was made to include all logic symbols including AND, NAND, OR, NOR, XOR, NOT, and various negative input gates. Isolated symbols are considered to be any of the gates listed above, but no specific patterns of negative input gates were included. See Figure 32 and Figure 34.

The data sets chosen are designed to show the feasibility of the system, and provide performance analysis that is useful to examine the scalability of the proposed system.

5.2 Investigations

There were two separate investigations of the performance of the algorithm. In Section 5.2.1 the parameters of the algorithm are kept constant, and a number of trials are performed on a wide variety of input images. There were 198 touching character and circuit diagram images. Initialization was always done using the *initializeIndividual* algorithm in Appendix A, and Crossover was performed on every pair of compatible parents. The population size was kept constant at 10 individuals, and evolution was terminated after 50 epochs for all trials. Results for this investigation are contained in Appendix C1, Appendix C2, and Appendix C3.

In Section 5.2.2, the parameters and other variations to the algorithm are tested. There are fewer input images, all of which are small sections of the largest image. Trials were run on all images varying a number of parameters. For all images, all combinations of the following were performed, with the evolution terminated at 50 epochs:

- Seeded initialization using *initializeIndividual* and random initialization.
- Performing every possible crossover and randomly selecting individuals for compatibility tests.
- Population sizes of 5, 10, 15, and 20.

An additional investigation was performed on a small subset of the images contained in Appendix C, with all of the same variations mentioned above, and the evolution terminated at 100 epochs. Data was also collected on the time in seconds required to perform the segmentation for all investigations in Section 5.2.2. Results for this investigation are contained in Appendix D1, Appendix d2, and Appendix D3.

5.2.1 Performance on varied input.

5.2.1.1 Discussion of the Investigation

The proposed system is an based on an evolutionary algorithm, and therefore is not guaranteed to produce the same results every time. The algorithm is also meant to be used in a variety of domains, without the need for change. To investigate the algorithms performance in the general case, segmentation will be performed on a large number of varied input images. The algorithm were run 100 times on each of the 198 touching character and circuit diagram images contained in Appendix B. The following four factors are considered in this investigation.

- Accuracy and consistency, relating to how close and how often the solution is to the ideal segmentation. Accuracy is related in classes described below:
  - Class 0: All symbols correctly identified in all locations.
  - Class 1: All locations of symbols identified, but one or more symbols may be incomplete, such as OR being identified instead of XOR.
  - Class 2: One symbol location missing.
  - Class 3: Between two, and half the total number of symbol locations missing.
  - Class 4: More than half of the symbol locations missing.
  - Class 5: No symbols isolated.
- Initialization, relating to how long the algorithm has to work before the first symbol is found. Initialization takes the form of a distribution over the epoch when the first symbol was found.
- Termination, relating to how and when the algorithm terminates. There are two possibilities, either termination criteria is bad and the algorithm will run to an epoch limit in which case it is forced to stop, or the termination criteria is good and the algorithm will stop naturally with some confidence that it has a correct segmentation. Termination takes a similar form to initialization, a distribution over the epoch when the algorithm stopped. If the algorithm times out in all cases there will be a sharp spike over the maximum epoch allowed.
- Duration of evolution, which is the time between initialization and termination. It is important to examine duration on a case by case basis to draw some conclusions that are independent of the performance of the initialization. Duration is a distribution similar to termination and initialization.

It is very difficult to compare results obtained from such varied input. To compensate for this, accuracy is not considered as an absolute value, such as the number of symbols found. Instead, the accuracy is rated as a relative value, in one of the five six classes described above.

All of the data collected in this investigation is contained in Appendix B1 for the accuracy and duration of evolution data, and Appendix B2 for the epoch of initialization and epoch of termination data. It was necessary to split the data into to appendices to present the graphs in a large enough format. The splitting of appendices is organized so that graphs of related data can be easily compared, with the trade off being that all data for one input image is separated.

An exampk of the data that is presented in Appendix B1 is shown in Figure 22 and Appendix B2 is shown in Figure 23.

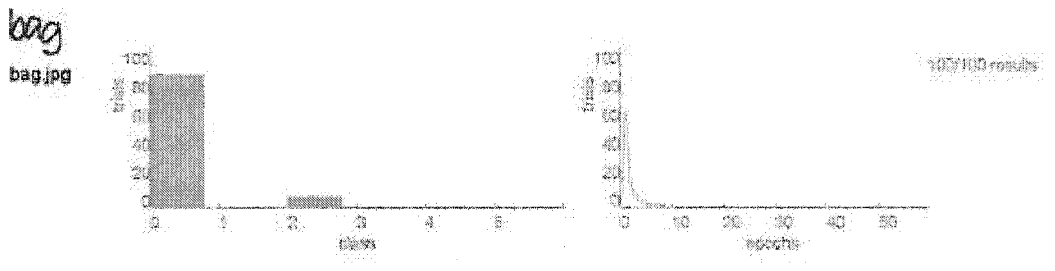


Figure 22. A sample of the data presented in Appendix B1. On the left is the input image of touching characters. In the middle is a bar graph representing the accuracy rating. On the right is a line graph representing the duration of active evolution.

In the sample shown in Figure 22, the proposed system was executed 100 times on the input shown, and performed very well in all trials. The majority of class 0 results indicate that it was usually successful at locating all correct symbols 92 times out of 100. There are a few class 2 results indicating that occasionally it missed a symbol.

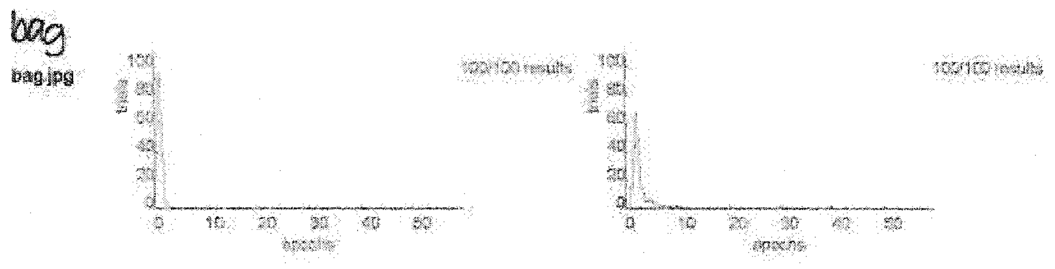


Figure 23. On the left is the input sample. In the centre is a line graph representing the distribution of the epoch when the first symbol was found. On the right is a line graph representing the distribution of the epoch of termination.

Examination of Figure 23 shows the initialization algorithm performed exceptionally, usually locating a symbol in the first 2 epochs. There were no instances of termination after epoch ten, indicating the algorithm never reached the epoch limit. Therefore, the class 2 results occurred when the termination criteria was satisfied prematurely. Figure 22 shows the duration of useful evolution, between the time when the first symbol was found and when the termination criteria was satisfied. This chart shows a sharp, narrow spike around 3 epochs, indicating that the algorithm quickly and consistently located the remaining symbols until termination.

5.2.1.2 Discussion of the Analysis

The metrics described above will be applied to the results produced from a number of trials on a single input. This is not sufficient to show the feasibility of the system since results must be compared from all data sets, however, it is difficult to identify simple ways that segmentation of a pair of touching characters can be related to segmentation of a logic circuit with four gates. The problem lies in the number of domains of the input, and the fact they do not share many common features, but the graph representation of the inputs provides some help.

The major factors affecting how well the system will perform are the complexity of the layout of the symbols, the number of symbols present, and the complexity of those symbols. It is impossible to evaluate the layout (or segmentation would be unnecessary), or the number of symbols before segmentation is done. One metric that provides some insight into the overall complexity of the input is the number of edges the graph representation contains, since instinctually it would be harder to identify the appropriate partition in a larger graph. For this reason, the number of edges in the graph is used as a ranking heuristic to appropriately ignore the domain of the input and compare all data sets.

To summarize all of the inputs into a single chart for comparison, a characteristic point is determined for each some of the distributions described above. For good results, where class 0 and class 1 results were produced, the 'x' co-ordinate of the characteristic point represents the epoch associated with the mean epoch of the distribution, and the 'y' co-ordinate is the number of edges in the input. Points extracted from all inputs can then be placed on a single set of axes, and useful conclusions drawn from this graph. A sample for demonstration purposes only is shown in Figure 24.

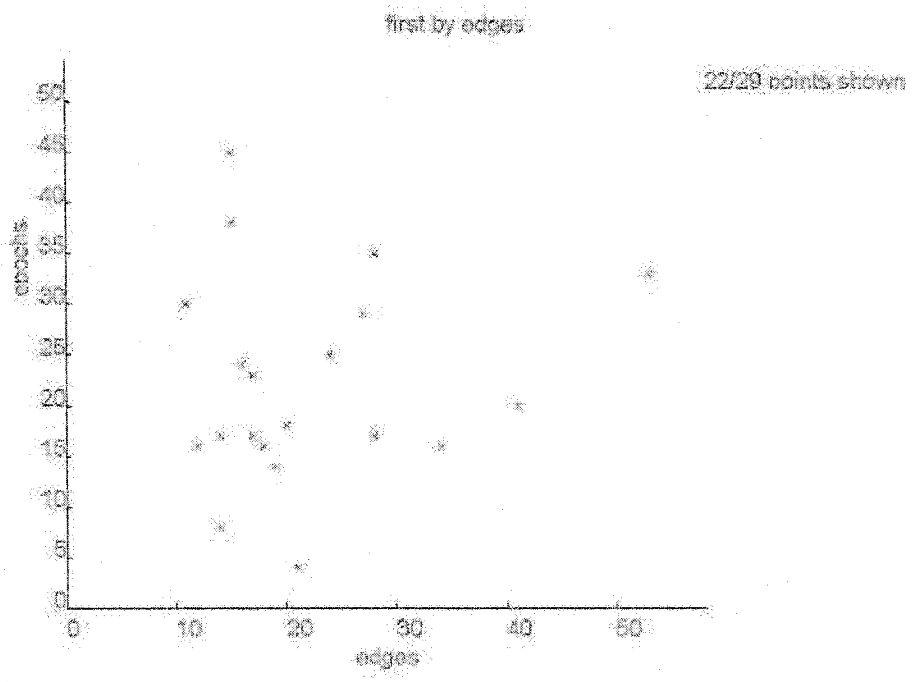


Figure 24. In this sample chart, initialization results from 22 of 29 input images are summarized, as indicated in the top right corner. The x-axis represents the number of edges in the input, so results from inputs of more edges appear farther to the right. The y-axis is now the number of epochs initialization took to find the first symbol.

5.2.1.3 Discussion of the Final Results on Varied Input

A few samples of the final results are explored here, and the remainder are presented in Appendix B. It will be left to the reader to examine the results in appendix A in greater detail. The following examples are meant to illustrate some of the most interesting behaviour of the proposed system, and are not representative of the results as a whole.

Analysis of Single Inputs

The experiments performed on touching character data were interesting in a number of ways. First of all, there were very few patterns stored for the characters, with one pattern each except for 'd' and 'f' which had two variations each. As we saw in figure 22, the touching character pattern 'bag' produced very good results. The majority of trials produced class 0 (Figure 25(b)), or perfect results with a few class 2 (Figure 25(c)) results as well.

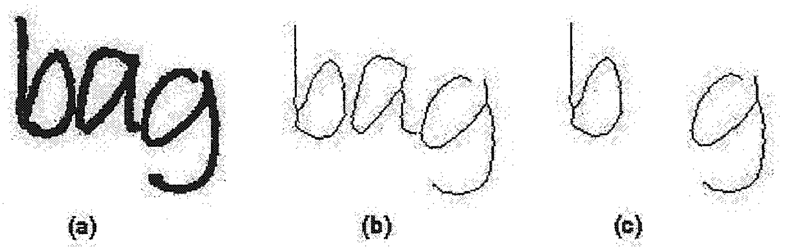


Figure 25. (a) An original touching character sample. (b) A class 0 segmentation result. (c) A class 2 segmentation result.

In at least one case only class 2 results were obtained, a symbol could not be identified. The symbol could not be found in isolated form because there were no partitions of the graph that could be reduced to only the symbol lines. There was a single edge that represented a significant amount of two symbols, namely 'c' and 'f', as shown in figure 26 (d).

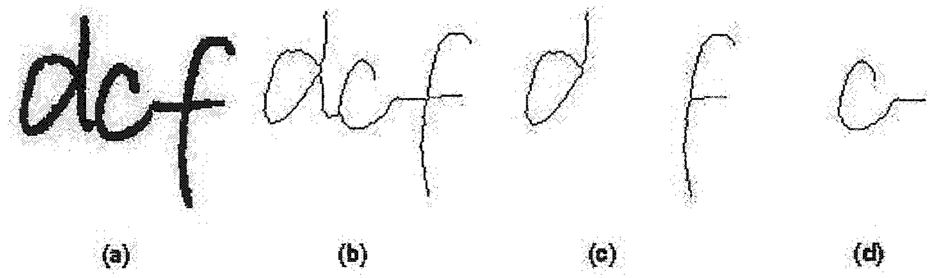


Figure 26. (a) An original touching character sample. (b) A thinned version of the characters in (a). (c) Only class 2 results were obtained for this input. (d) The largest connected region. It is two edges, separated where the 'd' and 'c' meet in (b). The large tail can not be removed to isolate the character 'c'.

It was noted earlier that there are two patterns for both of 'd' and 'f'. Accepting variations on a symbol is the responsibility of the symbol verification system, and can serve to help improve both the accuracy and speed of segmentation. In Figure 27 we see an input string of the characters 'ddf', and two different class 0 results (Figure 27 (c) and figure 27 (d)) produced by segmentation. It is easy to see the two different patterns for 'd' in Figure 27 (d), as well as the two different patterns for 'f' in Figures 27 (c) and (d). The full pattern for 'd', with a tail, was never found for the first 'd' because in the thinned image, Figure 27 (b), this tail forms a dramatic hook at the end. This pattern is varies too much from what a 'd' should look like and was rejected by the symbol verification.

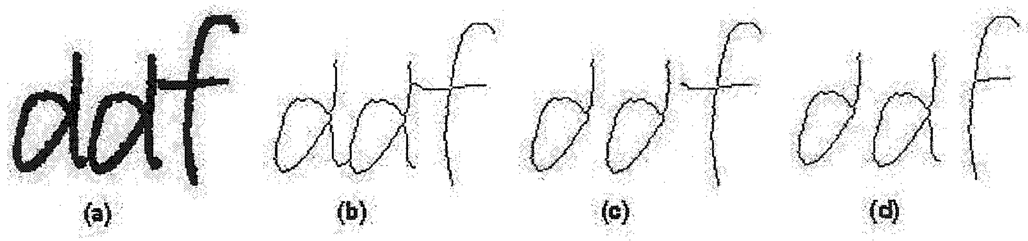


Figure 27. (a) An original touching character input string. (b) A thinned version of the string in (a). (c) A class 0 result from segmentation. (d) A second class 0 result from segmentation.

First we should examine the overall success of the proposed system. This can be done by combining all of the accuracy results into a single accuracy bar graph, as shown below.

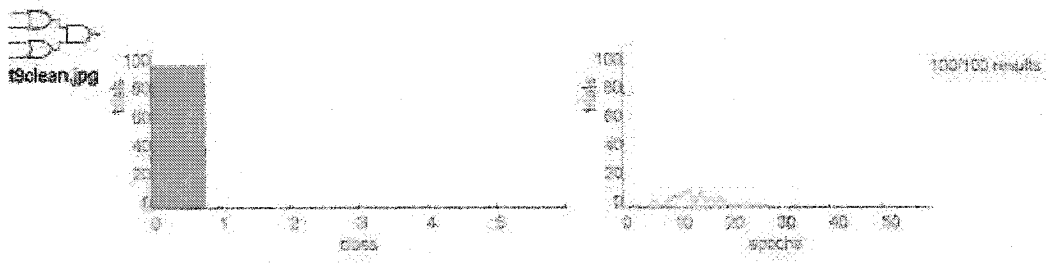


Figure 28. A circuit image input. All results were class 0. The duration distribution was wide, but all trials terminated before 35 epochs.

Throughout chapter 4 we used a simple circuit diagram, shown again in Figure 28, as a demonstration for initialization, mutation, and crossover. All trials on this input produced perfect class 0 results, similar to figure 29(b).

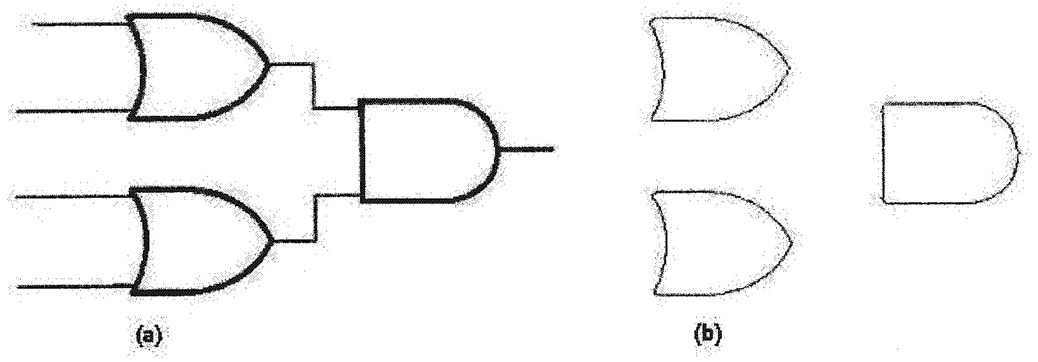


Figure 29. (a) A sample line diagram input. (b) All segmentation results were class 0.

We will also take this chance to discuss the performance of experiments on this image as shown in Figure 29. From Figure 28 we can confirm all results were class 0, which is ideal for this system.

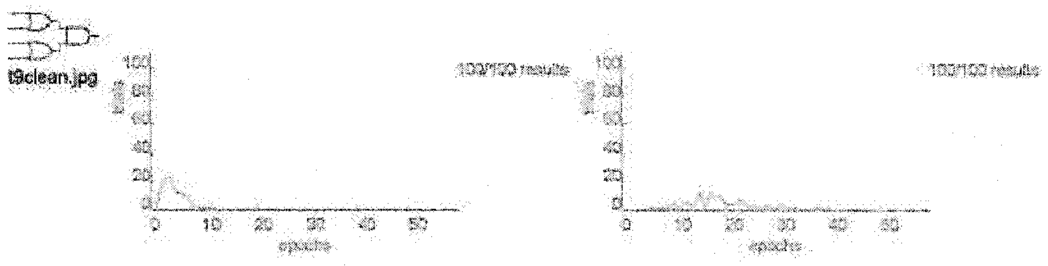


Figure 30. The distribution of the initialization is in the chart on the left, and the distribution of termination is in the chart on the right.

In Figure 30 we can see that the initialization algorithm was always successful at isolating the first symbol within ten epochs, most often before five. This is a very good result, because it shows that few mutations were needed to isolate a symbol. The termination results in Figure 30 produce a wider distribution with a maximum near fifteen epochs. The termination criteria was always satisfied before the epoch limit indicating mutation and crossover were always successful at finding the remaining symbols. Figure 28 shows that the duration of useful evolution follows a similar distribution to the termination, but has a maximum near ten epochs. This shows that most often the first and third symbols were found ten epochs apart.

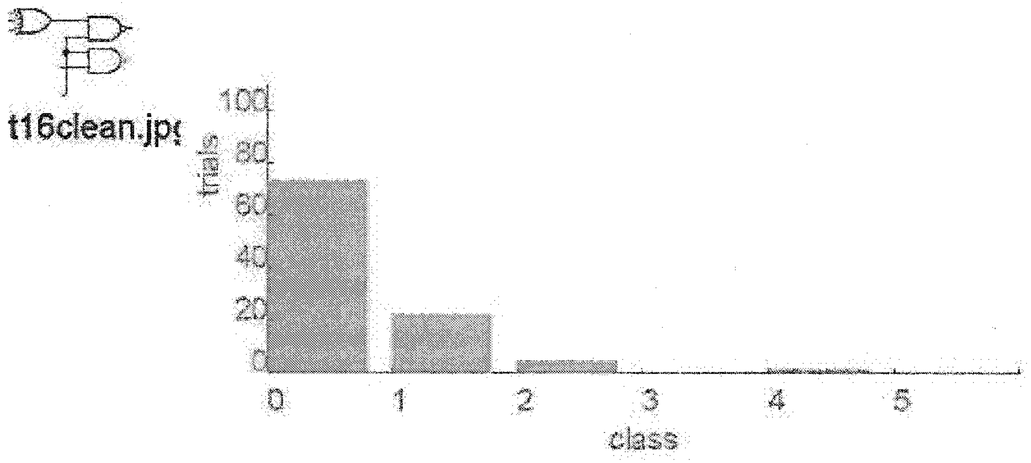


Figure 31. A circuit image that produced class 0 and class 1 results.

Another interesting circuit diagram is shown below in Figure 31. In this diagram, there were three symbols as well, and both class 0 (Figure 32(b)) and class 1 (Figure 32(c)) results were found.

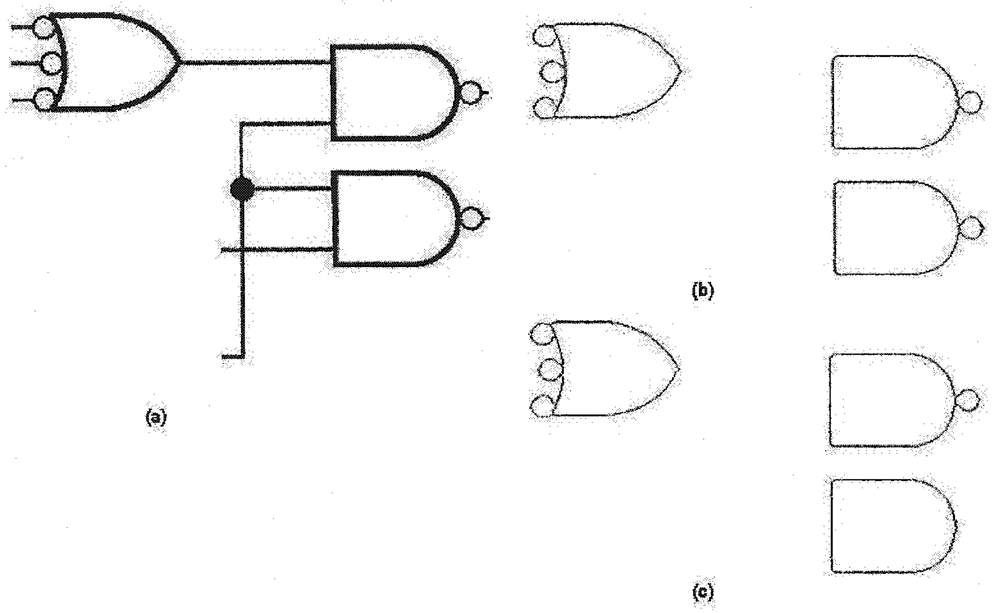


Figure 32. (a) An sample line diagram input. (b) A class 0 result from segmentation. (c) A class 1 result from segmentation.

Here we can see that the gates are more complicated, with up to three negation bubbles each. This increases the complexity of the gates, and makes segmentation harder, but it is still possible. The symbol verification is trained to recognized the OR gate in original form, as well as the form seen here, with three negation bubbles on the inputs. It was also trained to recognize both the NAND gates, as well as the base AND patterns. It will be interesting to examine the performance of the algorithm on this input, as shown in figure 26.

We can see in Figure 31 that the results were mostly confined to class 0 and class 1 results. There were also a few class 2 results when a single gate was not isolated, and even fewer class 4 results when two gates were missed. Since the majority of results are class 0 and class 1, the performance is still considered acceptable on this input. In Figure 26 (c) we can see the initialization performed well, with a narrow distribution contained before the ten epoch mark. The termination distribution shown in figure 26 (d) has a sharp spike after the 50 epoch limit. This indicates the termination criteria was never satisfied, and thus the algorithm always ran as long as was allowed. This is undesirable since a great deal of time is wasted searching for symbols when no more are present, and we can see in figure 25(a) that there is a large connected, non-symbol region on the left of the NAND gates. This region causes the termination criteria to fail.

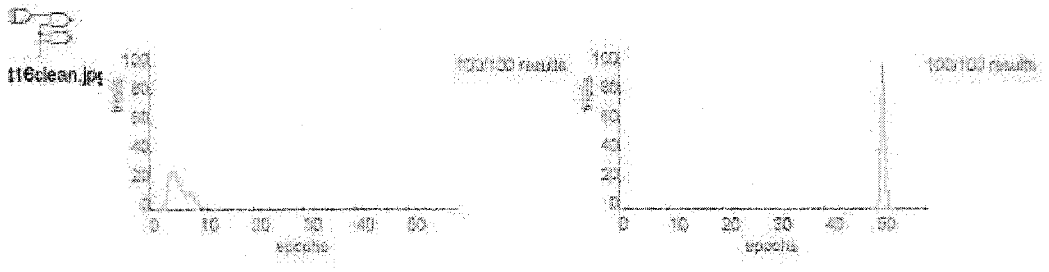


Figure 33. The distribution of the initialization is in the chart on the left, and the distribution of termination is in the chart on the right.

The final sample line diagram we will discuss in detail is one that contains five symbols, as shown in Figure 34. In all the previous discussions, the input samples contained three symbols, but the only reason for this is that they were interesting discussions.

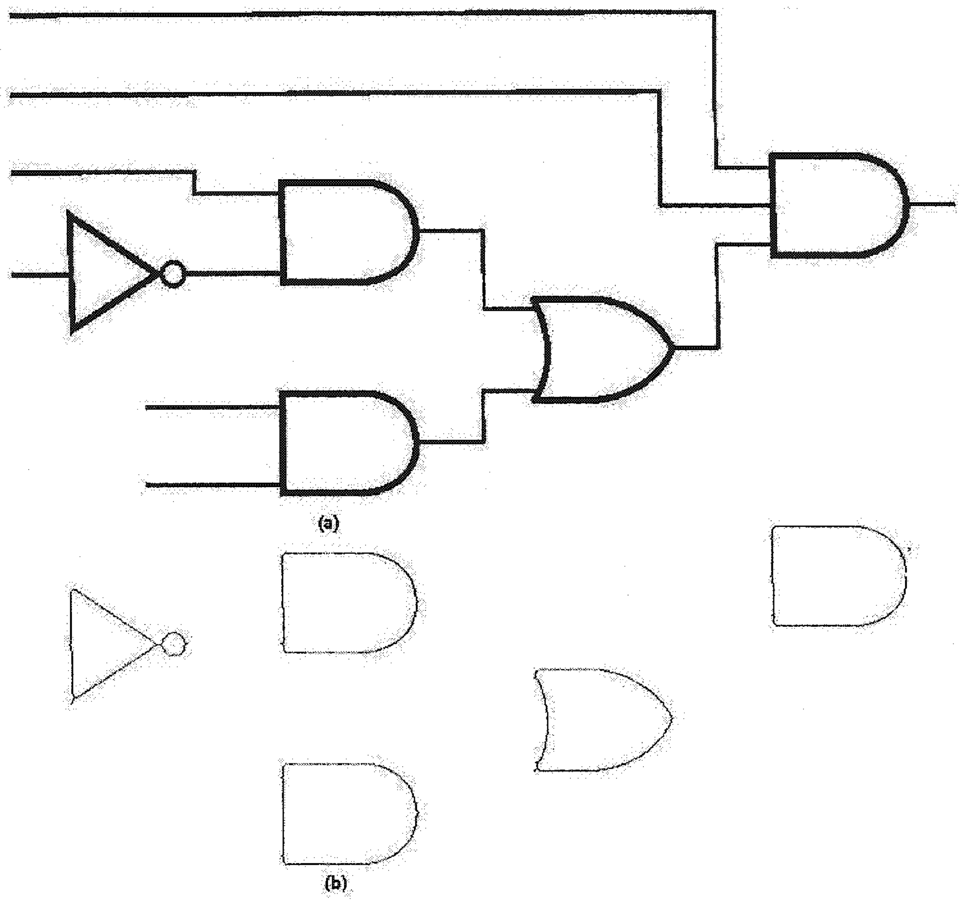


Figure 34. (a) An sample line diagram with 5 symbols. (b) A class 0 result from segmentation.

Between performing experiments on diagrams with three symbols, and diagrams with five symbols, there were no changes made to the proposed system. The initialization routine, population size of ten, and epoch limit of fifty, were all left constant. The results shown in Figure 35 show mostly class 2 results when one symbol was missed, some class 3 results when two or three symbols were missed, and few class 4 results when only one symbol was found. This does not indicate a fatal problem however, since some class 0 results were obtained. When we look closely at the initialization distribution shown in Figure 36 we can see that the distribution spreads as far as twenty epochs. This means that in some cases almost half of the time was spent looking for the first symbol, so it is not surprising that not all were found. Figure 36 shows that the termination criteria was never satisfied, even when class 0 results were obtained, because of the long non-symbol lines. The presence of some class 0 results in conjunction with the failure of termination leads the author to suggest that fifty epochs is not enough time to reliably find all symbols in this case, and that the results would be improved if the limit was increased to a higher number.

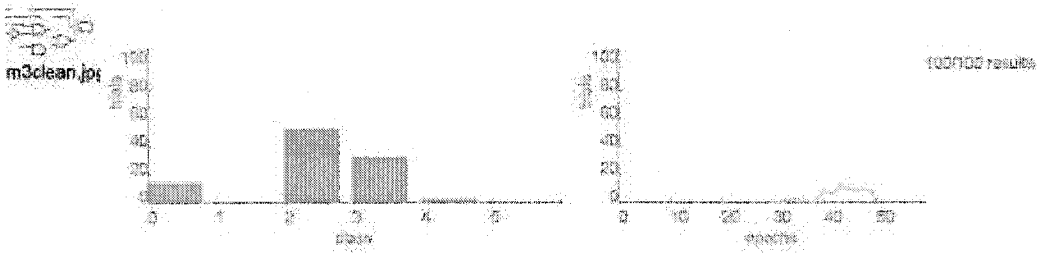


Figure 35. A few class 0 results, mostly class 2 results.

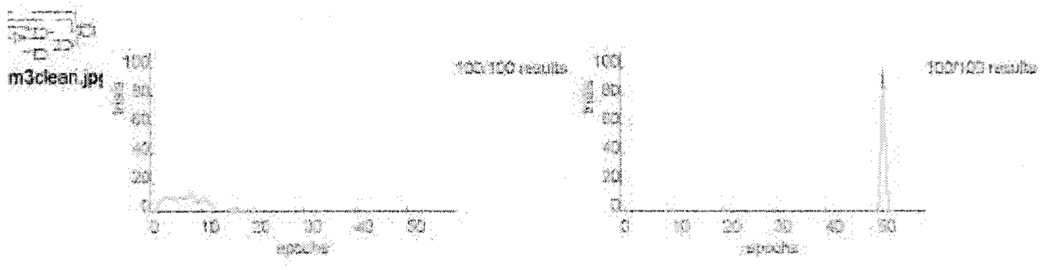


Figure 36. The initialization and termination distributions for a larger circuit image.

Now that we have discussed a few results from both touching character input, and line diagrams, we can discuss the results of all experiments as a whole. Due to the different nature of all the input images, with variations in number and types of symbols, the data shown in each performance analysis is summarized and plotted on the following charts. Only class 0 and class 1 results are considered for the later sections of this group analysis, since the goal is to show how the system performs when results are obtainable. The trials that are not considered produced class 2, class 3, class 4, and class 5 or no results and can be ignored until the specific faults (unrecognizable symbols, epoch limit too short, etc.) can be identified and corrected.

Analysis of All Inputs

First, we can look at the accuracy of segmentation over all images. The chart shown in Figure 37 are shrunk by a factor of 1000 for display purposes.

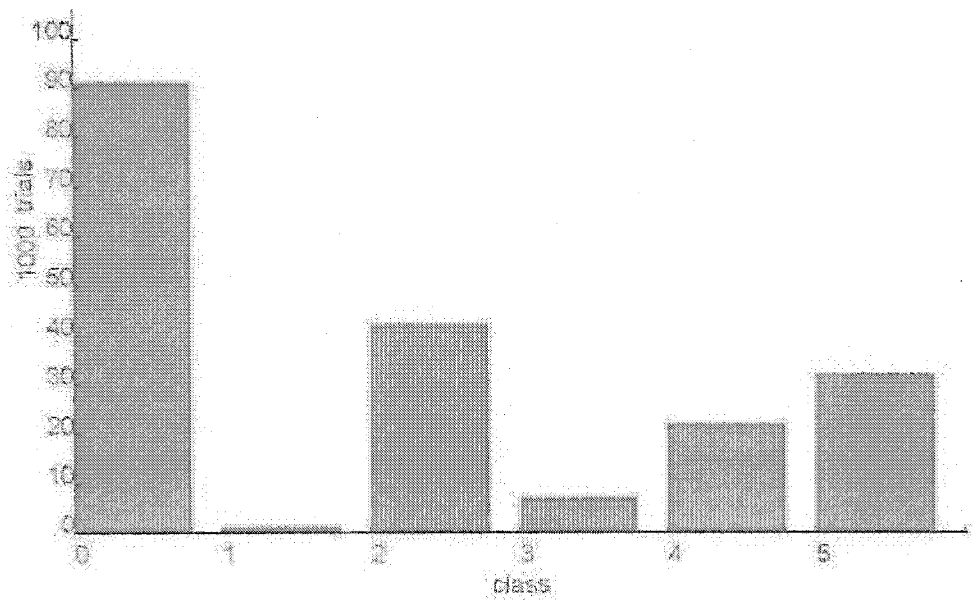


Figure 37. The total accuracy results for all experiments. The number of trials that attain a given accuracy are 1000 times more than shown here. Clearly Class 0 results are produced most often.

In the majority of cases, class 0 or perfect results are obtained. The majority of the failure cases were class 2 results, where a single symbol location was missed. The class 2 results could be reduced in number by improving the symbol verification system to identify more variants of symbols, or by allowing the algorithm to run for longer epochs. In a few cases, the termination criteria was always satisfied prematurely.

Next, we should examine the performance of the initialization on successful cases. We will only look at cases where class 0 and class 1 results were obtained in the following charts.

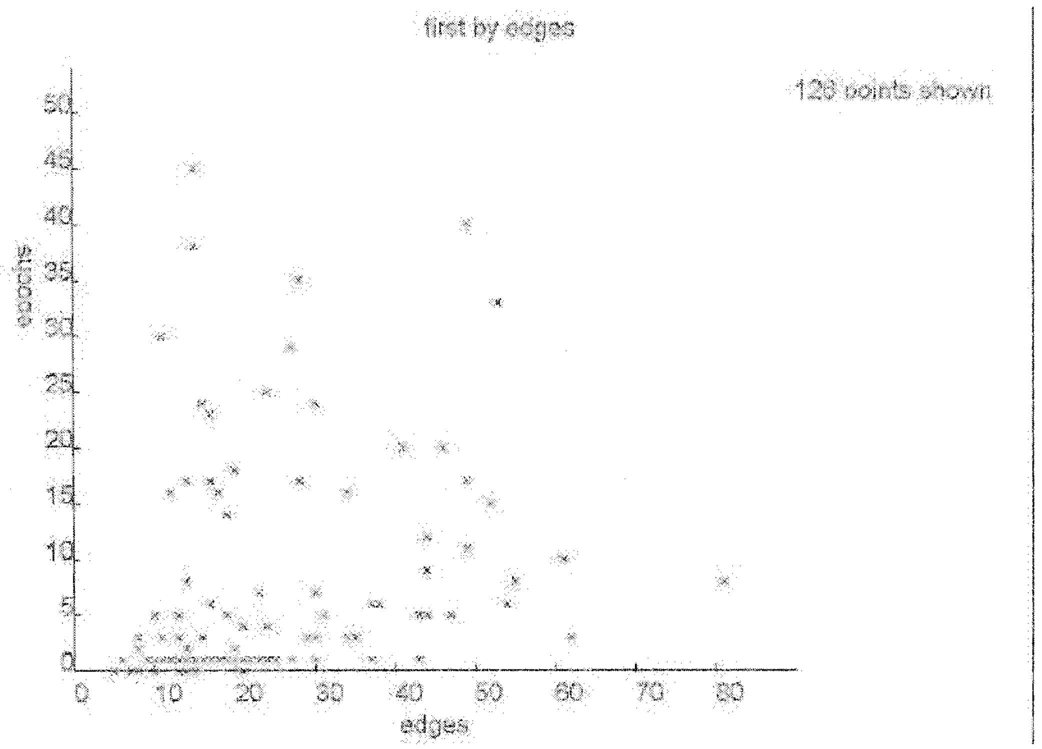


Figure 38. The initialization results for the entire experiment, over all data sets. There is a cardi nality of each point that is greater than or equal to 1, with a total of 126 points represented.

The chart in figure 38 shows the initialization performed well in nearly all cases. There few points that are very high, indicating that the first symbol was not found until after epoch 45 at the latest, which is near the 50 epoch limit for the experiment. This could cause results for a single input to appear poor, if the duration for that particular input was longer than five epochs. There are also a few points where on the 0 epoch line, showing that in some cases, mutation is not necessary to isolate a symbol. It is interesting to note that the cases where initialization took the longest time to find the first symbol occur when the graph is smaller, and has less edges. This indicates that the initialization is impacted less by graph size, and more by layout.

Next, it is important to examine the performance of the termination criteria.

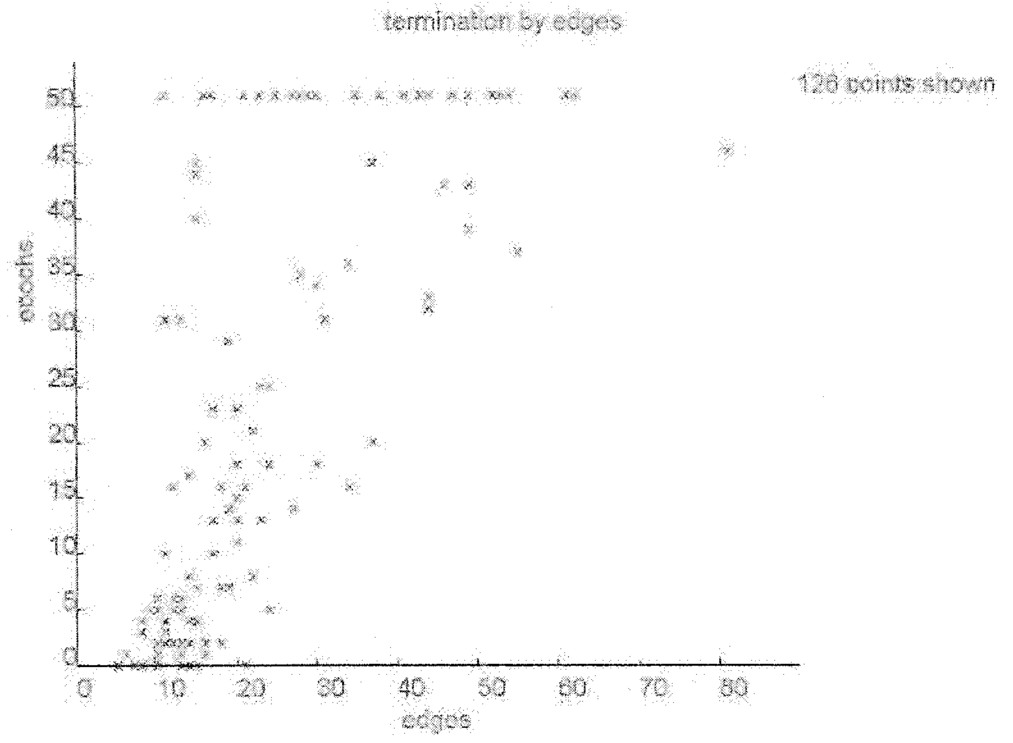


Figure 39. The termination results for the entire experiment, over all data sets. There is a cardinality of each point that is greater than or equal to 1, with a total of 126 points represented.

From the chart in Figure 39, it is clear that the characteristic points can be clustered into two clusters, separated by a horizontal line. The upper cluster, representing all inputs where the termination criteria was never satisfied, and evolution proceeded until the epoch limit was reached, is the row of points above 50 on the y-axis. These points generally correspond to inputs where class 0 or class 1 results were not obtained. In these cases, termination was not satisfied because there was a large, connected, non-symbol region in the input (this occurs frequently in circuit images). In these cases the algorithm continues to assume it can find another symbol in that region when none exists. The termination criteria requires more refinement. The remainder of the points shown on the chart indicate that the termination criteria performed well, stopping the algorithm at an appropriate time. This chart also shows that larger graphs will seldom terminate in only a few epochs, indicating that the epoch limit should be longer for large graphs.

An interesting result is that at least seven points in both figure 30 and figure 31 exist on the zero epoch line. These points represent the cases when the initialization algorithm isolated symbols to a point where the termination criteria was satisfied. In these cases, it was unnecessary to evolve the population since the optimal individual was found without mutation or crossover.

The combination of the initialization and termination results show that the proposed system is in fact feasible, and that it can solve the problem as described in the previous chapters. To analyze the performance of the system, and it's scalability from smaller to larger inputs, the duration of evolution must be examined. This analysis will compare the performance of the mutation and crossover operators, while eliminating the performance of the initialization algorithm. The duration used here represents the time, or work done between the epoch when the first symbol was found, and the epoch the algorithm terminated.

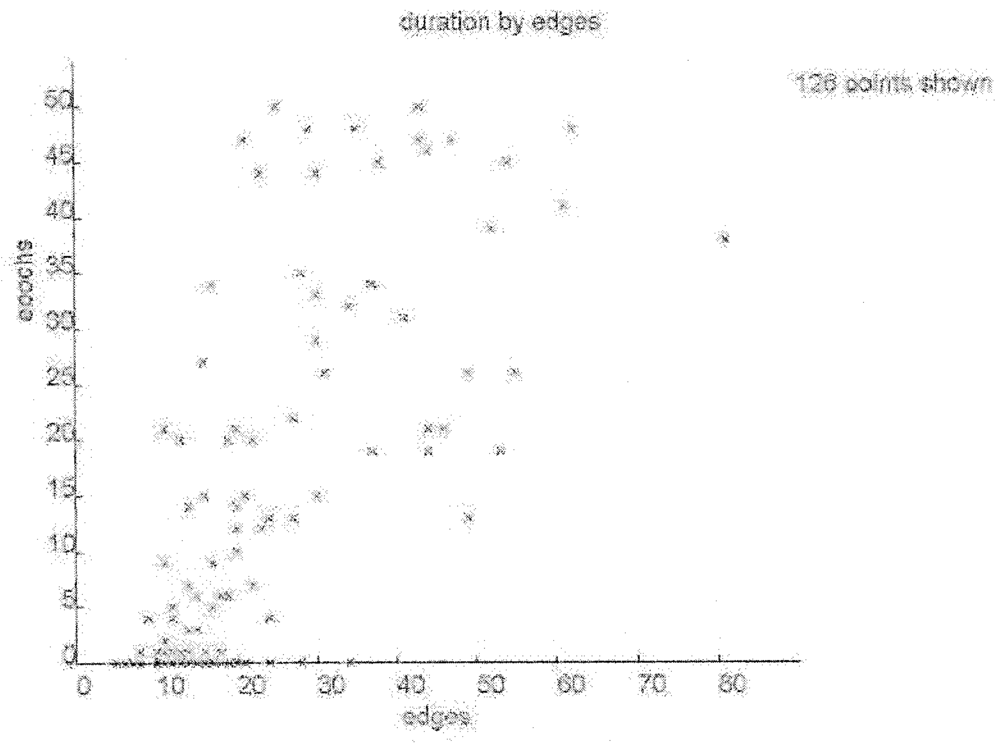


Figure 40. The duration results for the entire experiment, over all data sets. There is a cardinality of each point that is greater than or equal to 1, with a total of 100 points represented.

The points in Figure 40, like the termination chart in Figure 39, show that larger graphs take longer to evolve. There are a great number of points on the 0 and 1 epoch lines, showing a number of cases where isolation of the first symbol and the last symbol occurred within one epoch of each other. In these cases, it is likely different individuals isolated different symbols, and the information was shared with a crossover.

This is a good result when combined with the initialization result, that the first symbol is generally found within the first 10 epochs. Thus, if we can expect to find a symbol in 10 epochs, and we expect the duration to last at most 25 epochs, we can expect a good solution in at most 35 epochs of evolution.

5.2.2 Performance with Varied Parameters

5.2.2.1 Discussion of the Investigation

This investigation compares the performance of the algorithm on a small set of input images, with 64 different parameter configurations. There were two choices for initialization, either using the *initializeIndividual* algorithm in Appendix A, or a random initialization algorithm. There were two choices for the parent selection strategy, either the full parent selection described in Chapter 4, or a random selection of two parents from the set that had a symbol matched. Every combination of initialization and parent selection strategy was tested with population sizes of 5, 10, 15, and 20 individuals, and evolution was terminated after 50 epochs.

The goal of this investigation was to learn about how the various strategies and population sizes affect the evolution, as well as to see how the algorithm performs on a range of input sizes. The set of input images for this investigation was formed in a different manner from the investigation above. Each input image is a small section of the long character string shown in Figure 41, with lengths of 3, 5, 7, 9, and 11 characters. This was done to show the scalability of the algorithm, and to provide some escalation of the input size, while not changing the nature of the input in any significant way. To show how the algorithm performs differently on various sizes of input, data was collected as described in Section 5.2.1, and also time in seconds to perform the segmentation. All of the experiments were run on a computer that was only used for collecting data. This ensured that the real time evaluations were not skewed by the peak usage and idle time on the machine. The computer used was a laptop with an AMD Duron 900MHz processor, 256 MB of RAM, and a 10 GB hard disk. The operating system was Windows XP, and effort was made to ensure no automatic updating, virus scanning, or other maintenance was performed for the duration of the experiments.

The accuracy and duration of evolution (in epochs) data is contained in Appendix C1. The epoch of initialization, and epoch of termination data is contained in Appendix C2, and the new time in seconds data is contained in Appendix C3. Some specific results will be discussed below.

The largest two images, of 9 and 11 characters were selected for an additional related investigation. Segmentation was performed on both of these images, for all configurations, with an evolution terminated after 100 epochs, instead of 50. This investigation was intended to see how the low epoch limit affects the final results. The results for the additional investigation are contained in Appendix D1, Appendix D2, and Appendix D3.

5.2.2.2 Discussion of the Analysis

There was a large amount of data collected in this investigation, that is very difficult to draw conclusions about. The analysis will be performed by investigating a number of simple questions, and drawing conclusions from those questions. The specific investigations will be:

a) Which configuration provides the best results?

b) How does population size affect the final result?

c) How does the time increase as the input size increases?

d) How does the epoch limit affect the final result?

A change was made in the way the accuracy results were classified for this experiment. Since the inputs are strongly related, an absolute scale was chosen, with classes ranging from 0 to 11 to represent the number of symbols isolated in the final result. In this way, a failure is the same for all input sizes, but a perfect result is different for each one.

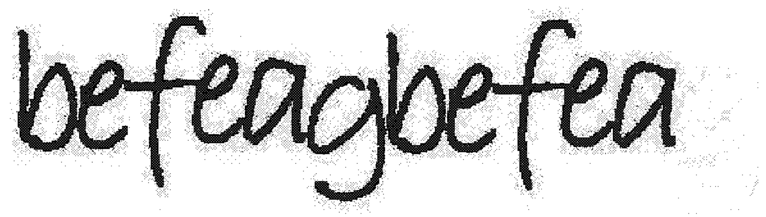


Figure 41. An 11 character input image. Smaller inputs were formed by taking small sections of this string. It does not matter than the string is not a word, only that the characters can be isolated and recognized.

5.2.2.3 Results on Varied Configurations.

To investigate the question posed above as point a) "Which configuration provides the best results?", we can look at the accuracy for every configuration. Figure 42 shows the accuracy for the 4 population sizes, with the seeded initialization, and full parent selection. Figure 43 is similar to Figure 42, but for the seeded initialization and random parent selection configuration. Figures 44 contains the results for the random

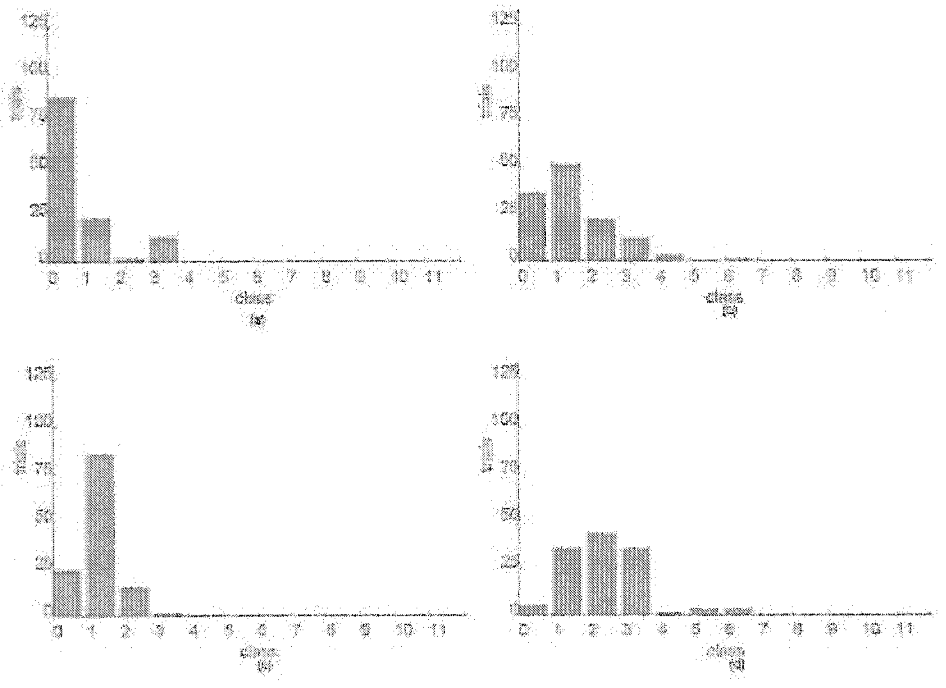


Figure 42. The accuracy results for all input images, for the seeded initialization and full parent selection configuration with a 50 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

initialization full parent selection configuration, and Figure 45 contains the results for the random initialization and random parent selection configuration. These four diagrams will be compared to determine which configuration proved to be the best.

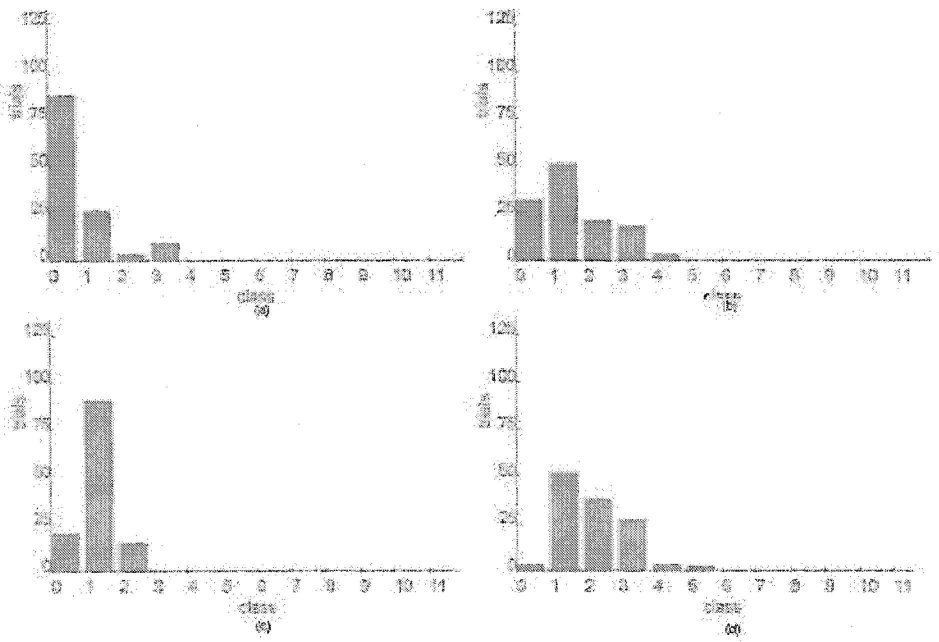


Figure 43. The accuracy results for all input images, for the seeded initialization and random parent selection configuration with a 50 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

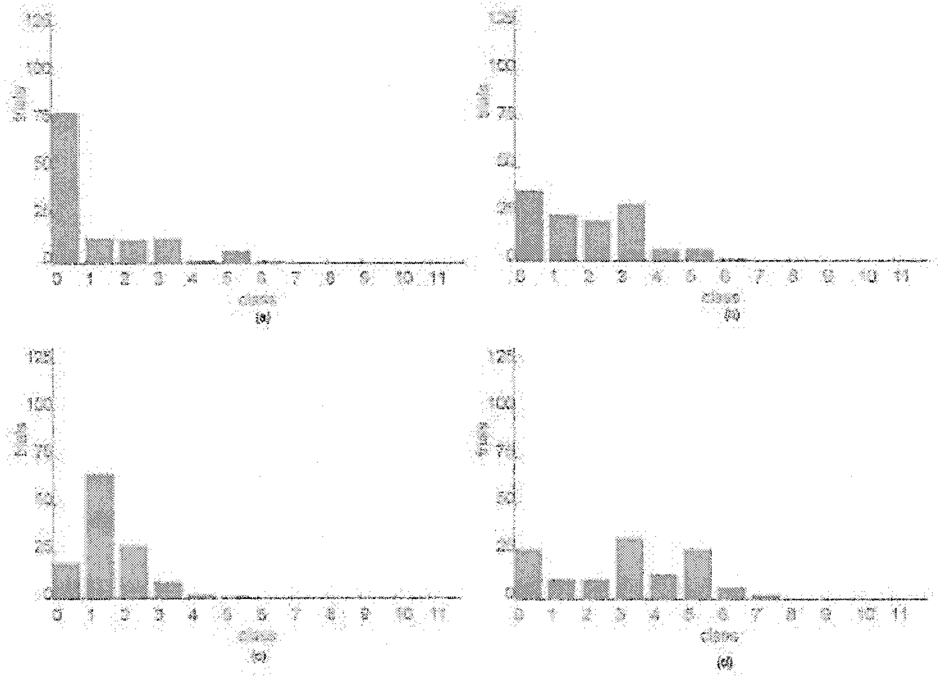


Figure 44. The accuracy results for all input images, for the random initialization and full parent selection configuration with a 50 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

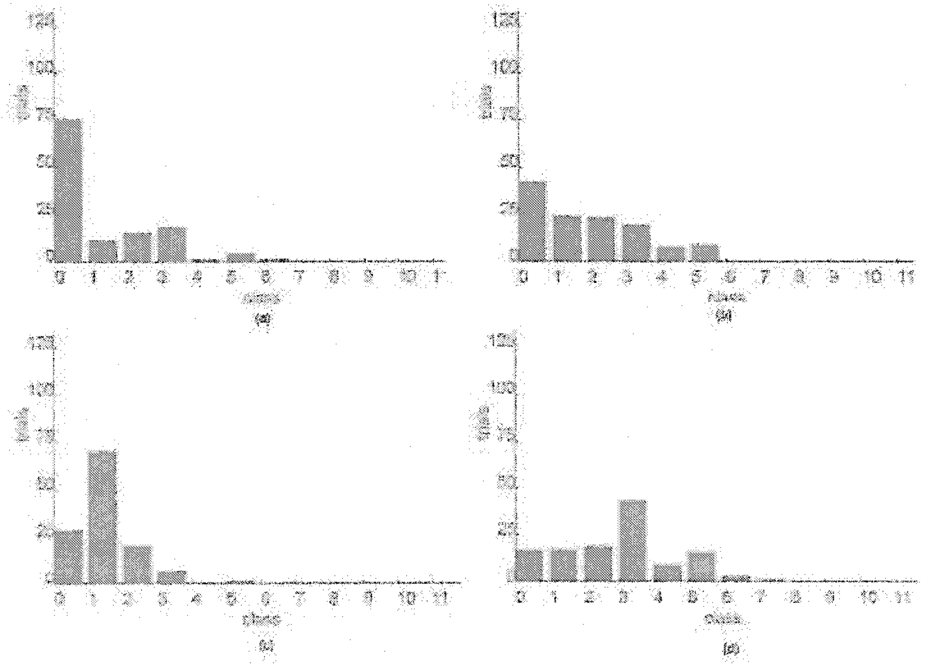


Figure 45. The accuracy results for all input images, for the random initialization and random parent selection configuration with a 50 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

To compare the results in Figures 42, 43, 44, and 45, one must compare Figure 42(a) to Figures 43(a), 44(a), and 45(a). The same must be done for Figures 42(b), 43(b), 44(b), and 45(b), and so on for (c) and (d). It is clear by this comparison that the charts in Figure 42 and Figure 43 are similar, as are the charts in Figure 44 and Figure 45. The similarities are between charts with identical initialization routines, which is an interesting result. Since charts that share initialization routines are more closely related than charts that share parent selection strategies, we can conclude that the initialization routine has a greater impact on the final result. All observations made here, are supported by the similar data presented in Figures 46, 47, 48, and 49 that represent the same experiments with a 100 epoch time limit.

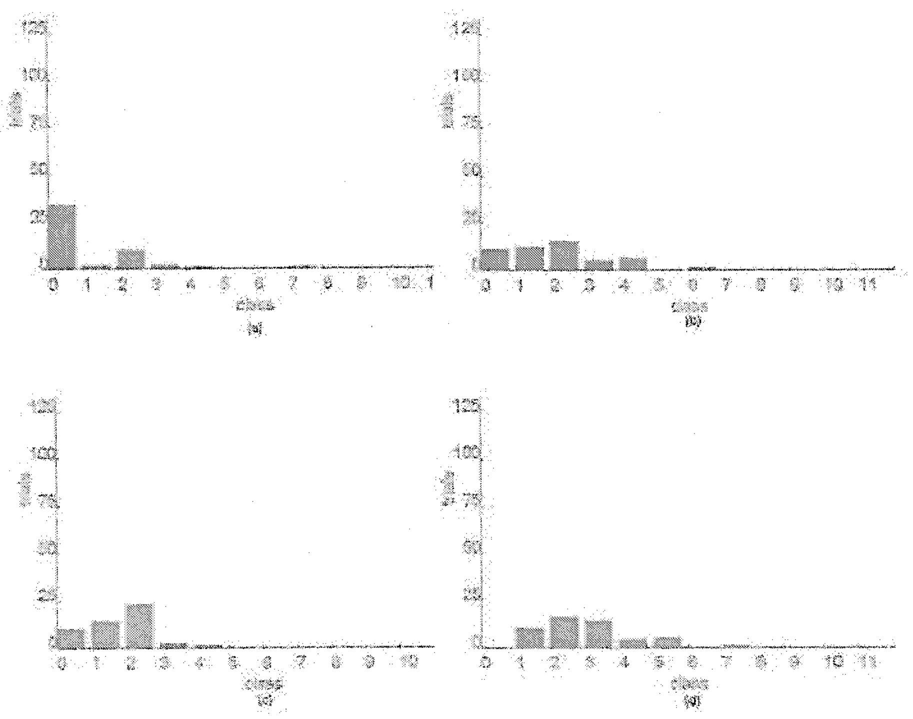


Figure 46. The accuracy results for the 9 and 11 character inputs, for the seeded initialization and full parent selection configuration with a 100 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

One further, unexpected conclusion is that for the experiments performed, the random initialization produced better results, since there are more, higher class accuracy results shown in Figures 44 and 45, than there are in Figures 42 and 43. Figure 44 presents slightly better results than Figure 45 in all charts (a), (b), (c) and (d), so the conclusion is that the random initialization with a full parent selection configuration performed the best.

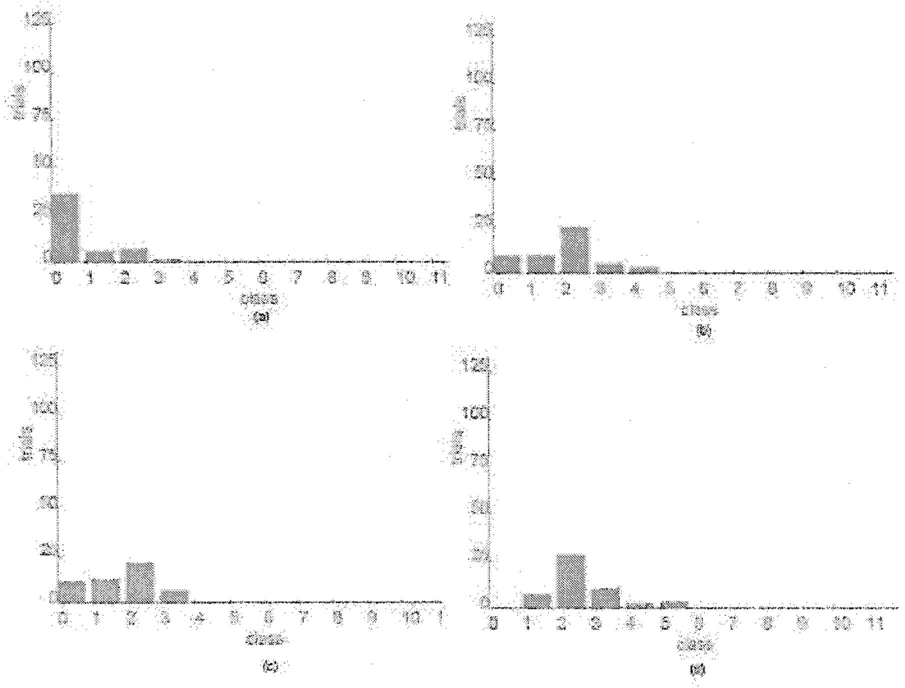


Figure 47. The accuracy results for the 9 and 11 character inputs, for the seeded initialization and random parent selection configuration with a 100 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

To investigate question (b) from Section 5.2.2.2, "How does population size affect the final result?" we can use Figures 42, 43, 44, and 45 in a different manner than above. To investigate this, we must compare Figure 42(a) to Figure 42(b), 42(c) and 42(d). The

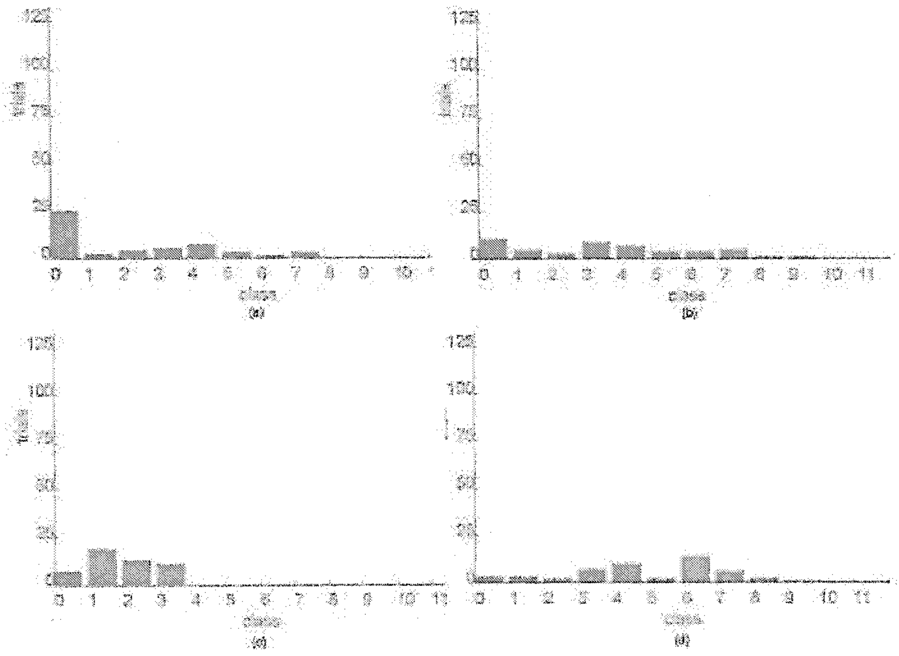


Figure 48. The accuracy results for the 9 and 11 character inputs, for the random initialization and full parent selection configuration with a 100 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

same goes for Figures 43, 44, and 45. All Figures 42, 43, 44, and 45 show the same trend with respect to population size. In all Figures, the most class 0 results are produced in the smallest population size of 5 individuals. Also, in all Figures, the most, higher class results appear in chart (d), with the largest population size of 20 individuals. A very interesting trend in all Figures is that chart (b), presenting results from a population size of 10, shows better performance than chart (c) that presents results from a population size of 15. The conclusion of this investigation then, is that the largest population size of 20 produced the best results. The second best results were produced by a population size of 10, followed by population sizes of 15 and 5 respectively.

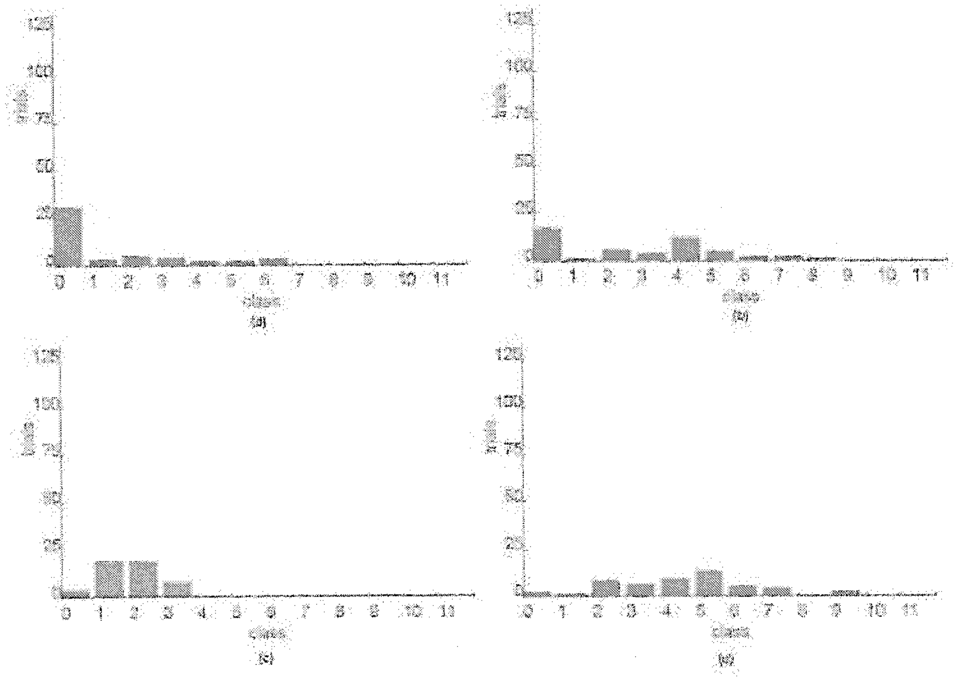


Figure 49. The accuracy results for the 9 and 11 character inputs, for the random initialization and random parent selection configuration with a 100 epoch evolution limit. (a) Population size of 5. (b) Population size of 10. (c) Population size of 15. (d) Population size of 20.

To investigate the third question above, (c) " How does the time increase as the input size increases?" we have to look at the results presented in Appendix C3 Figures 160 to 239, as well as Appendix D3 Figures 64 to 95. These figures represent the time in seconds that the algorithm was run for. In Appendix C3 evolution was terminated after 50 epochs, and in Appendix D3 evolution was terminated after 100 epochs. There are a great number of charts to compare in these two Appendices, but a few trends stand out when they are examined closely.

The first trend that is observable is that the larger inputs take longer to evolve. There is no solid relationship between the change in input size (measured in characters), and the change in evolution time, although it appears to be polynomial or exponential in nature.

There is a greater increase in time from 9 characters to 11, than there is from 3 characters to 5. This is expected however, since the algorithms for graphing and mutation would take longer for larger inputs. As the input size increases, the number of possible parts in the partition increase exponentially for each edge that is added to the graph. This implies that there would be an exponential increase in the number of symbol verification tests. The increase in time however is not as steep as the exponential increase in the number of possible parts, indicating the algorithm compensates by using directed evolution.

The second trend is that the distributions tend to grow wider as the population size increases. In the majority of all cases, evolution was terminated at the epoch limit of either 50 or 100 epochs, producing relatively narrow peaks in these charts. The same trend is observed in the accuracy data, that the larger inputs produce wider distributions of results. The correlation between these two charts suggests that the time required for evolution and the accuracy are related, which would be expected. When the algorithm can quickly find a good solution, or make a good start evolution will proceed faster. When poor results are obtained, the algorithm has spent a good deal of time searching through possible, but not correct solutions.

A third trend observed from comparing results from Appendix C3 and Appendix D3 is that when the epoch limit is doubled, the evolution time is roughly doubled. In most cases, the distributions appear similar but a bit wider in Appendix D3. This suggests that the time to complete one epoch of evolution is constant for a given input.

The fourth and final investigation was (d) "How does the epoch limit affect the final result?", and this can be done by comparing Figure 42 to Figure 46, Figure 43 to Figure 47, Figure 44 to Figure 48, and Figure 45 to Figure 49. This comparison is a bit skewed however, since Figures 42, 43, 44, and 45 contain results for 5 input images and Figures 46, 47, 48, and 49 only contain results for 2 input images. Despite the difference in the number of results shown, the trend is still quite obvious.

The data collected with a 100 epoch limit shows that many more symbols were isolated when the time was increased. Figures 48 and 49 show that as many as 9 symbols were isolated in a few trials, where the maximum number of symbols found in any trial with the 50 epoch limit was 7. This trend suggests that more the results would appear to be better for all of the experiments if the epoch limit was increased from the start, but this offers a serious trade-off. As was observed earlier, the time to complete one epoch of evolution is roughly constant, suggesting that doubling the allowed number of epochs would double the experimental time. The choice to keep a lower epoch limit allowed many more trials to be run on limited computing resources.

5.3 Final Conclusions

There are many different approaches to testing a new algorithm in any particular domain, with the most popular being to compare results of other related research. This was not possible however, since there was no other research encountered that attempted to perform segmentation in the general case. Instead, the results of almost 250 000 trials on over 200 different inputs are presented, and a series of observations are made. There were a number of different investigations proposed, which attempt to test all of the features of the algorithm.

In the first investigation, the goal was to see that the algorithm performed well in the general case. To show this 198 different touching character, and simple circuit diagram images were selected, with 100 trials performed on each. The configuration of the algorithm remained constant throughout all trials, and good results were obtained. A few cases were identified that caused insurmountable obstacles with this segmentation method, such as broken symbols, or when a symbol can not be defined as a partition of the graph.

It was also discussed that it is possible to produce more than one partition of the graph where symbols are isolated. This can cause problems when one valid symbol is a part of another valid symbol. It would be necessary to use high level reasoning to determine which is the appropriate segmentation.

In the second major investigation, the performance of the algorithm was examined on a smaller set of input images, when the configurations of the algorithm were allowed to change. There were a total of 16 configurations tested that varied in initialization strategy, parent selection strategy, and population size. It was discovered that random initialization with the full parent selection strategy produced the best results in these experiments. This was a surprising result, since other algorithms based on evolution find better results with a seeded population.

There was also an additional piece of data collected for this investigation that was the evolution time in seconds. This data, although skewed by the computer that was used to perform experiments, lends some insight into the scalability of the algorithm. Based on this data, it was concluded that the time required for segmentation increases in a non-linear fashion as the number of input symbols increases. It is very difficult to determine a relationship between time, problem size, and the number of symbols, because when a new symbol is added, the number of edges added to the graph is not constant. The specific symbol, and the way in which it is connected both influence the number of edges in the new graph.

Another set of data was collected by running the algorithm, in all configurations, on small subset of the data used for scalability testing. For this additional experiment, the epoch limit was doubled, and it was observed that the time required for evolution roughly doubled as well. From this it was concluded that the time for a single epoch is constant. It was also observed that the longer running trials found more symbols per trial than the shorter ones. This was not surprising, but represents a trade-off when performing experiments, that either the number of trials, or the maximum number of epochs must be limited so that data can be collected in a limited time.

This algorithm was found to be good in the general case, and could perform segmentation with no knowledge of what symbols it expected to find. The only domain specific component is accessed through a strict interface, and may be replaced for new domains, or as new technology permits. The algorithm does not scale well for larger inputs however, since the increase in running time is not linear to the increase in input size. This is a limiting factor of the usefulness, but does not indicate the algorithm is useless. There are many domains of segmentation that contain small or limited size inputs where this would be useful. Two examples include handwritten postal code processing, and automatic recognition of hand drawn engineering assignments for junior University students. It should be noted however that, at he time the experiments were performed the computer used for timing was slow by industry standards. There are computers available with nearly two times the processing speed, and much faster RAM. The hardware used has a great effect on the time required for running the algorithm.

Chapter 6: Comments and Future Work.

6.1 Comments

The proposed system has been shown to be feasible as an approach to a general problem. It successfully isolates symbols in a number of symbol domains, provided the domain can be defined as "touching or connected symbols in a line diagram". This definition includes touching characters, logic circuit diagrams, and many other types of line diagrams. The remainder of this section is a list of remarks that apply only to a specific area of the proposed system, and serve to break down the types of problems the system encountered.

The following remarks were observed from the experimental results presented in this chapter, and Appendix B, Appendix C, and Appendix D..

Initialization

1. The *initializeIndividual* algorithm prefers left to right arrangements of symbols, and thus performs very well for touching characters, but tends to repeatedly miss symbols in numerous trials. Random initialization was more sucessful according to the data in Appendix B and Appendix C.
2. In some cases, the initialization found character symbols, and occasionally it found the optimal partition. (Appendix B, Figure 211, Figure 224...).

Termination

3. The termination criteria worked well for characters, but occasionally caused premature termination (Appendix B Figure 7, Figure 20...).
4. The termination criteria worked well for logic diagrams, but encountered difficulty when there was a large, connected, non-symbol region present. Results were

produced with perfect accuracy that always terminated at the epoch limit (Appendix B, Figure 179 and Figure 381, ...)

Evolution

5. In most cases, evolution is successful and productive.
6. The crossover operator is successful at accelerating evolution by collecting different symbols into a single individual.
7. The time required to perform segmentation increases with the input size, as the graphing and searching algoirthms require more time.

Overall

8. The success of the segmentation is controlled by the accuracy of the symbol verification. Failure to recognize a symbol will make it impossible to produce class 0 results.
9. Class 0 and class 1 results are indistinguishable without high level knowledge, both are considered optimal.
10. The algorithm is successful at producing both class 0 and class 1 results. This is a valuable feature since a high level process can choose which is most appropriate.

6.2 Contributions to Current Research

The system presented here makes a number of contributions to the area of document analysis and recognition. The most significant are outlined below:

1. The problems of segmenting touching characters, and segmenting line diagrams have been unified into a single problem of segmenting connected symbols. This allows for a more general approach, and more versatile document recognition system. No previous work was encountered that approached both problem domains.
2. The domain of segmentation is now controlled only by a symbol verification system. This allows for previous, highly specialized recognition work of many researchers to be incorporated into one common system. Previous research had a great dependency

between the way characters were to be recognized, and how they were isolated, leading to monolithic or domain specific techniques.

6.3 Suggestions for Future Work

It is common for a project of this scope to be left with many areas for improvement. Below is a list of the author's suggestions to enhance, accelerate, and improve segmentations using the proposed system.

1. The evolutionary algorithm used employs many modular components, each of which may be enhanced for the general segmentation problem, or possibly specialized for any domain.
2. The current initialization algorithm attempts to separate two symbols arranged east to west on the image. This could be modified in any number of ways.
3. Mutations could intelligently move edges from one part to another.
4. Parallel implementation would increase the speed.
5. Symbol verification could be improved for handwritten script and diagrams.
6. Information on what symbols have been isolated could be used to help form symbols of a common domain, or high level information could be used to direct mutations to form special shapes or common features in symbols.
7. The interface between symbol verification and the evolutionary algorithm can be better defined, so that the slow operations of drawing and tracing the line images are eliminated.

Literature References

1. A. Globus, E. Langhirt, M. Livny, R. Ramamurthy, M. Solomon, and S. Traugott, "JavaGenes and Condor: Cycle-Scavenging Genetic Algorithms", *Technical Report NAS 00-006,* NASA Ames Research Centre, 2000.
2. A.P. Maubant, Y. Autret, G. Lé onhard, G. Ouvradou, A. Thépaut, "An Efficient Handwritten Digit Recognition Method on a Flexible Parallel Architecture", *Proceedings of MicroNeuro '96, 5th International Conference on Microelectronics for Neural Networks and Fuzzy* Systems, IEEE Computer Society, pp. 355-362, 1996.
3. A.Y. Commike, and J.J. Hull, "Syntactic Pattern Classification by Branch and Bound Search", Department of Computer Science, State University of New York at Buffalo, Buffalo NY, 14260
4. B. Al-Badr, R.M. Haralick, "Segmentation-Free Word Recognition with Application to Arabic", *Third International Conference on Document Analysis and Recognition,* IEEE Press pp. 355-359, 1995.
5. B.A. Julstrum, "A Scalable Genetic Algorithm for the Rectilinear Steiner Problem", *Proceedings of the 2002 Congress on Evolutionary Computation,* IEEE Press, pp. 1169-1173, 2002.
6. B.A. Julstrum, "Encoding Rectilinear Steiner Trees as Lists of Edges", *Proceedigns of the 16th Annual Symposium on Applied Computing,* IEEE Press, pp. 356-360, 2001.
7. C. C. Palmer, A. Kershenbaum "Representing Trees in Genetic Algorithms", Handbook of Evolutionary Computation, Institute of Physics Publishing (IOP Publishing), Dirac House, Temple Back, Bristol, UK. Edited by. T. Back, D. Fogel, and Z. Michalewicz, 2000.
8. C. Dunn, P. Wang, "Character Segmentation Techniques for Handwritten Text - A Survey", *Proceedings of the 11th IAPR Int. Conf. on Pattern Recognition*, Vol. 2, IEEE Press, pp. 577-580, 1992.
9. C.C. Palmer, A. Kershenbaum, "Representing Trees in Genetic Algorithms", *Proceedings of the first IEEE Conference on Evolutionary Computation*, Vol. 1, pp. 379-384, 1994.
10. D. Blostein, "General Diagram-Recognition Methodologies" in Graphics Recognition - Methods and Applications, Edited by. R. Kasturi and K. Tombre, LNCS Vol. 1072, Springer-Verlag, pp. 106-122, 1996.
11. D. Blostein, E. Lank, and R. Zanibbi, "Treatment of Diagrams in Document Image Analysis", *Diagrams*, pp. 330-344, 2000.
12. D. Blostein, H. Fany, and A. Grbavec, "Practical Use of Graph Rewriting", *Technical Report No. 95-373,* Department of Computing and Information Science, Queen's University, Kingston, ON, 1995.
13. D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing, 1989.
14. D. Kazakov, S. Mandhar, "A Hybrid Approach to Word Segmentation", *Proceedings of the 8th International Conference on Inductive Logic Programming*, Vol. 1446, pp. 125-134, 1998.
15. D. Motawa, A. Amin, and R. Sabourin, "Segmentation of Arabic Cursive Script", *ICDAR'97 (Fourth International Conference on Document Analysis and Recognition, Ulm German, August 18-20, 1997)*, pp. 625-628, 1997 .
16. D.S. Lee, C.R. Nohl, and H.S. Baird, "Language Identification in Complex, Unoriented, and Degraded Document Images", *Proceedings IAPR'96 Workshop on Document Analysis Systems,* 1996.
17. E. Lecolinet, "Cursive Script Recognition by Backward Matching", *Advances in Handwriting and Drawing: A Multidisciplinary Approach,* pp. 117-135, 1994.
18. F. Glover, M. Laguna, "Tabu Search", *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Sceintific Publising, Oxford, England. Edited by C. Reeves, 1993.
19. F. Kimura, Y. Miyake, and M. Shridhar, "Handwritten ZIP Code Recognition Using Lexicon Free Word Recognition Algorithm", *Proceedings of 3rd International Conference on Document Analysis and Recognition (ICDAR'95),* IEEE Press pp. 906-910, 1995.
20. G. Congedo, G. Dimauro, S. Impedovo, and G. Pirlo, "Segmentation of Numeric Strings", *Third International Conference on Document Analysis and Recognition Vol III.,* IEEE Press pp. 1038-1041, 1995.
21. G. Dzuba, A. Filatov, A. Volgunin, "Handwritten Zip Code Recognition", *Proceedings of the 4th International Conference on Document Analysis and Recognition*, pp. 766-770, 1997.

22. G. von Laszewski, "Intelligent Structural Operators for the k-way Graph Partitioning Problem", *4th International Conference on Genetic Algorithms (ICGA '91)*, Plenum, pp.45-52, 1991.
23. H. Maini, K. Mehrotra, C. Mohan, and S. Ranka, "Genetic Algorithms for Graph Partitioning and Incremental Graph Partitioning", *Supercomputing*, pp. 449-457, 1994.
24. H. Winkler, "Symbol Recognition in Handwritten Mathematical Formulas[sic]", *Int. Workshop on Mondern Modes of Man-Machine-Communication*, Maribour, Slovenia, pp.7/1-7/10, 1994.
25. I.S.I. Abuhaiba, S.A. Mahmoud, and R.J. Green, "Recognition of Handwritten Cursive Arabic Characters", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 6, pp. 664-672, 1994.
26. J. J. Hull, "A Database of Handwritten Text Recognition Research", *IEEE Transactiosn on PAMI*, Vol. 16, No. 5, pp. 550-554 May 1994.
27. J.F. Hebert, M.Parizeau, and N. Ghazzali, "Learning to Segment Cursive Words using Isolated Characters", *Proc. of the Vision Interface Conference*, pp. 33-40, 1999.
28. J.H. Shamilian, H.S. Baird, and T.L. Wood, "A Retargetable Table Reader", *4th International Conference on Document Analysis and Recognition (ICDAR '97)*, pp. 158-163, 1997.
29. K. Kise, O. Yanagida, and S. Takamatsu, "Page Segmentation Based on Thinning of Background", *IEEE Proceedigns of ICPR '96*, pp. 788-792, 1996.
30. M. Ahmed and R. Ward, "A Rotation Invariant Rule-Based Thinning Algorithm for Character Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 12, 2002.
31. M. Ahmed and R.Ward, "An Expert System for General Symbol Recognition", *Pattern Recognition 33(12)*, pp 1975-1998, 2000.
32. M. Ali Ozdil-Fatos, T. Yarman Cural, "Optical Character Recognition Without Segmentation", *4th International Conference on Document Analysis and Recognition (ICDAR '97)*, pp. 483-488, 1997.
33. M. Blumenstein, B. Verma, "A New Segmentation Algorithm for Handwritten Word Recognition", *IEEE International Joint Conference on Neural Networks*, Washington D.C. pp. 878-882, 1999.
34. M. Blumenstein, B. Verma, "Conventional V. Neuro-Conventional Segmentaiton Techniques for Handwriting Recongition: A Comaprison", *Proceedings of the Second IEEE International Conference on Intelligent Processing Systems (IEEE ICIPS'98)*, pp. 473-477, 1998.
35. M. Burge, G. Monangan, "Grouping Line Drawing Elements based upon their Area Voronoi Tessellation (Extracting Words and Multi-part Symbols in Graphics Rich Documents)" *Technical Report 131-95*, 1995.
36. M. Cheriet, R. Thibault, and R. Sabourin, "A Multi-Resolution Based Approach for Handwriting Segmentation in Gray-Scale Images", IEEE pp 159-163, 1994.
37. M. Dorigo, G. DiCaro, "Ant Colony Optimization: (A) New Meta -Heuristic", *Proceedings of the Congress on Evolutionary Computation*, IEEE Press, Vol. 2, Month 6-9, pp. 1470-1477, 1999.
38. M. Garris, "Component-Based Handprint Segmentation Using Adaptive Writing Style Model", *SPIE Web Document Recognition IV*, Vol. 3027, pp. 19-30, 1997.
39. M. Mirmehdi, P.L. Palmer, and J. Kittler, "Robust Line Segment Extraction Using Genetic Algorithms", *6th International Conference on Image Processing and its Applications*, IEE Publications, pp. 141-145, 1997.
40. M. Parizeau, R. Plamondon, "A Handwriting Model for Syntactic Recognition of Cursive Script", *Proc. of the 11th International Conference on Pattern Recognition (ICPR)*, Vol. 2, pp 308-312, 1992.
41. M. Shridhar, F. Kumura "Handwritten Address Interpretation Using Word Recognition With and Without Lexicon",*Proceedings of 3rd International Conference on Document Analysis and Recognition (ICDAR'95)*, IEEE Press, pp. 2341-2346, 1995.
42. M. Shrihdar, G. Houle, and F. Kimura, "Handwritten Word Recognition Using Lexicon Free and Lexicon Directed Word Recognition Algrithms", *4th International Conference on Document Analysis and Recognition (ICDAR'97)*, IEEE Press, pp. 861-865, 1997.
43. M. Yoshimura, T. Shimizu, and I. Yosimura, "A Zip Code Recognition System Using the Localized Arc Pattern Method", *Proceedings of the 2nd International Conference on Document Analysis and Recognition*, pp. 183-186, 1993.
44. N. Sitkoff, M. Wazlowski, A. Smith, and H. Silverman, "Implementing a Genetic Algorithm on a Parallel Custom Computing Machine", *IEEE Symposium on FPGA's for Custom Computing Machines (FCCM '95)*, pp. 180-185, 1995.

45. N.W. Strathy, and C.Y. Suen., "A New System for Reading Handwritten Zip Codes", *Proc. 3rd International Conference on Document Analysis and Recognition,* pp. 74-77, 1995.
46. R. Casey, E. Lecolinet, "A Survey of Methods and Strategies for Character Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7), pp. 690-695, 1996.
47. R. P. Futrelle, and N. Nikolakis, " Efficient Analysis of Complex Diagrams using Constraint-Based Parsing", *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR'95),* pp. 782-787, 1995.
48. R.W. Smith, J.F. McNamara, and D.S. Bradburn, "Pattern Classification Techniques Applied to Character Segmentation", *Proceedings of the 5th USPS Advanced Technology Conference,* 1998.
49. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecci, "Optimization by Simulated Annealing", *Science*, No. 4598, Vol. 220, pp. 671-680, 1983.
50. S. Lee, J. Chien-Jan Pan, "Unconstrained Handwritten Numeral Recognition Based on Radial Basis Competitive and Cooperatirve Networks with Spatio Temporal Representation", *IEEE Transactions on Neural Networks,* Vol. 7, No.2, pp. 455-474, 1996.
51. S. Lehmberg, H. Winkler, M. Lang, "A Soft-Decision Approach for Symbol Segmentation Within Handwritten Mathematical Expressions", *1996 Int. Conference on Acoustics, Speech, and Signal Processing (ISCASSP-96)*, Atlanta, USA, Vol. 6, pp. 3434-3437, May 1996.
52. S. Tsuruoka, N. Watanabe, N. Minamide, F. Kimura, Y. Miyake, and M. Shridhar, "Base Line Correction for Handwritten Work Recognition", *3rd International Conference on Document Analysis and Recognition (ICDAR'95),* IEEE Press, pp. 902-905, 1995.
53. S.W. Lee, D.J. Lee, and H.S. Park, "A New Methodology for Gray-Scale Character Segmentation and Recognition", *IEEE Transactions on PAMI*, pp. 1045-1050, 1996.
54. T. Bruel, "Segmentation of Handprinted Letter Strings Using a Dynamic Programming Algorithm", *Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR'01)*, pp. 821-827, 2001.
55. T. Floyd, Digital Fundamentals 6th Edition, Prentice-Hal Inc., Upper Saddle River, New Jersey, USA, 1997.
56. T. Goehring, Y. Saad, "Heuristic Algorithms for Automatic Graph Partitioning", *University of Minnesota Technical Report UMSI 94-24,* 1994.
57. T.M. Breul, "Representations and Metrics for Off-Line Handwriting Segmentation", *8th International Workshop on Frontiers in Handwriting Recognition,* 2002.
58. W.G. Kropatsch, and M. Burge, "Minimizing the Topological Structure of Line Images", *Proceedings of the 7th International Conference on Statistical and Structural Pattern Recognition*, Springer-Verlag, 1998.
59. Y Yu, A Samal, and S Seth "Isolating Symbols from Connection Lines in a Class of Engineering Drawings", *Pattern Recognition*, Vol. 27, No. 3. pp 391-404, 1994.
60. Y. Yu, A. Samal, and S.C. Seth "A System for Recognizing a Large Class of Engineering Drawings" *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 4. pp 868 - 890. 1997
61. Y. Zheng, H. Li, and D. Doermann, "The Segmentation and Identification of Handwriting in Noisy Document Images"", *Document Analysis Systems*, LNCS Vol. 2423, Springer-Verlag, pp. 95-105, 2002.

Appendix A: Useful Algorithms.

Algorithm 1: *initializeGraph*
Algoirthm 2: *initialzeIndividual*
Algorithm 3: *mutation*
Algoirthm 4: *crossover*

Algorithm 1 *initializeGraph* is used to create a graph from a single pixel width line image.

```
initializeGraph()
    G is empty graph
    I is thinned input image

    for x is 0 to I.width
        for y is 0 to I.height
            if I.pixel(x,y) equals ink then
                c is number of neighbours of I.pixel(x,y)
                if c not equal 2 then
                    G.addNode (x,y)
                end if c not equal 2
            end if I.pixel(x,y) is ink
        end for x
    end for y

    for n is 0 to G.numNodes()
        N1 is G.node(n)
        for d is direction N, NE, E, SE, S, SW, W, NW
            if an I.pixel(N1.x(), N1.y()) has neighbour in direction d
                En is new edge from N1 to N2 in direction d
                ~En is equivalent to En, but is from N2 to N1
                if G does not contain En and G does not contain ~En then
                    G.addEdge (En)
                end if G does not contain En and G does not contain ~En
            end if I.pixel(N1.x(), N1.y()) has neighbour in direction d
        end for d
    end for n

    return G
end initializeGraph
```

Algorithm 2 *initializeIndividual* is used to create a unique individual given a balance parameter p. p is a number between 0 and 1.

```
initializeIndividual (double p)
    Neast is null
    Nwest is null

    Neast becomes eastern node with degree = 1
    Nwest becomes western node with degree = 1

    if Neast equals null or Nwest equals null
```

```
            Neast becomes eastern node
            Nwest becomes western node
        end if Neast equals null or Nwest equals null

        Deast is depthsFromBFS(Neast)
        Dwest is depthsFromBFS(Nwest)

        Seast is new segment
        Swest is new segment
        Sextra is new segment
        Ind is new Individual made up of Seast, Swest, and Sextra

        if p < 0.5
            deast is maximum in Deast
            dcut is p * deast
            for i is 0 to dcut
                if edge e has depth i in Deast then
                    Seast.add (e)
            end for i
        end if p < 0.5
        else
            dwest is maximum in Dwest
            dcut is (1-p) * dwest
            for i is 0 to dcut
                if edge e has depth i in Dwest then
                    Swest.add (e)
            end for i
        end else

        for e is all edges in G
            if e is not assigned to Seast ,Swest, or Sextra then
                if depth of e in Deast < depth of e in Dwest then
                    Seast.add (e)
                end if depth of e in Deast < depth of e in Dwest
                else if depth of e in Deast > depth of e in Dwest then
                    Swest.add (e)
                end else if depth of e in Deast > depth of e in Dwest
                else
                    Sextra.add (e)
                end else
            end if e is not assigned
        end for e

        return Ind
end initializeIndividual
```

Algorithm 3 *mutation* is used to mutate a single individual Ind into something new.

```
mutation (Ind)
        if Ind has 0 segments matched
            Sshrink is randomly selected, unmatched segment in Ind
            Sgrow is highest index unmatched segment in Ind and is not Sshrink

            D1 is set of degree one nodes in Sshrink
            if D1 is empty
```

```
                Er is random edge in Sshrink
            end if D1 is empty
            else
                N is random node in D1
                Er is edge connecting N to the rest Sshrink
            end else
    end if Ind has 0 segments matched
    else
        amatched is average symbol area of all matched symbols in Ind
        path is random from {0, 1}
        if path is 0
            Segs is set of all unmatched segments in Ind that satisfies
            Sshrink is random from Segs
            Sgrow is highest index unmatched segment in Ind and is not Sshrink
            D1 is set of degree one nodes in Sshrink
            if D1 is empty
                Er is random edge in Sshrink
            end if D1 is empty
            else
                N is random node in D1
                Er is edge connecting N to the rest Sshrink
            end else
        end if path is 0
        else if path is 1
            find segments p1 and p2 such that p1 has area
                $a_{matched} + (a_{matched} * p_{close})$ ,
                p2 has area <$a_{matched} + (a_{matched} * p_{close})$ ,
                and $p_1 \cup p_2$ has one component
            Ind.removeSegment (p1)
            Ind.removeSegment (p2)
            Ind.addSegment ( $p_1 \cup p_2$ )
        end else if path is 1
    end else
    return Ind
end mutation
```

Algorithm 4 *crossover* is used to share matched segments between individuals Ind1 and Ind2.

```
crossover (Ind1, Ind2)
    G is original graph
    Indchild is empty individual

    for S is each segment in Ind1
        if S is a symbol then
            Ind.addSegment (S)
        end if S is a symbol
    end for S

    for S is each segment in Ind2
        if S is a symbol then
            Indchild.addSegment (S)
        end if S is a symbol
    end for S

    Sunassigned is new segment
    for E is each edge in G
        if E does not belong to a segment in Indchild
            Sunassigned.addEdge (E)
        end if E does not belong to a segment
    end for E

    if Sunassigned is not empty
        Indchild.addSegment (Sunassigned)
    end if Sunassignedis not empty

    return Indchild
end crossover
```

Appendix B:

This appendix contains the results for the experiments on various input images, from various domains. The Appendix is separated into two sections:

- The first seciton presents the accuracy and duration of active evolution.
- The second section presents the epoch of first symbol found, and the epoch of termination data.

The separation is necessary to present the charts clearly on standard size paper.

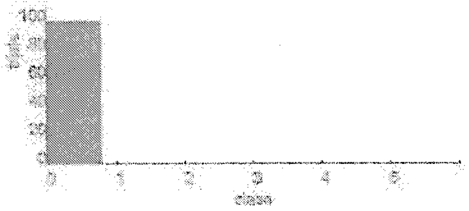
Analysis of Numerous Trials on Each Input Image

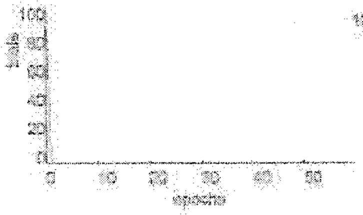
Input Image

Accuracy Rating

Distribution of Active Evolution

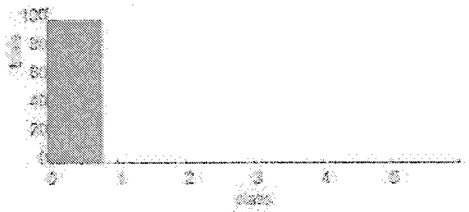
aa.jpg
fig. 0

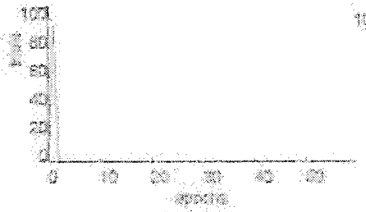




100/100 results

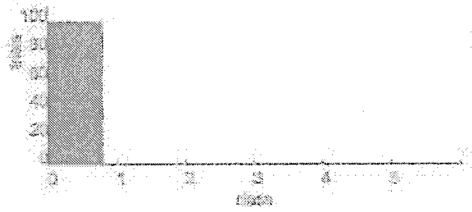
ab.jpg
fig. 1

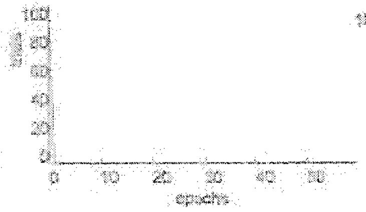




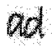
100/100 results

ac.jpg
fig. 2

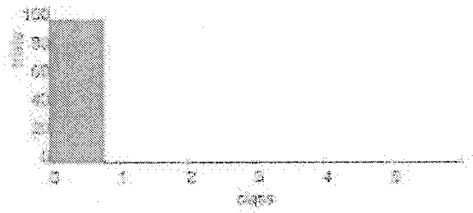


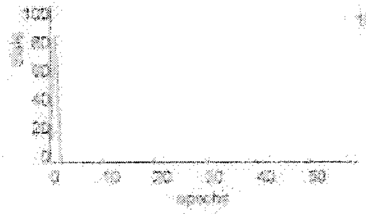


100/100 results



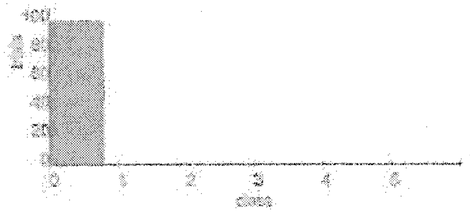
ad.jpg
fig. 3

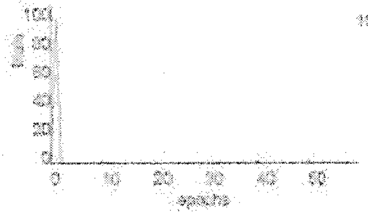




100/100 results

ae.jpg
fig. 4





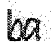
100/100 results

Analysis of Numerous Trials on Each Input Image

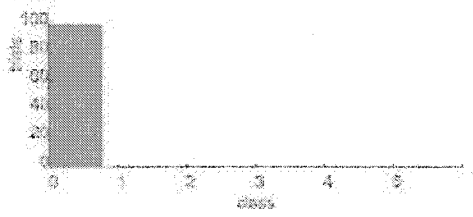
Input Image

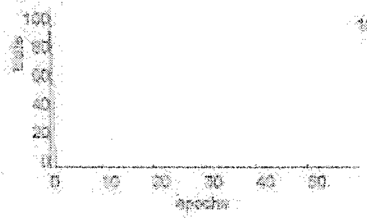
Accuracy Rating

Distribution of Active Evolution

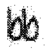


ba.jpg
fig. 5

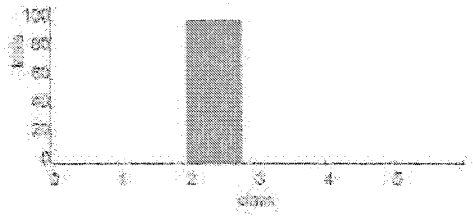


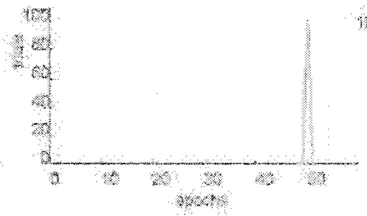


100/100 results



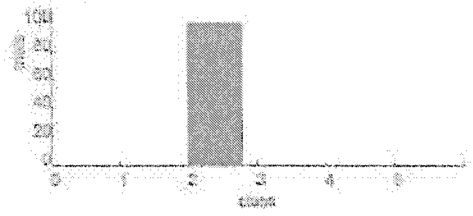
bb.jpg
fig. 6

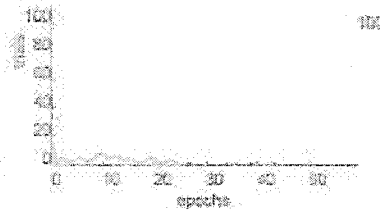




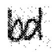
100/100 results

bc.jpg
fig. 7

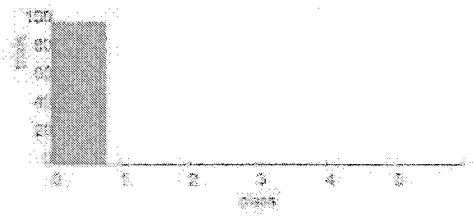


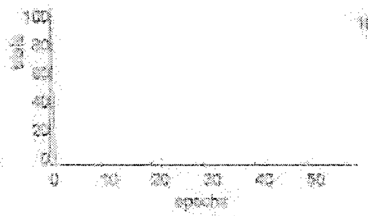


100/100 results



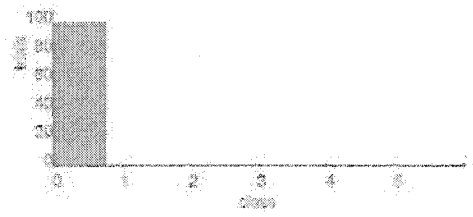
bd.jpg
fig. 8

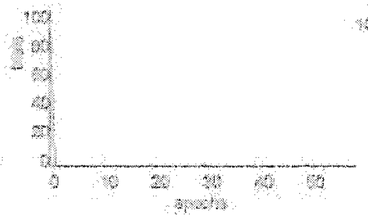




100/100 results

be.jpg
fig. 9





100/100 results

Analysis of Numerous Trials on Each Input Image

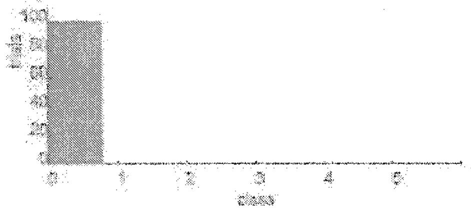
Input Image

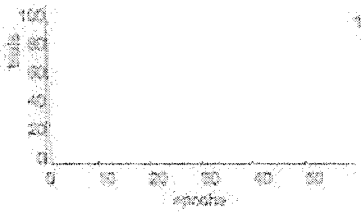
Accuracy Rating

Distribution of Active Evolution

ca

ca.jpg
fig. 10

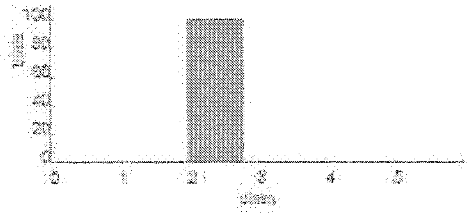


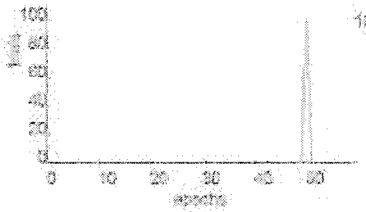


100/100 results

cb

cb.jpg
fig. 11

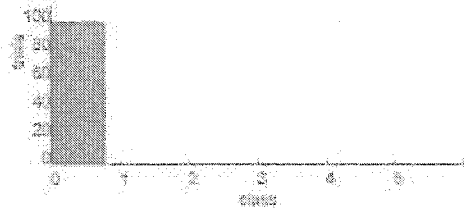


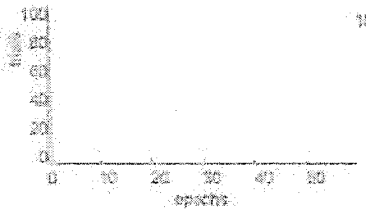


100/100 results

cc

cc.jpg
fig. 12

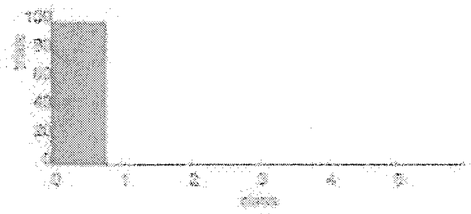


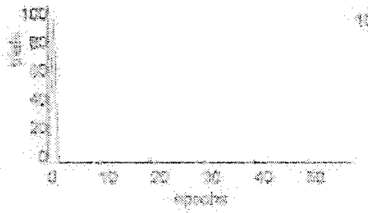


100/100 results

cd

cd.jpg
fig. 13

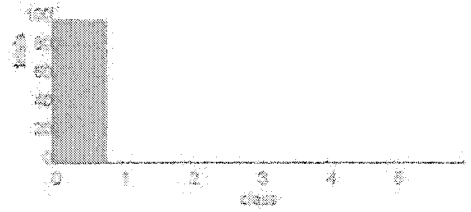


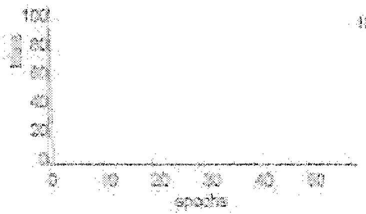


100/100 results

ce

ce.jpg
fig. 14

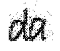


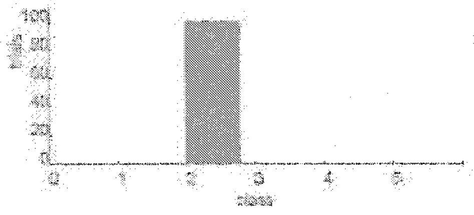


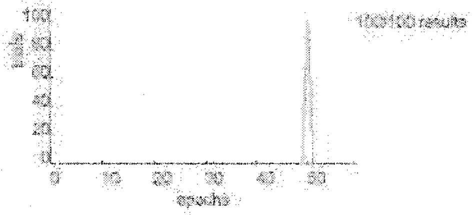
100/100 results

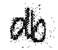
Analysis of Numerous Trials on Each Input Image

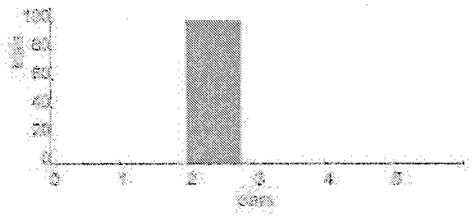
Input Image Accuracy Rating Distribution of Active Evolution

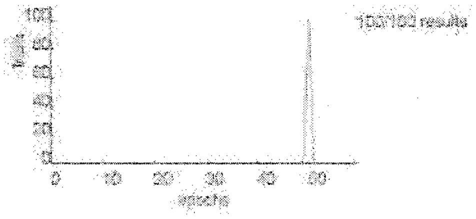

da.jpg
fig. 15

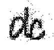


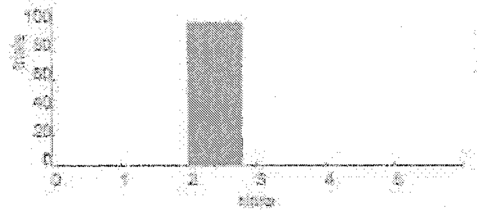


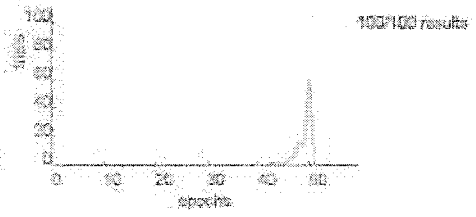

db.jpg
fig. 16

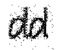


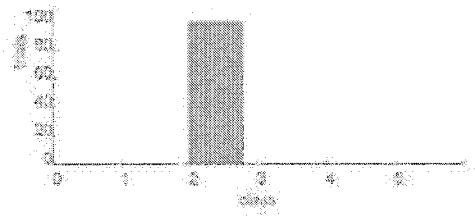


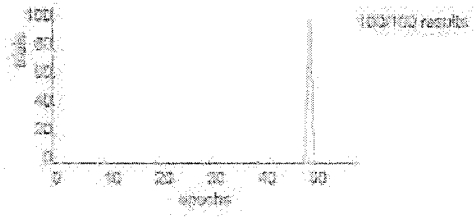

dc.jpg
fig. 17

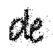


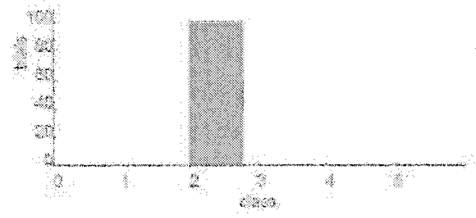


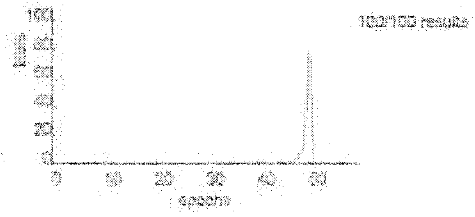

dd.jpg
fig. 18






de.jpg
fig. 19



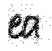


Analysis of Numerous Trials on Each Input Image

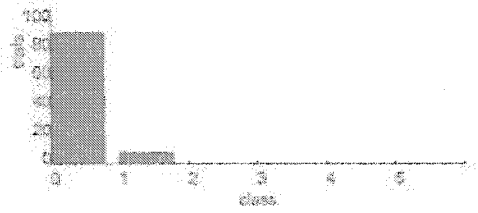
Input Image

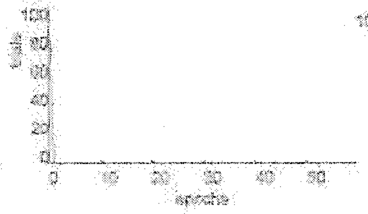
Accuracy Rating

Distribution of Active Evolution



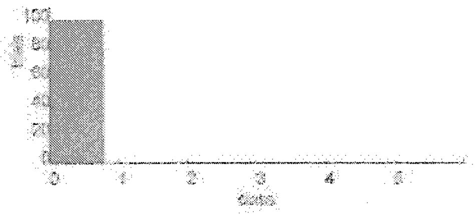
ea.jpg
fig. 20

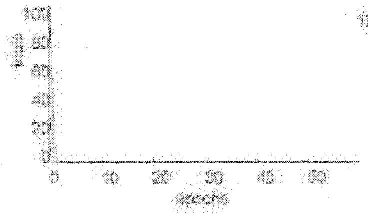




100/100 results

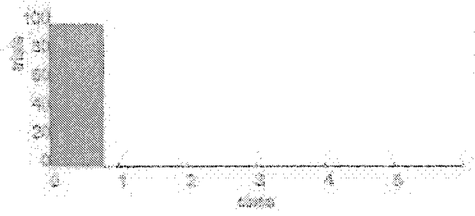
eb.jpg
fig. 21

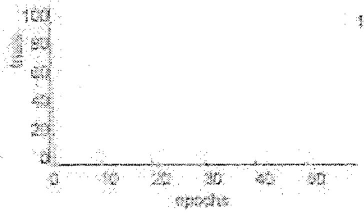




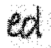
100/100 results

ec.jpg
fig. 22

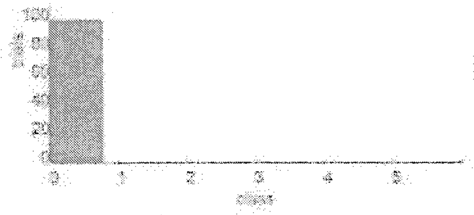


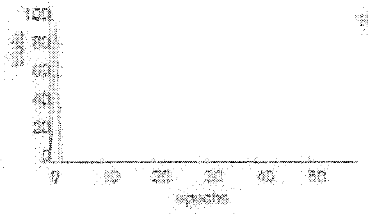


100/100 results



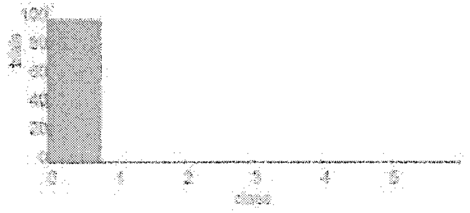
ed.jpg
fig. 23

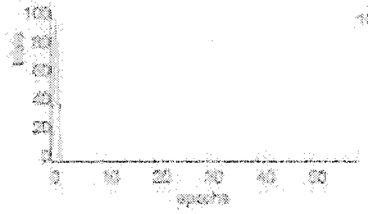




100/100 results

ee.jpg
fig. 24





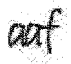
100/100 results

Analysis of Numerous Trials on Each Input Image

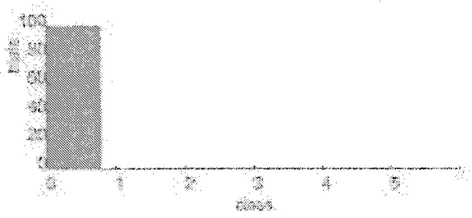
Input Image

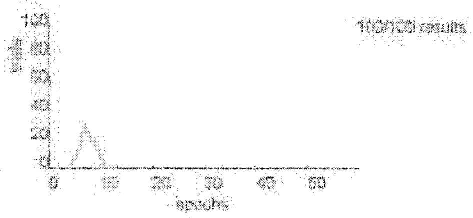
Accuracy Rating

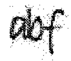
Distribution of Active Evolution



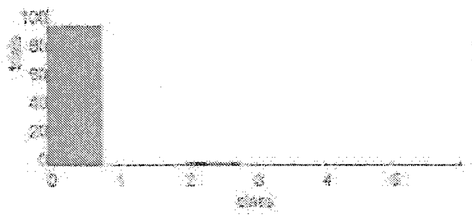
aaf.jpg
fig. 25

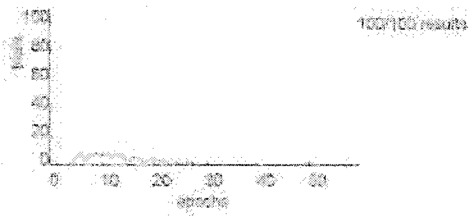


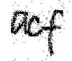




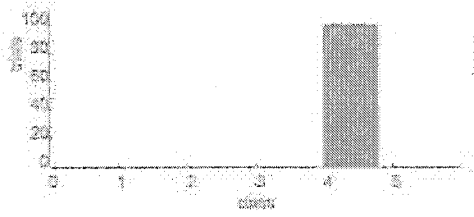
abf.jpg
fig. 26

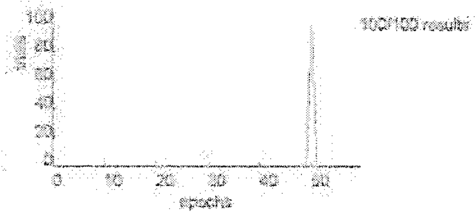


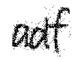




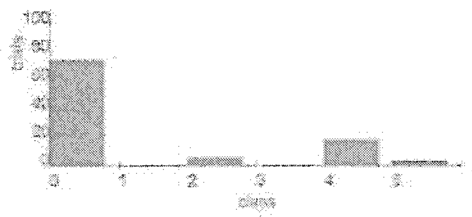
acf.jpg
fig. 27

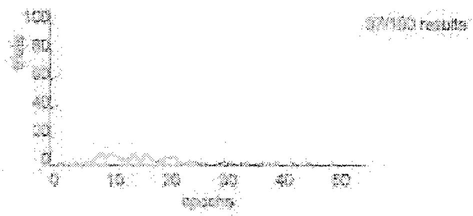


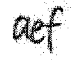




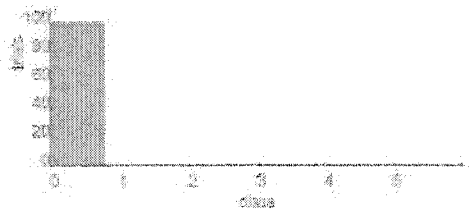
adf.jpg
fig. 28

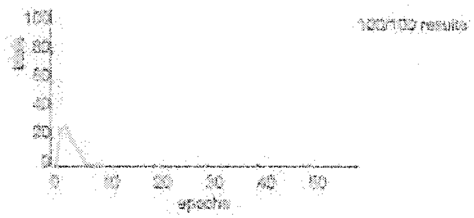






aef.jpg
fig. 29



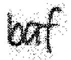


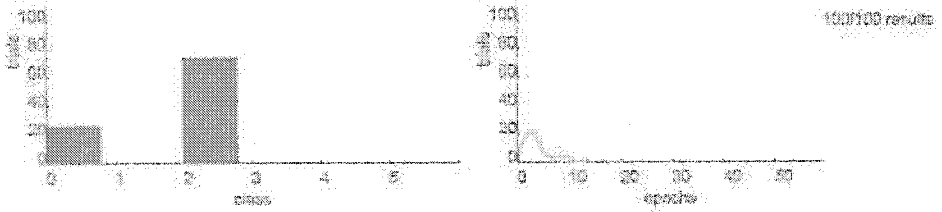
Analysis of Numerous Trials on Each Input Image

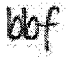
Input Image

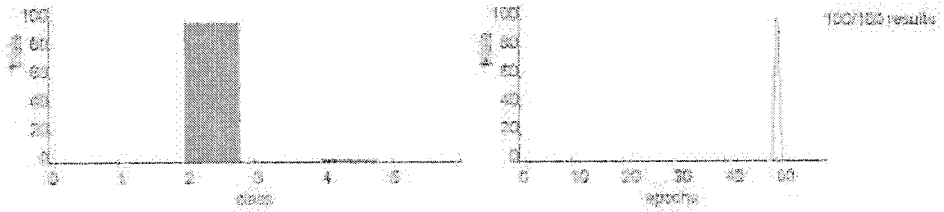
Accuracy Rating

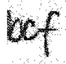
Distribution of Active Evolution

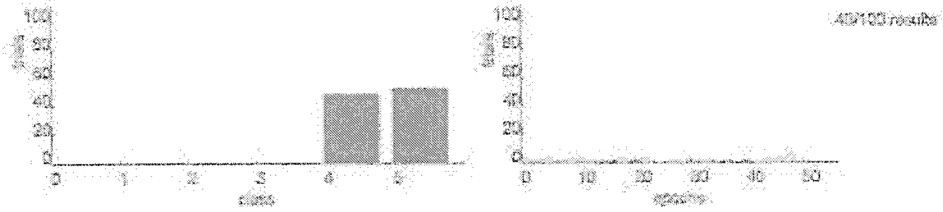


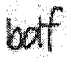
baf.jpg
fig. 30


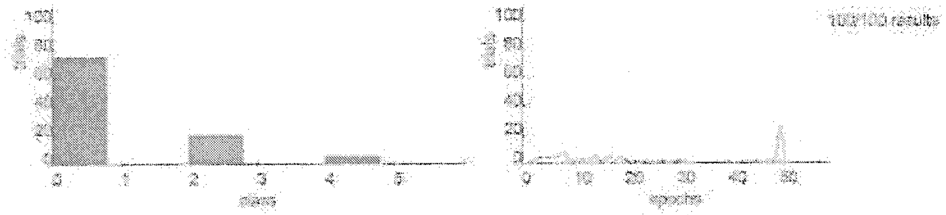


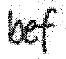
bbf.jpg
fig. 31


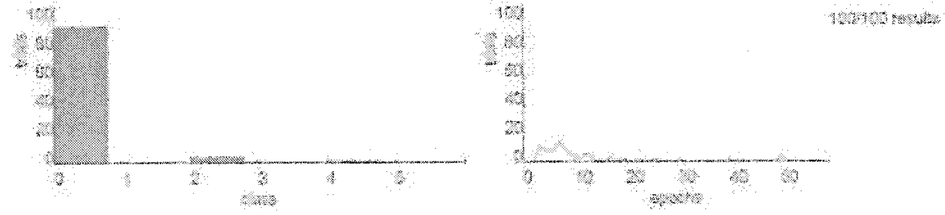


bcf.jpg
fig. 32




bdf.jpg
fig. 33




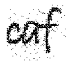
bef.jpg
fig. 34


Analysis of Numerous Trials on Each Input Image

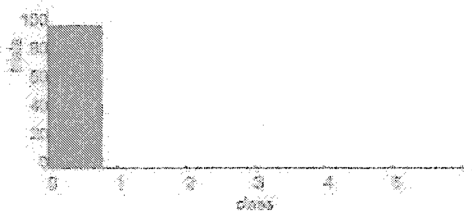
Input Image

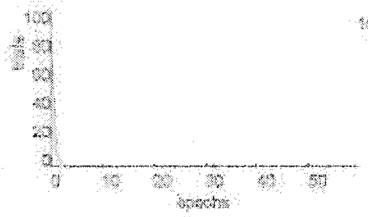
Accuracy Rating

Distribution of Active Evolution

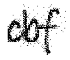


caf.jpg
fig. 35

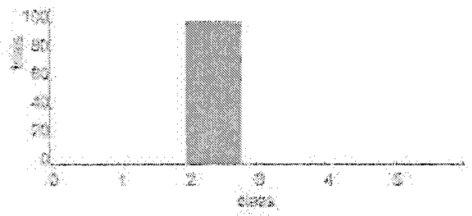


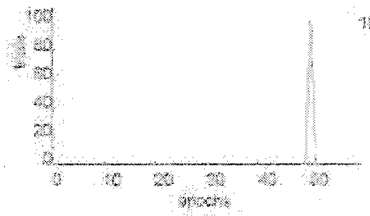


100/100 results

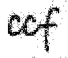


cbf.jpg
fig. 36

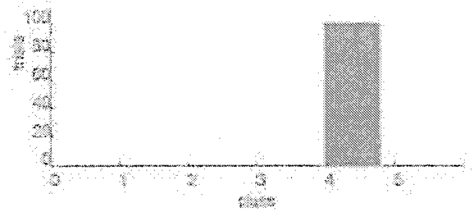


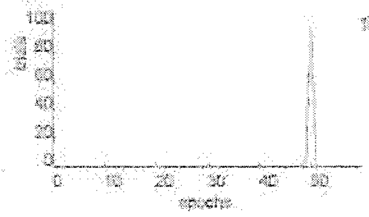


100/100 results

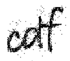


ccf.jpg
fig. 37

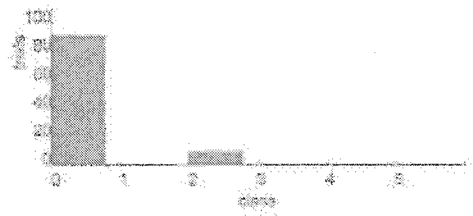


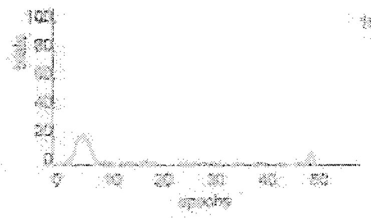


100/100 results

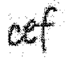


cdf.jpg
fig. 38

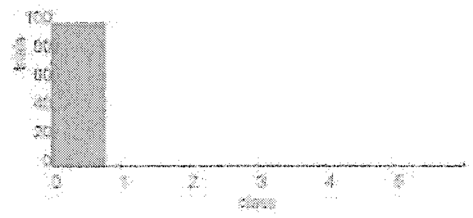


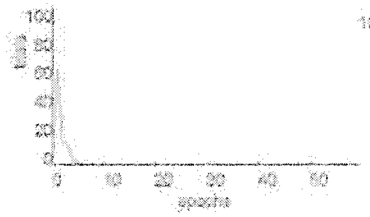


100/100 results



cef.jpg
fig. 39





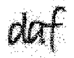
100/100 results

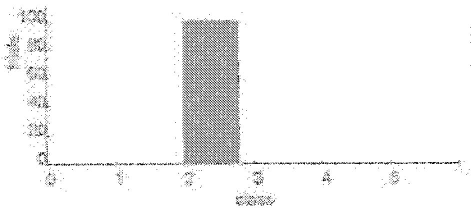
Analysis of Numerous Trials on Each Input Image

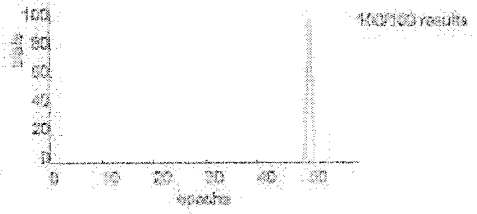
Input image

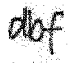
Accuracy Rating

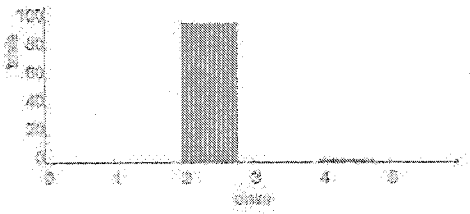
Distribution of Active Evolution

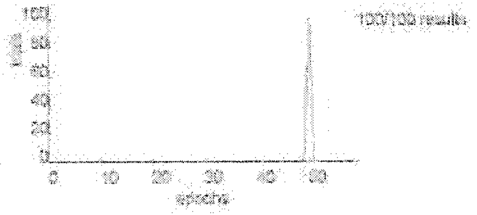

daf.jpg
fig. 40

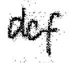


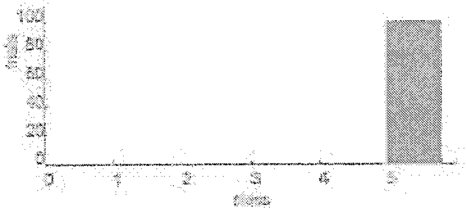


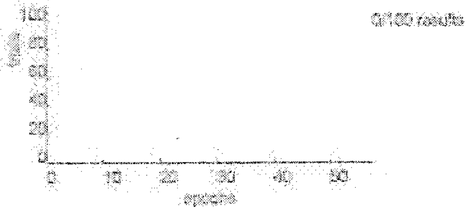

dbf.jpg
fig. 41

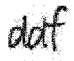


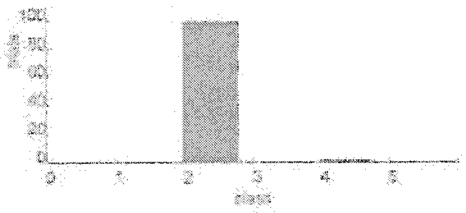


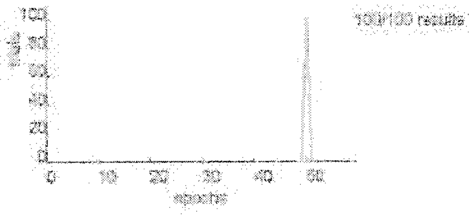

dcf.jpg
fig. 42

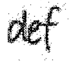


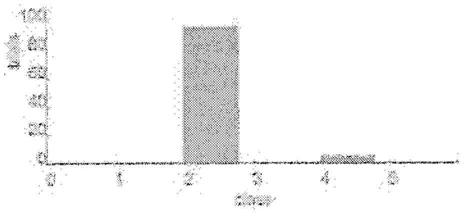


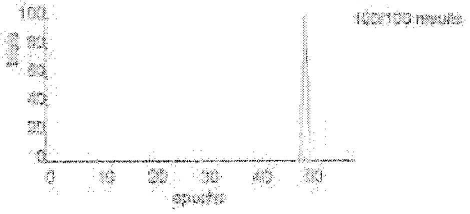

ddf.jpg
fig. 43






def.jpg
fig. 44



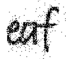


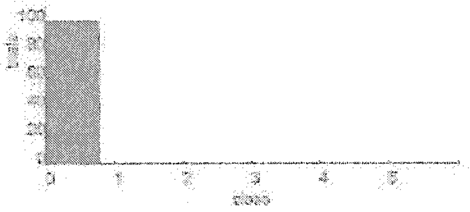
Analysis of Numerous Trials on Each Input Image

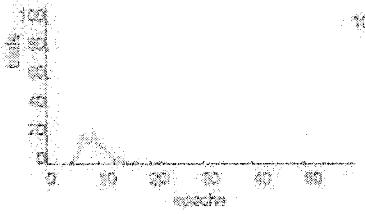
Input Image

Accuracy Rating

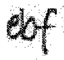
Distribution of Active Evolution

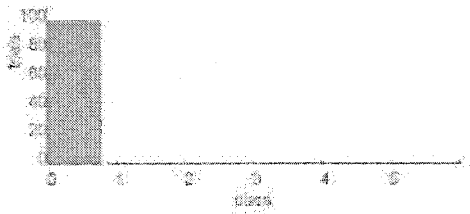

eaf.jpg
fig. 45

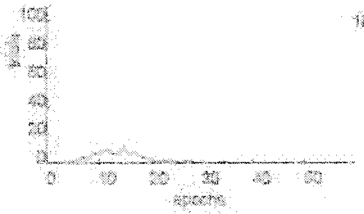




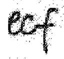
100/100 results

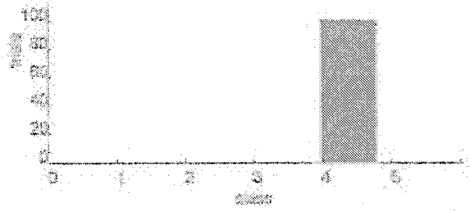

ebf.jpg
fig. 46

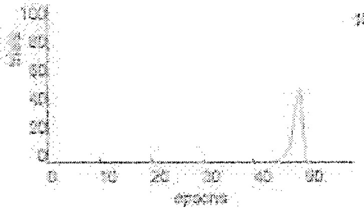




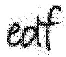
100/100 results

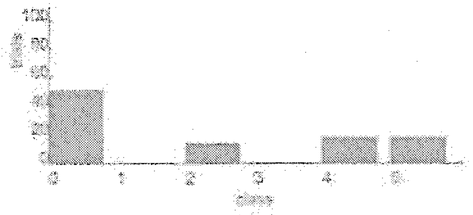

ecf.jpg
fig. 47

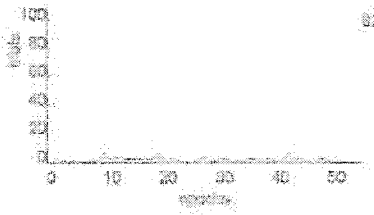




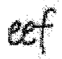
100/100 results

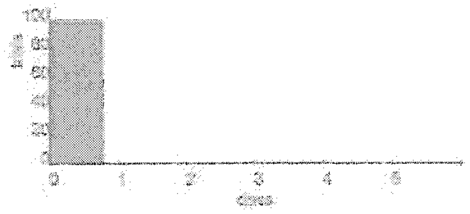

edf.jpg
fig. 48

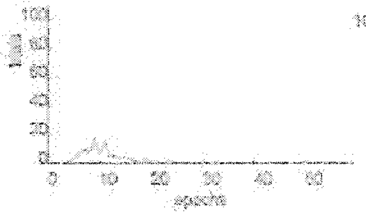




83/100 results


eef.jpg
fig. 49





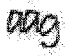
100/100 results

Analysis of Numerous Trials on Each Input Image

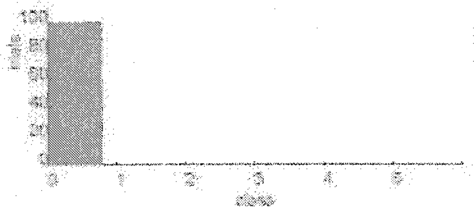
Input Image

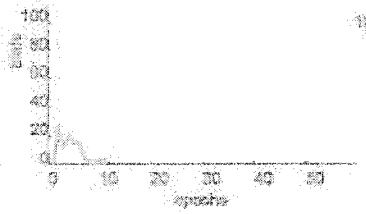
Accuracy Rating

Distribution of Active Evolution

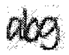


aag.jpg
fig. 50

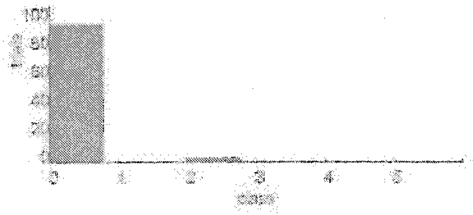


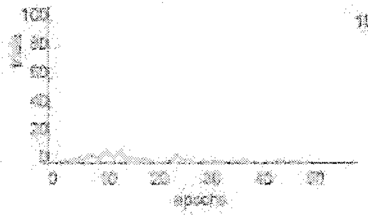


100/100 results

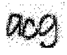


abg.jpg
fig. 51

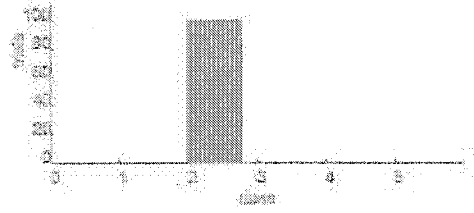


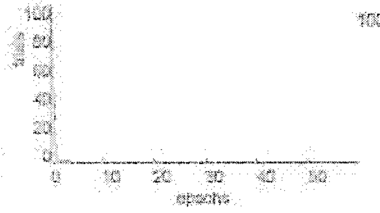


100/100 results

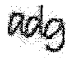


acg.jpg
fig. 52

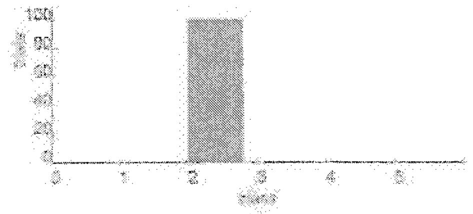


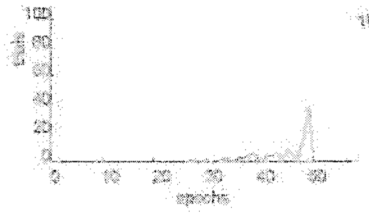


100/100 results

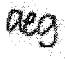


adg.jpg
fig. 53

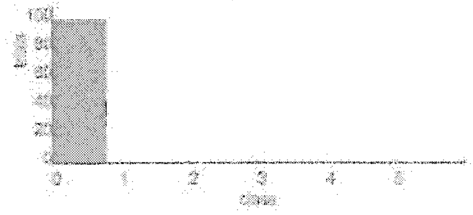


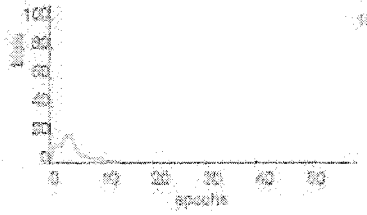


100/100 results



aeg.jpg
fig. 54





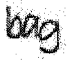
100/100 results

Analysis of Numerous Trials on Each Input Image

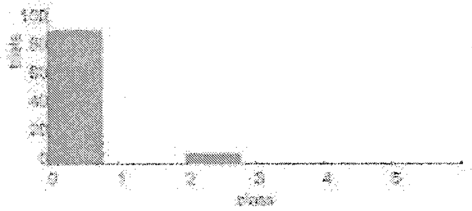
Input Image

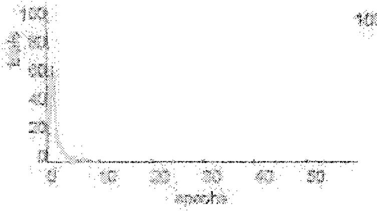
Accuracy Rating

Distribution of Active Evolution

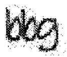


bag.jpg
fig. 55

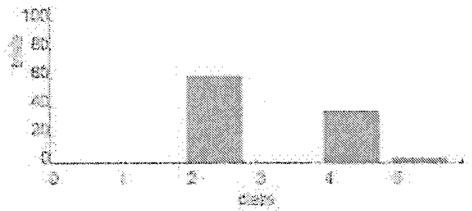


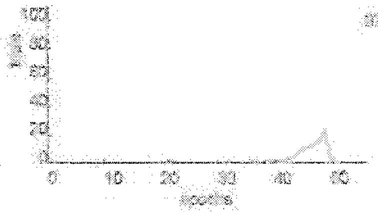


100/100 results

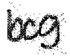


bbg.jpg
fig. 56

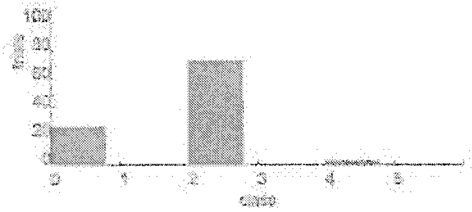


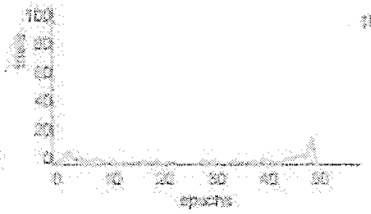


97/100 results

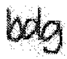


bcg.jpg
fig. 57

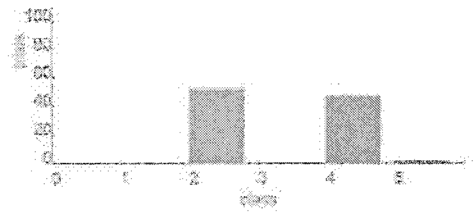


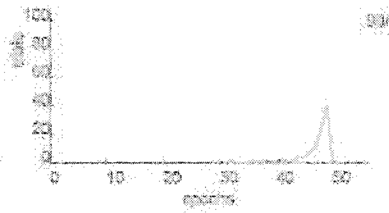


100/100 results

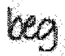


bdg.jpg
fig. 58

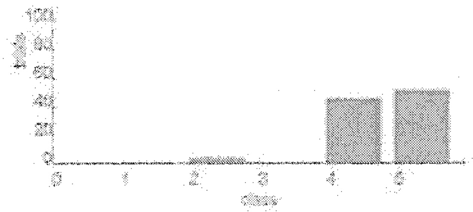


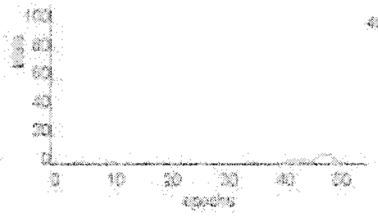


98/100 results



beg.jpg
fig. 59





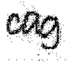
40/100 results

Analysis of Numerous Trials on Each Input Image

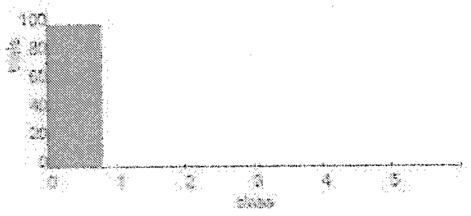
Input Image

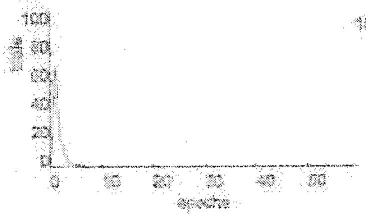
Accuracy Rating

Distribution of Active Evolution

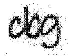


cag.jpg
fig. 60

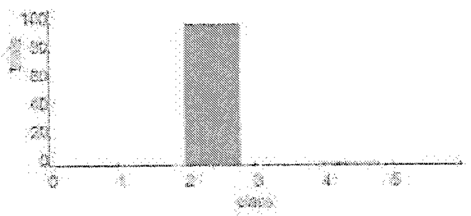


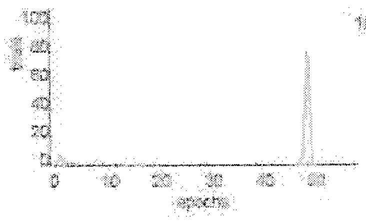


100/100 results

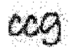


cbg.jpg
fig. 61

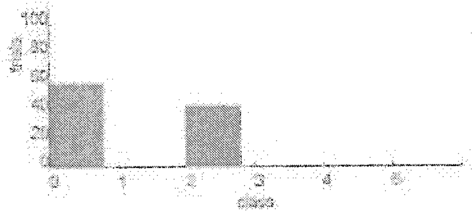


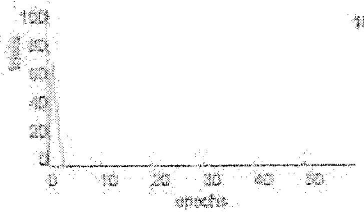


100/100 results

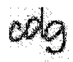


ccg.jpg
fig. 62

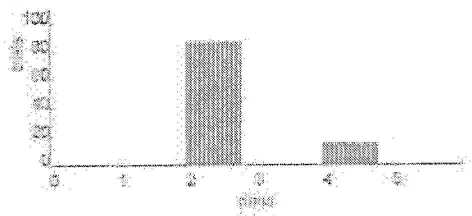


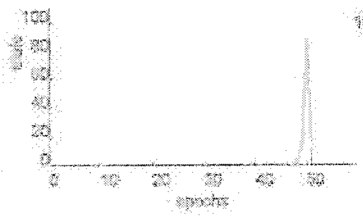


100/100 results

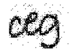


cdg.jpg
fig. 63

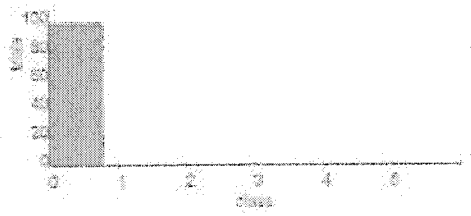


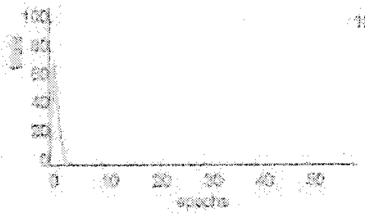


100/100 results



ceg.jpg
fig. 64

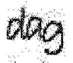




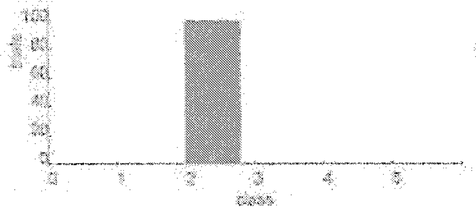
100/100 results

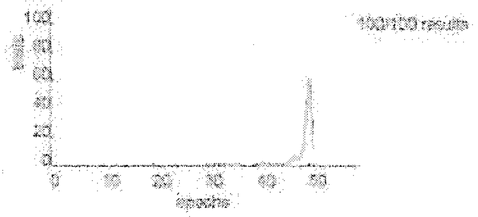
Analysis of Numerous Trials on Each Input Image

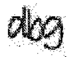
Input Image Accuracy Rating Distribution of Active Evolution



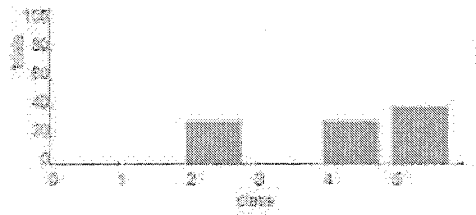
dag.jpg
fig. 65

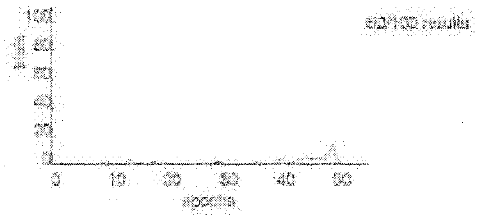


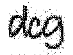




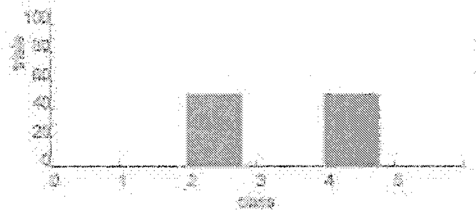
dbg.jpg
fig. 66

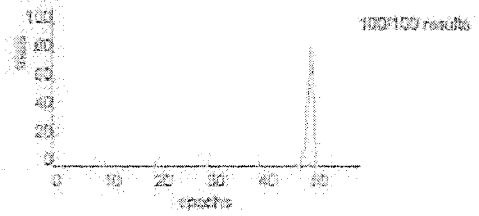


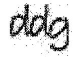




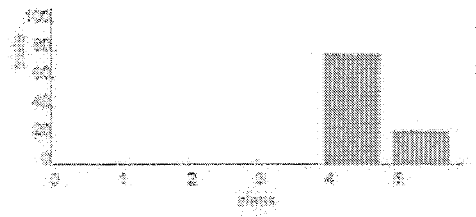
dcg.jpg
fig. 67

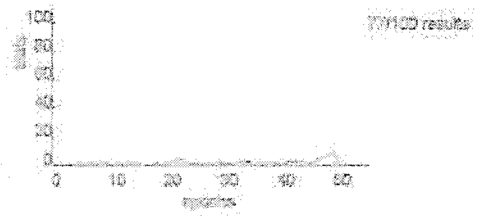


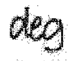




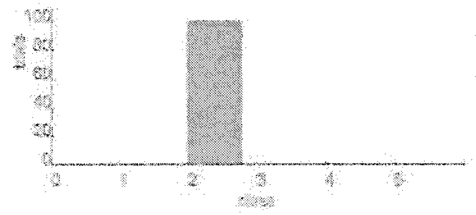
ddg.jpg
fig. 68

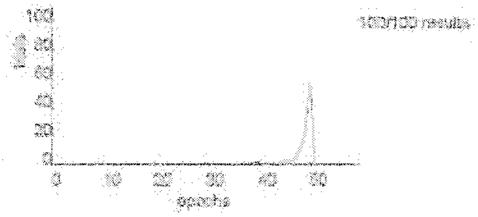






deg.jpg
fig. 69



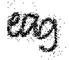


Analysis of Numerous Trials on Each Input Image

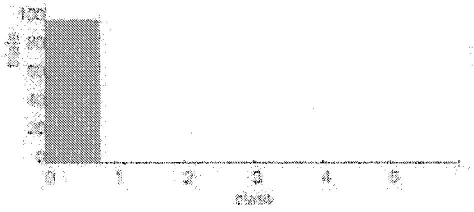
Input Image

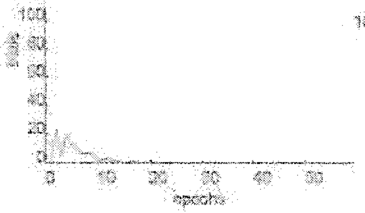
Accuracy Rating

Distribution of Active Evolution

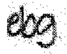


eag.jpg
fig. 70

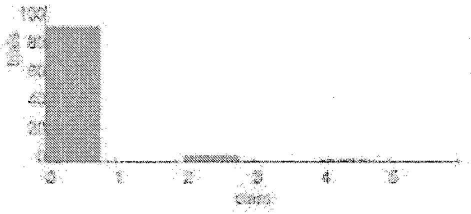


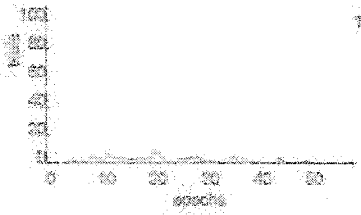


100/100 results

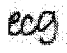


ebg.jpg
fig. 71

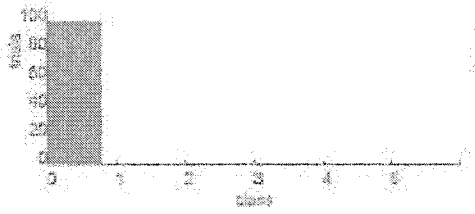


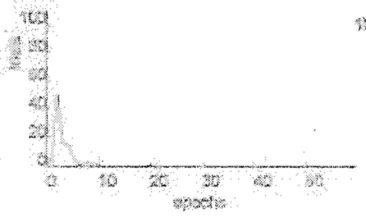


100/100 results

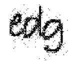


ecg.jpg
fig. 72

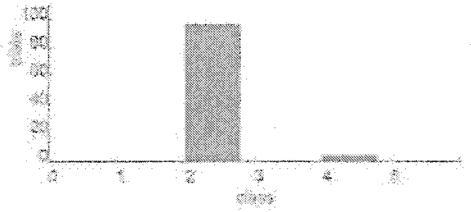


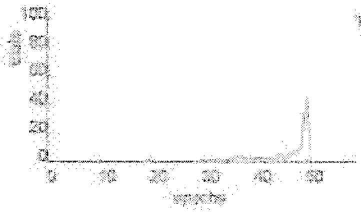


100/100 results

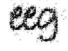


edg.jpg
fig. 73

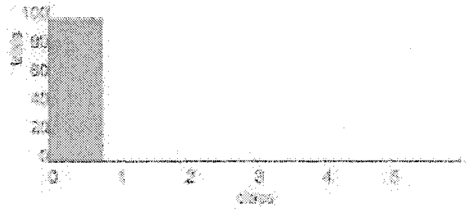


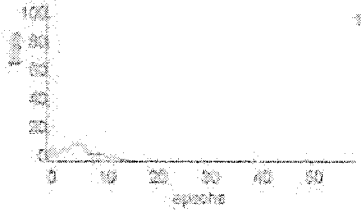


100/100 results



eeg.jpg
fig. 74





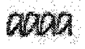
100/100 results

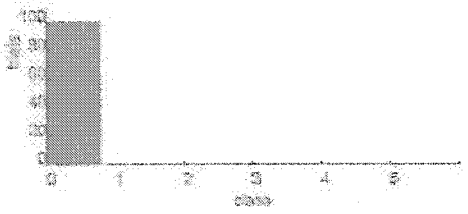
Analysis of Numerous Trials on Each Input Image

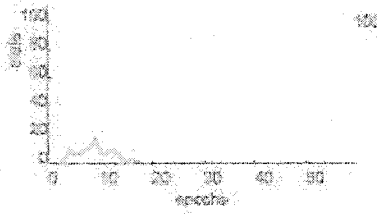
Input Image

Accuracy Rating

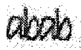
Distribution of Active Evolution

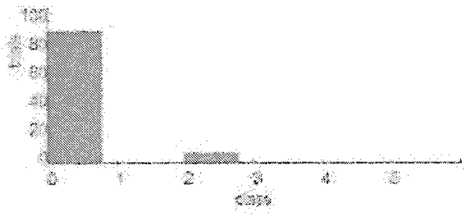

aaaa.jpg
fig. 75

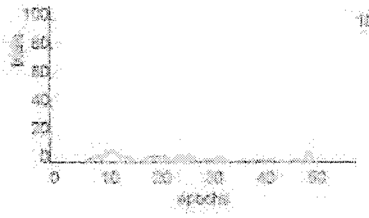




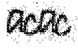
100/100 results

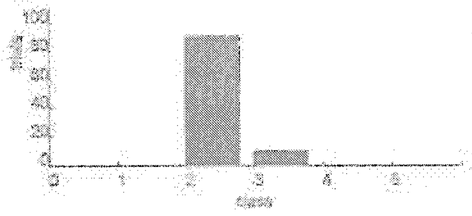

abab.jpg
fig. 76

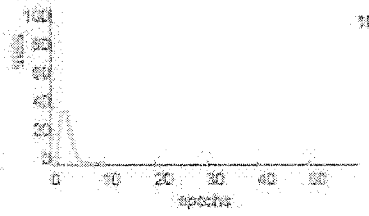




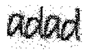
100/100 results

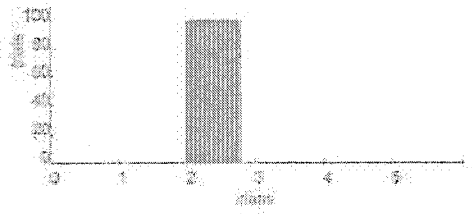

acac.jpg
fig. 77

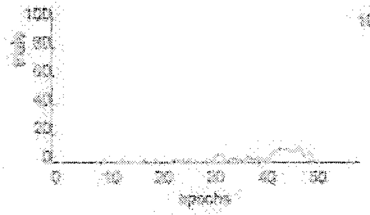




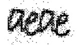
100/100 results

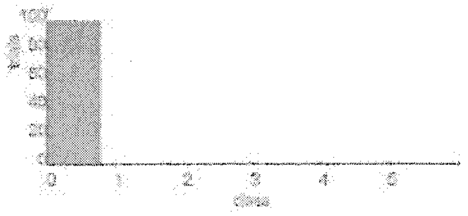

adad.jpg
fig. 78

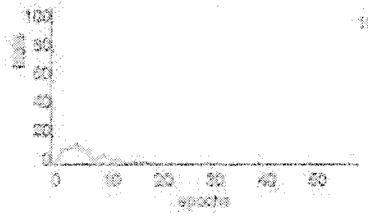




100/100 results


aeae.jpg
fig. 79





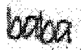
100/100 results

Analysis of Numerous Trials on Each Input Image

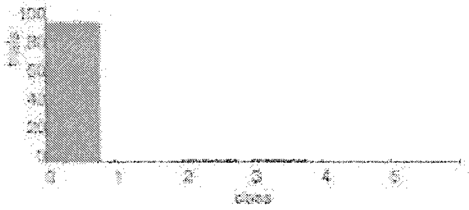
Input Image

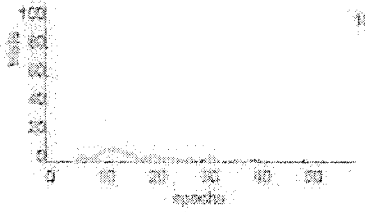
Accuracy Rating

Distribution of Active Evolution

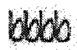


baba.jpg
fig. 80

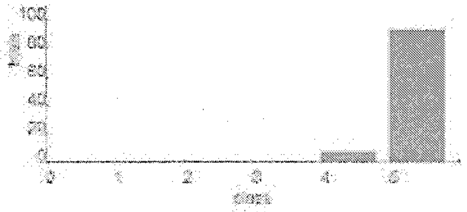


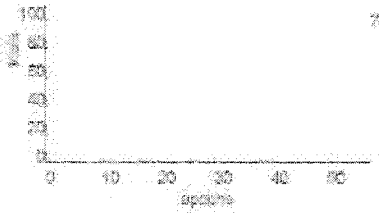


100/100 results

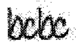


bbbb.jpg
fig. 81

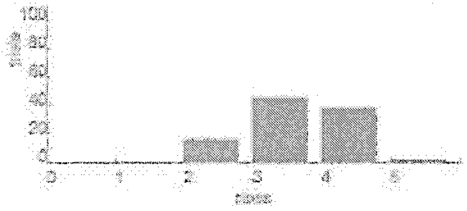


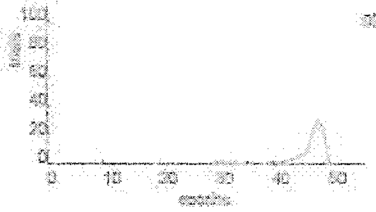


7/100 results

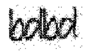


bcbc.jpg
fig. 82

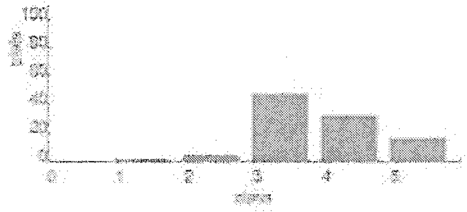


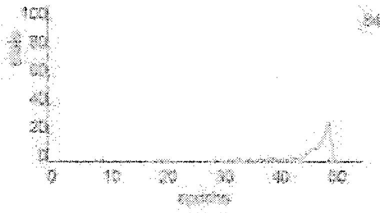


30/100 results

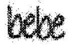


bdbd.jpg
fig. 83

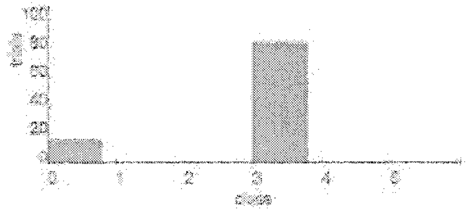


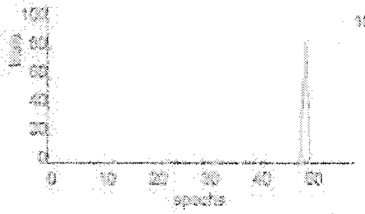


84/100 results



bebe.jpg
fig. 84





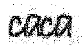
100/100 results

Analysis of Numerous Trials on Each Input Image

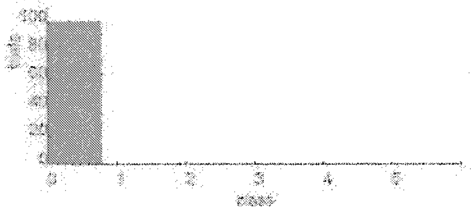
Input Image

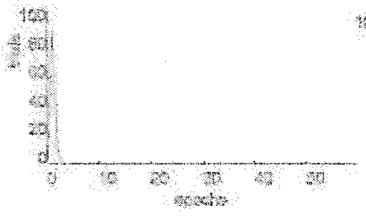
Accuracy Rating

Distribution of Active Evolution

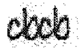


caca.jpg
fig. 85

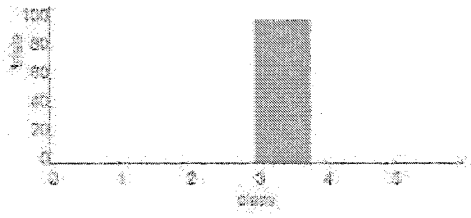


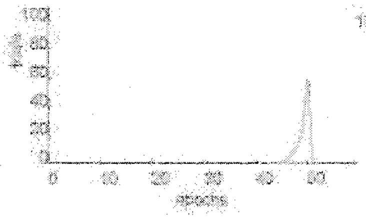


100/100 results

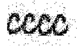


cbcb.jpg
fig. 86

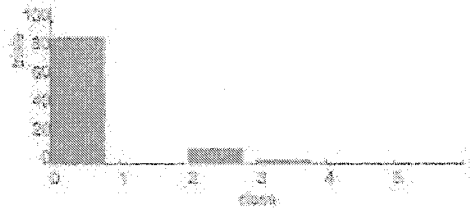


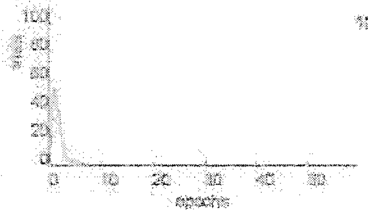


100/100 results

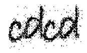


cccc.jpg
fig. 87

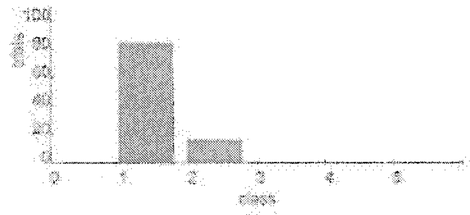


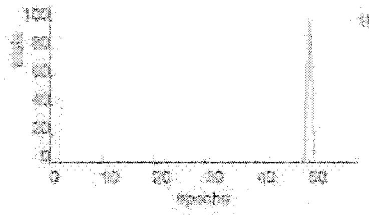


100/100 results

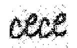


cdcd.jpg
fig. 88

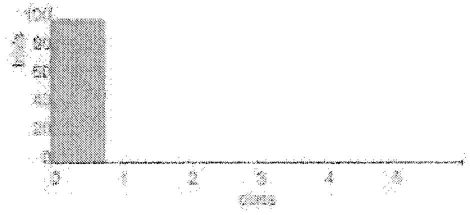


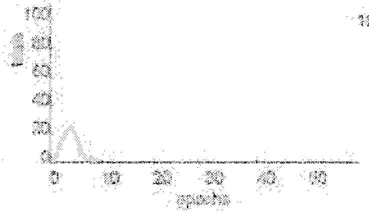


100/100 results



cece.jpg
fig. 89





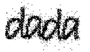
100/100 results

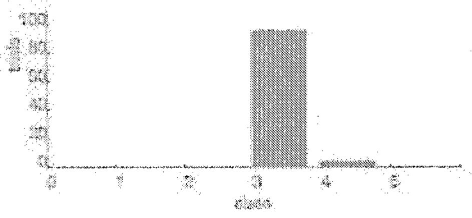
Analysis of Numerous Trials on Each Input Image

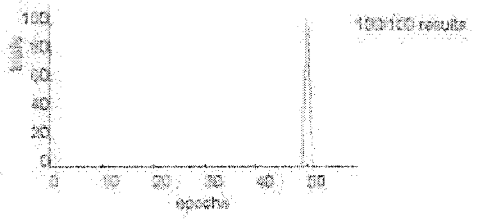
Input Image

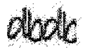
Accuracy Rating

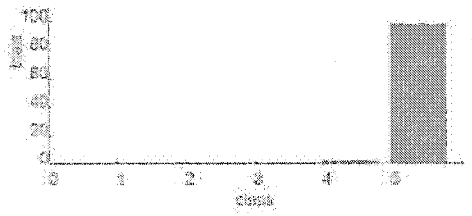
Distribution of Active Evolution

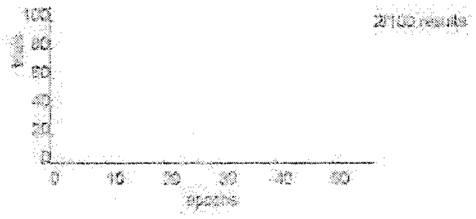

dada.jpg
fig. 90

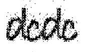


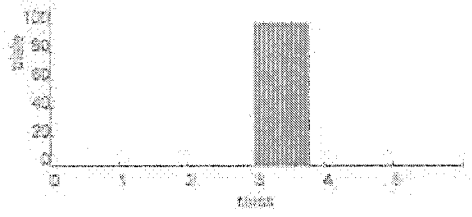


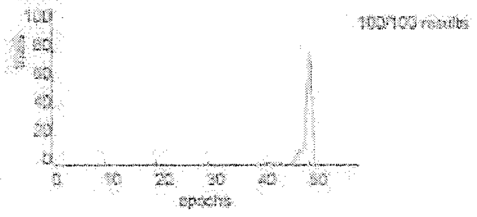

dbdb.jpg
fig. 91

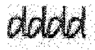


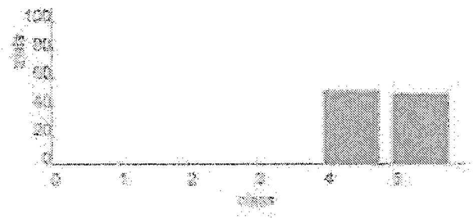


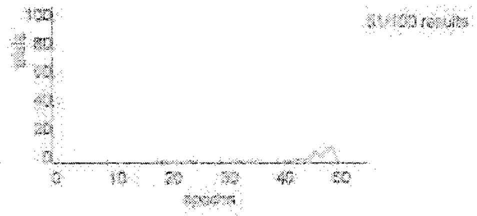

dcdc.jpg
fig. 92

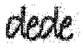


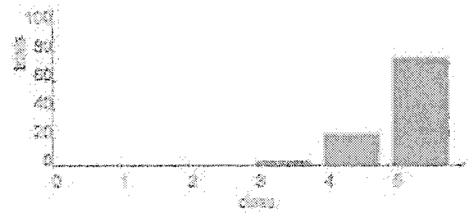


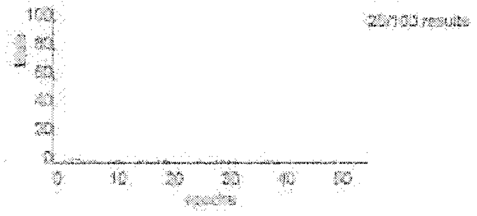

dddd.jpg
fig. 93






dede.jpg
fig. 94



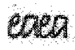


Analysis of Numerous Trials on Each Input Image

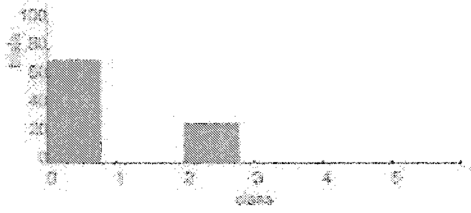
Input Image

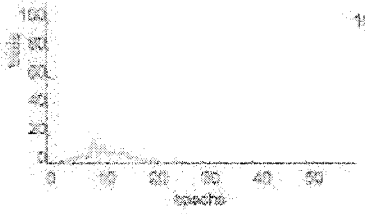
Accuracy Rating

Distribution of Active Evolution

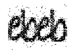


eaea.jpg
fig. 95

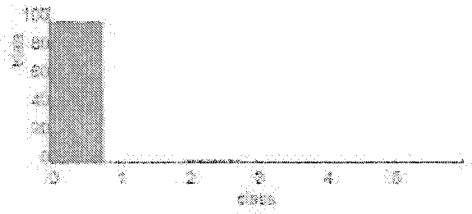


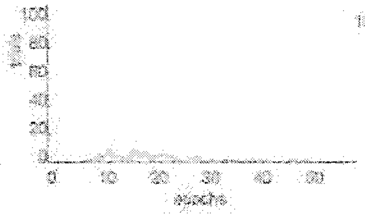


100/100 results

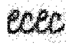


ebeb.jpg
fig. 96

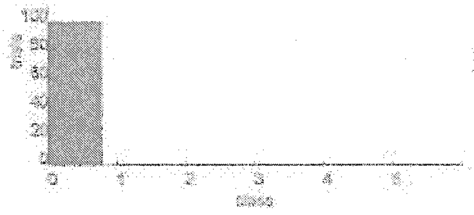


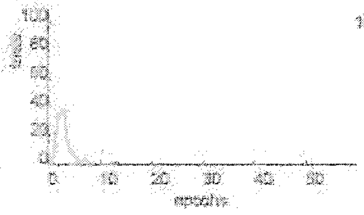


100/100 results

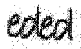


ecec.jpg
fig. 97

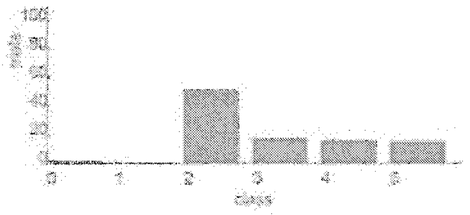


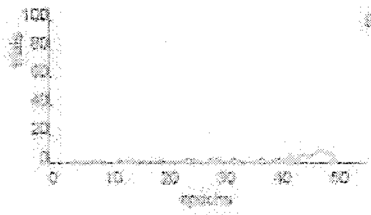


100/100 results

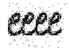


eded.jpg
fig. 98

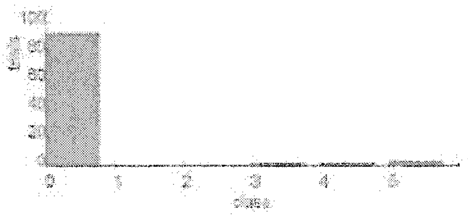


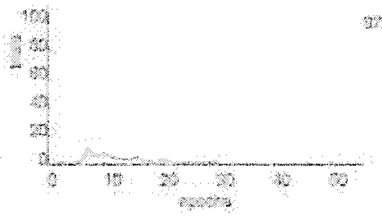


95/100 results



eeee.jpg
fig. 99





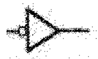
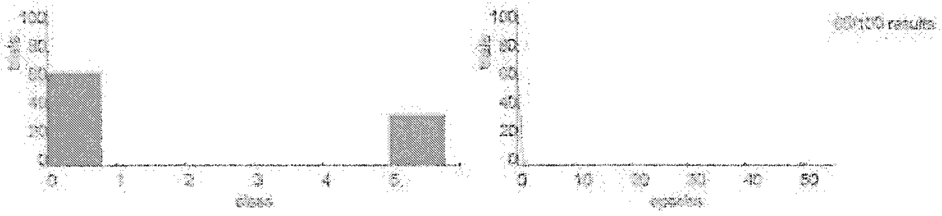
97/100 results

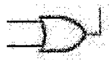
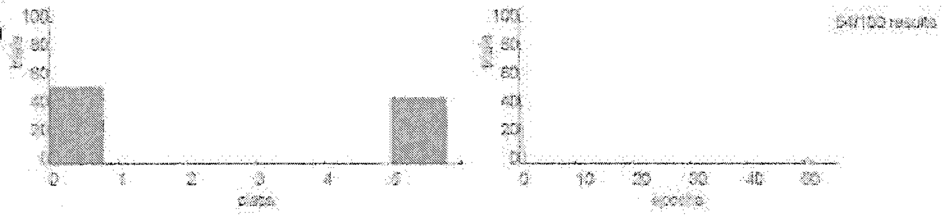
Analysis of Numerous Trials on Each Input Image

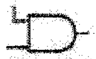
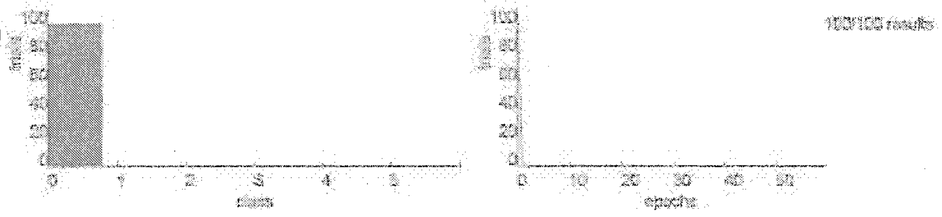
Input Image

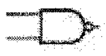
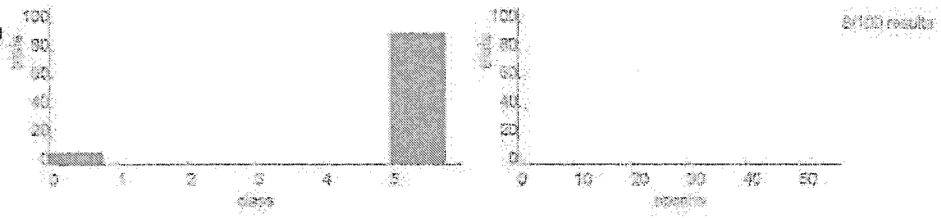
Accuracy Rating

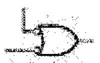
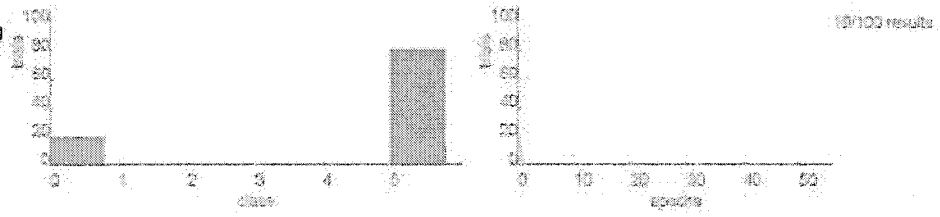
Distribution of Active Evolution


o0clean.jpg
fig. 100



o1clean.jpg
fig. 101



o2clean.jpg
fig. 102



o3clean.jpg
fig. 103


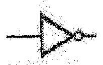

o4clean.jpg
fig. 104


Analysis of Numerous Trials on Each Input Image

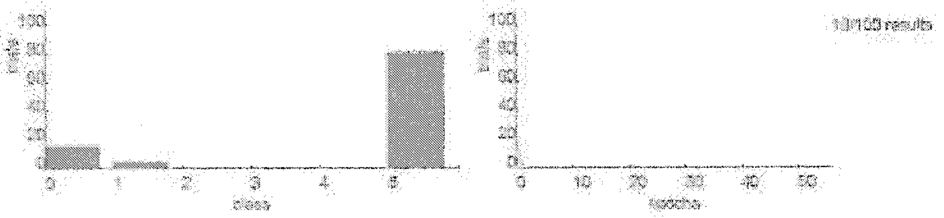
Input Image

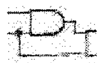
Accuracy Rating

Distribution of Active Evolution

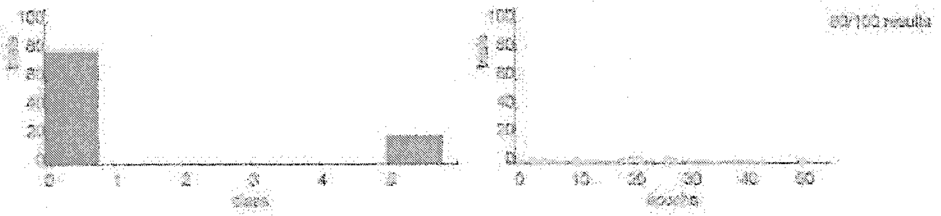


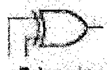
o5clean.jpg
fig. 105



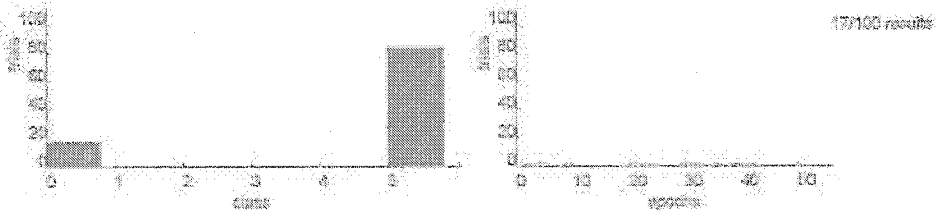


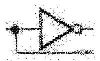
o6clean.jpg
fig. 106



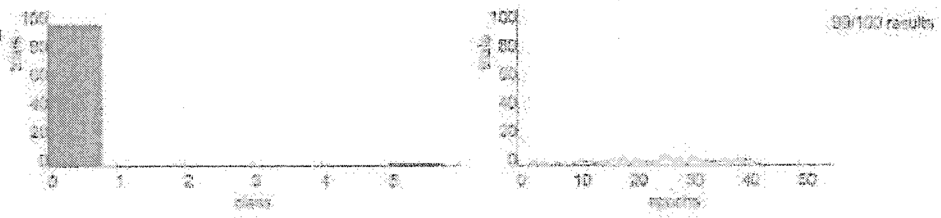


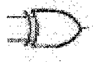
o7clean.jpg
fig. 107



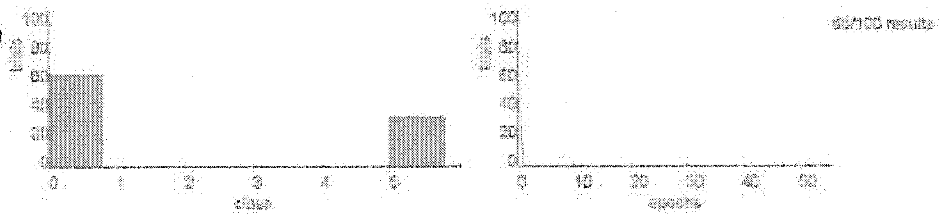


o8clean.jpg
fig. 108





o9clean.jpg
fig. 109

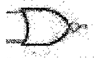


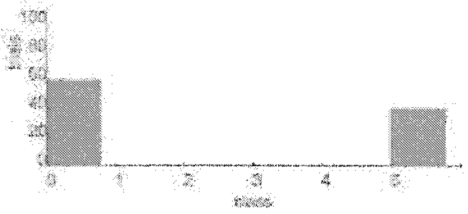
Analysis of Numerous Trials on Each Input Image

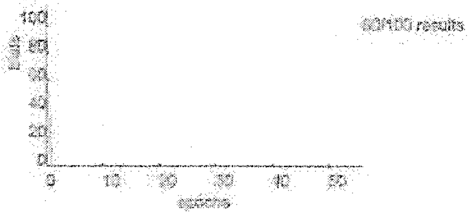
Input Image

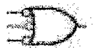
Accuracy Rating

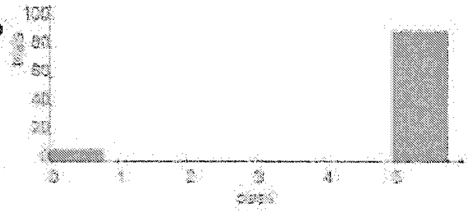
Distribution of Active Evolution

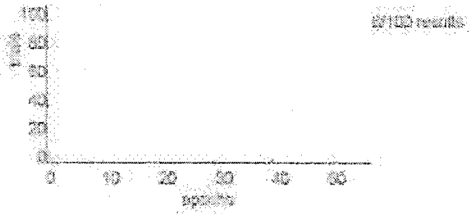

o10clean.jp
fig. 110

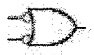


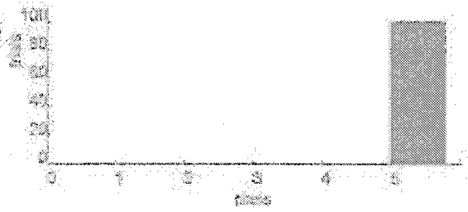


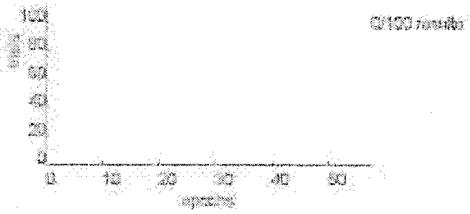

o11clean.jp
fig. 111

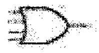


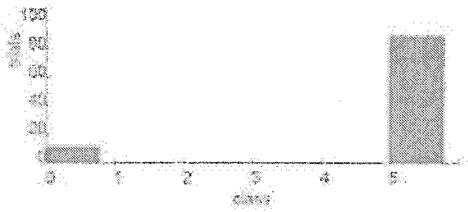


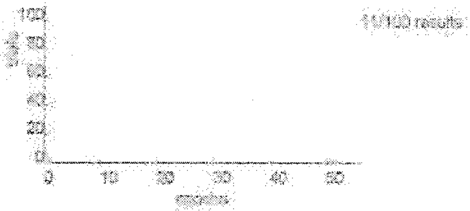

o12clean.jp
fig. 112

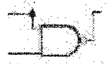


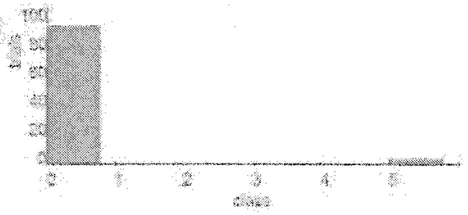


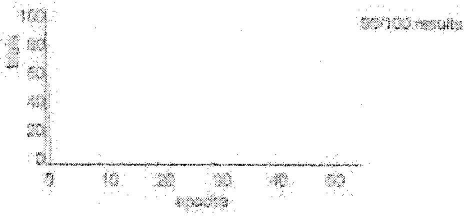

o14clean.jp
fig. 113






o15clean.jp
fig. 114



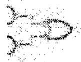


Analysis of Numerous Trials on Each Input Image

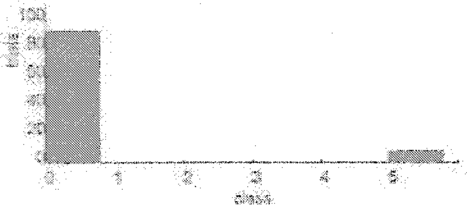
Input Image

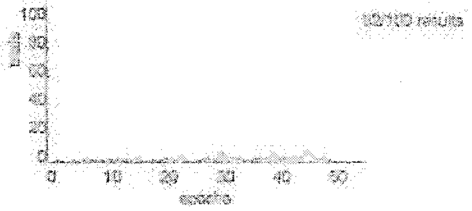
Accuracy Rating

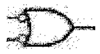
Distribution of Active Evolution



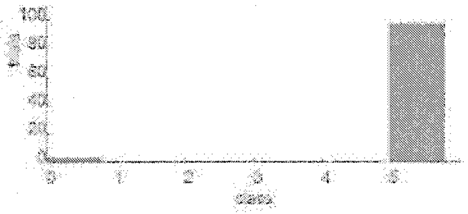
o16clean.jp
fig. 115

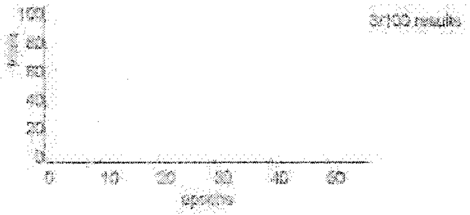


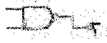




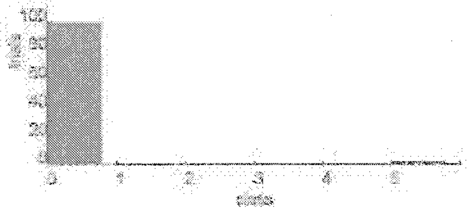
o17clean.jp
fig. 116

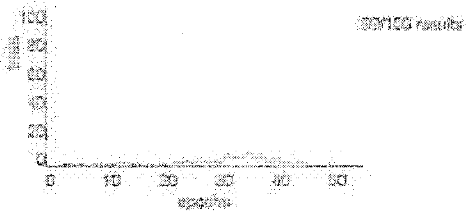


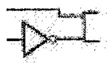




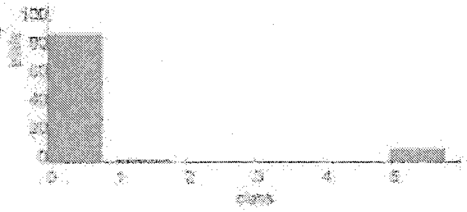
o18clean.jp
fig. 117

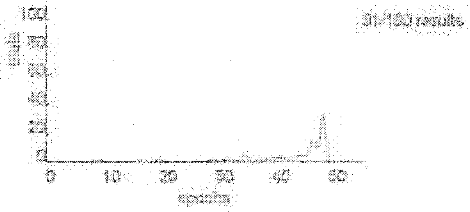


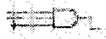




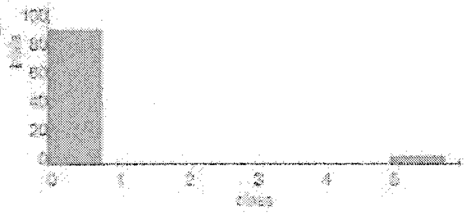
o19clean.jp
fig. 118

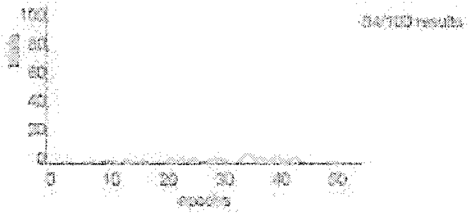






o20clean.jp
fig. 119



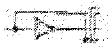


Analysis of Numerous Trials on Each Input Image

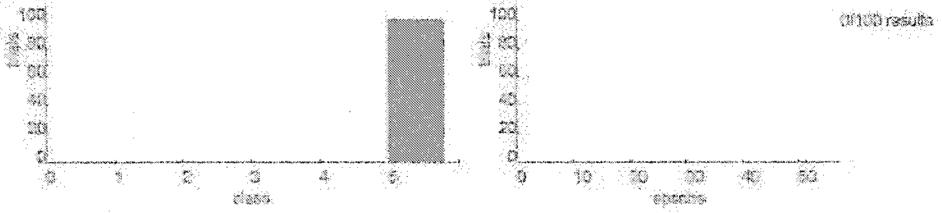
Input Image

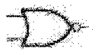
Accuracy Rating

Distribution of Active Evolution

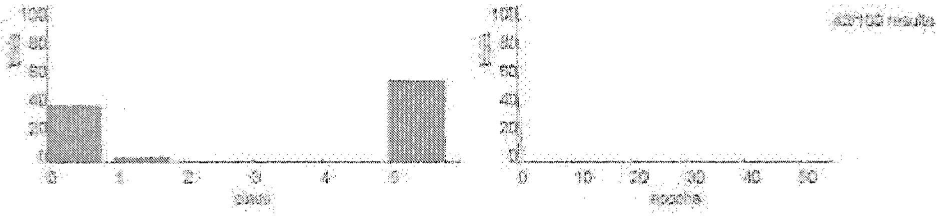


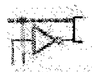
o21clean.jp
fig. 120



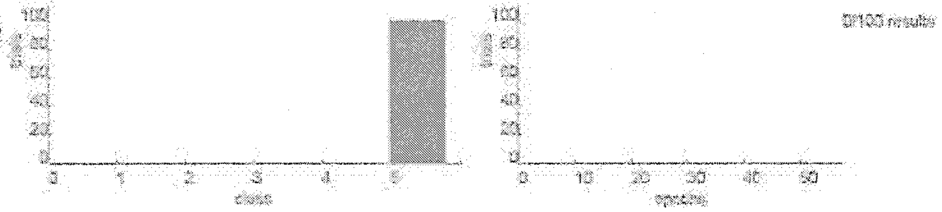


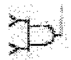
o22clean.jp
fig. 121



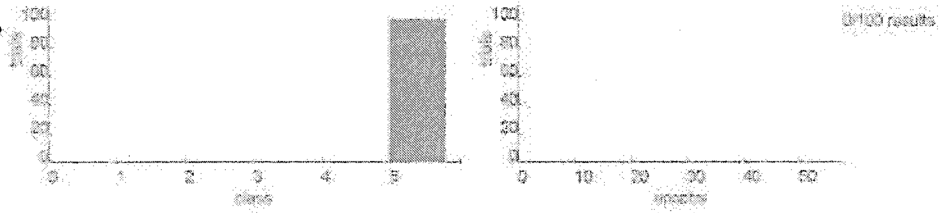


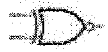
o23clean.jp
fig. 122



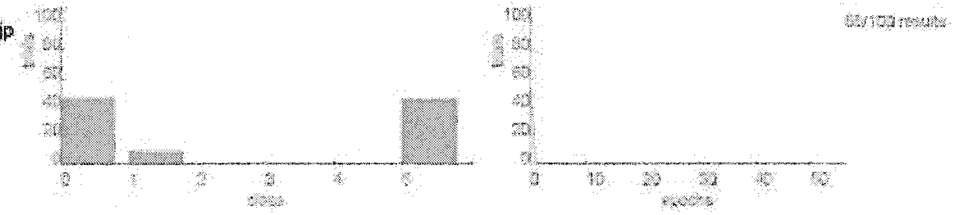


o24clean.jp
fig. 123





o25clean.jp
fig. 124

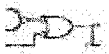


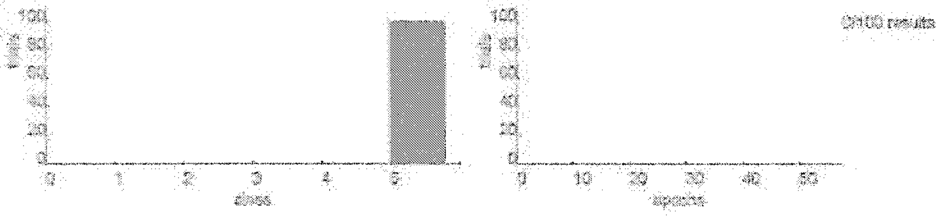
Analysis of Numerous Trials on Each Input Image

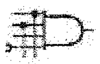
Input Image

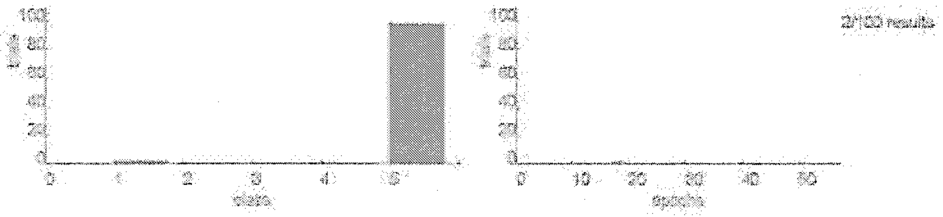
Accuracy Rating

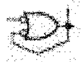
Distribution of Active Evolution

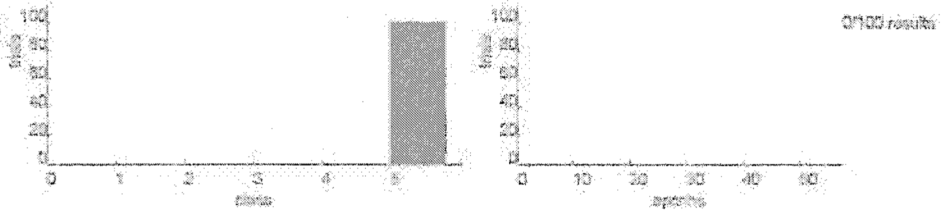

o26clean.jp
fig. 125

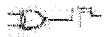


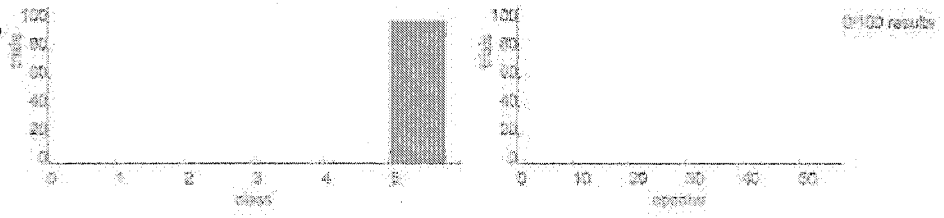

o27clean.jp
fig. 126




o28clean.jp
fig. 127




o29clean.jp
fig. 128

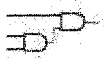


Analysis of Numerous Trials on Each Input Image

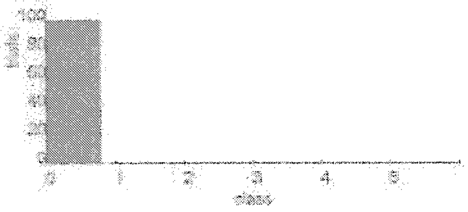
Input Image

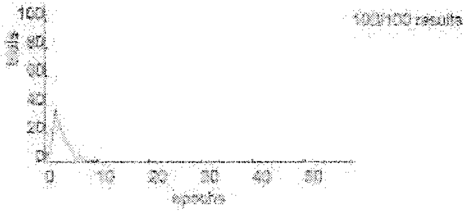
Accuracy Rating

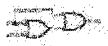
Distribution of Active Evolution



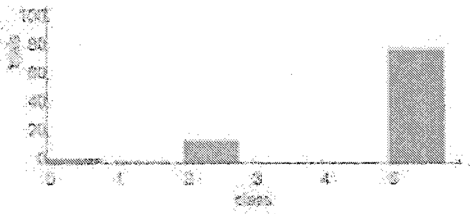
d0clean.jpg
fig. 129

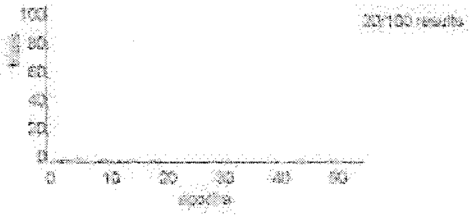


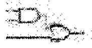




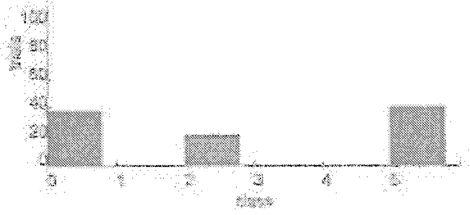
d1clean.jpg
fig. 130

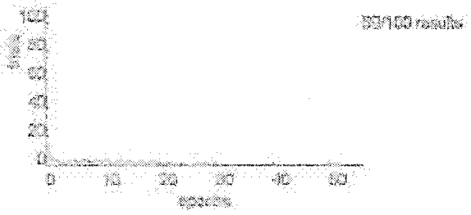


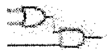




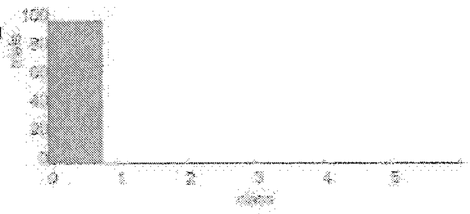
d2clean.jpg
fig. 131

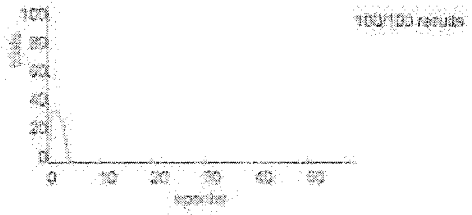


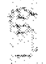




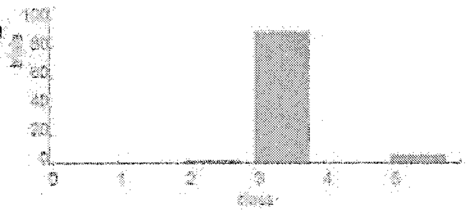
d3clean.jpg
fig. 132

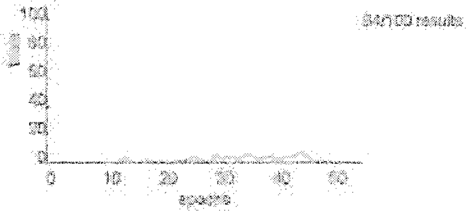






d4clean.jpg
fig. 133



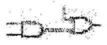


Analysis of Numerous Trials on Each Input Image

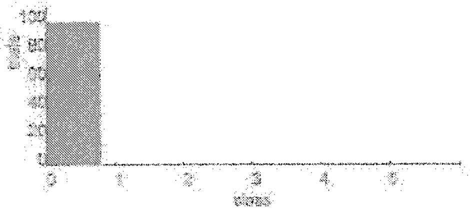
Input Image

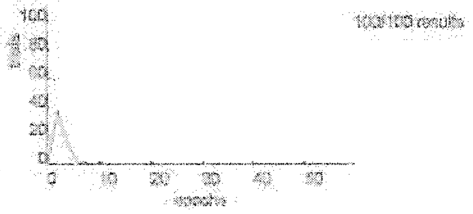
Accuracy Rating

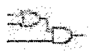
Distribution of Active Evolution



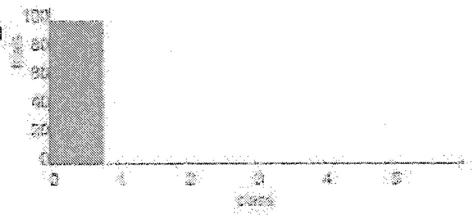
d5clean.jpg
fig. 134

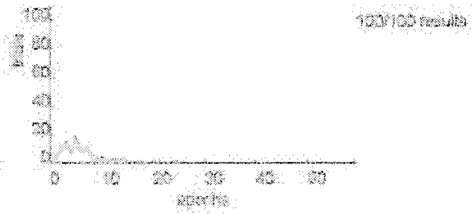


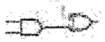




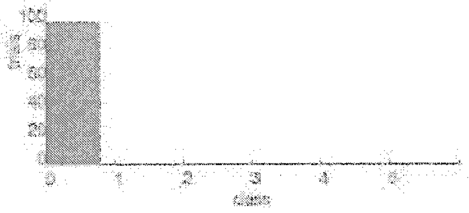
d6clean.jpg
fig. 135

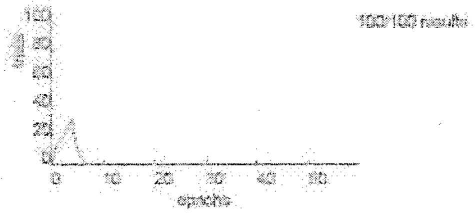


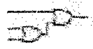




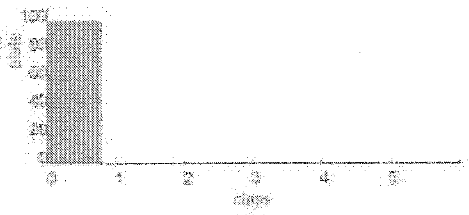
d7clean.jpg
fig. 136

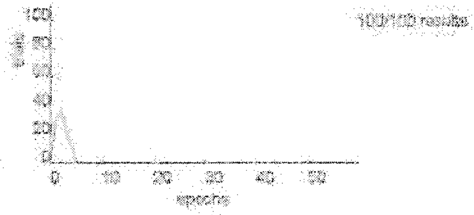


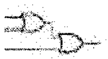




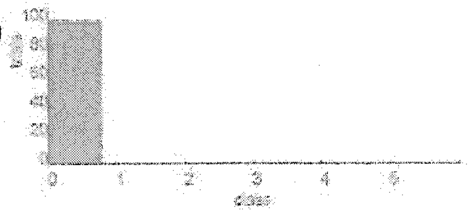
d8clean.jpg
fig. 137

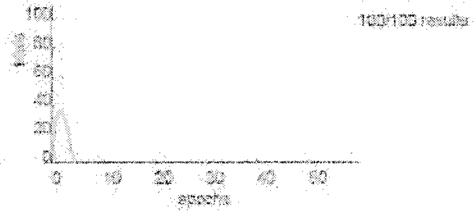






d9clean.jpg
fig. 138



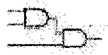


Analysis of Numerous Trials on Each Input Image

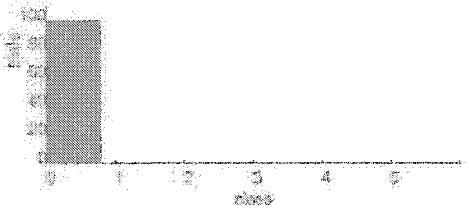
Input Image

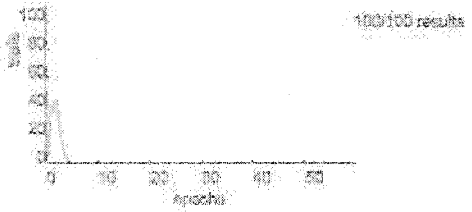
Accuracy Rating

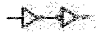
Distribution of Active Evolution



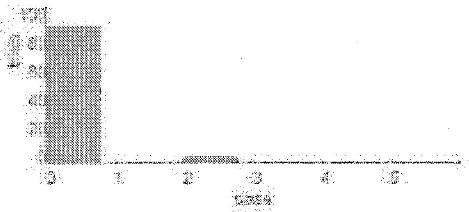
d10clean.jp
fig. 139

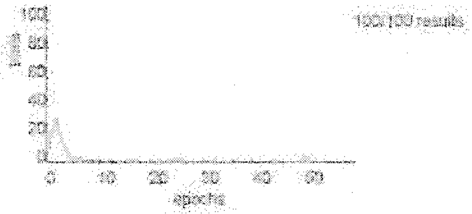


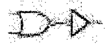




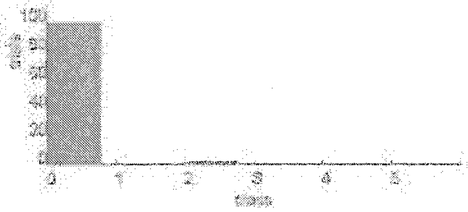
d11clean.jp
fig. 140

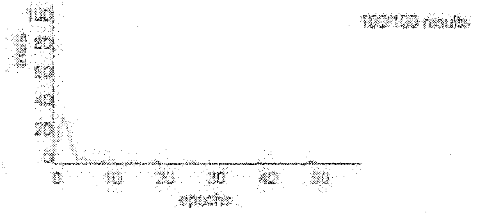


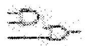




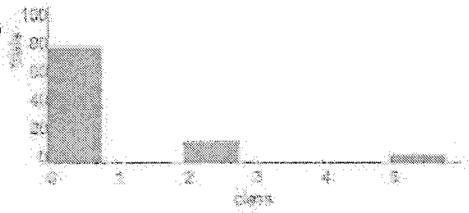
d13clean.jp
fig. 141

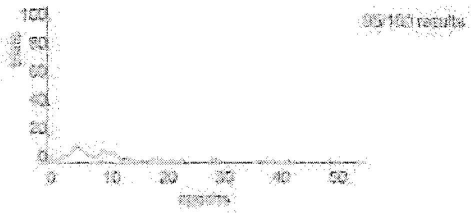


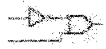




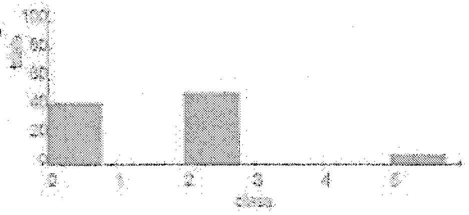
d12clean.jp
fig. 142

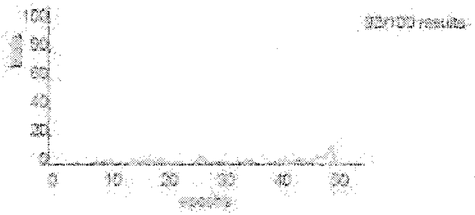






d14clean.jp
fig. 143



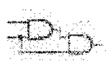


Analysis of Numerous Trials on Each Input Image

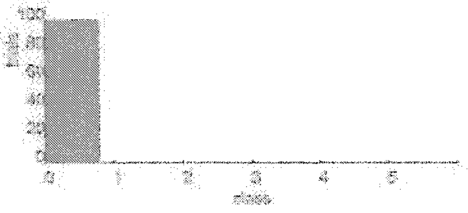
Input Image

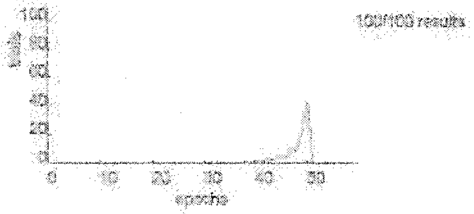
Accuracy Rating

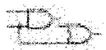
Distribution of Active Evolution



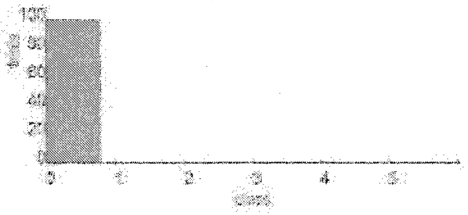
d15clean.jp
fig. 144

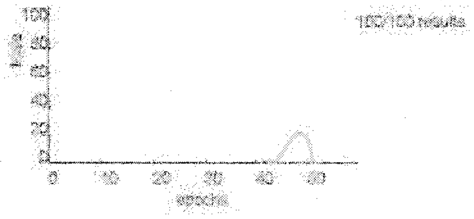


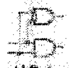




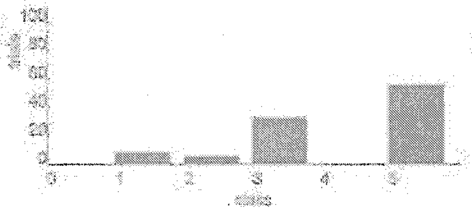
d16clean.jp
fig. 145

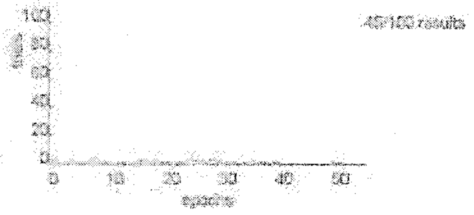




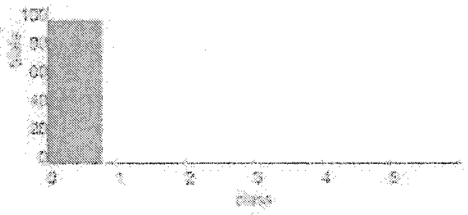


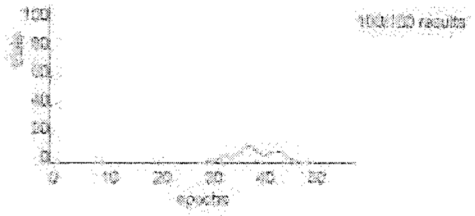
d17clean.jp
fig. 146

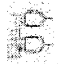




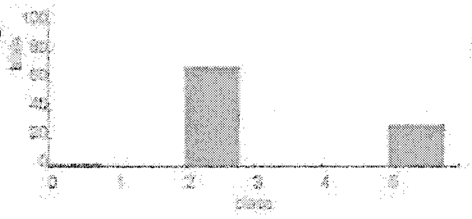
d18clean.jp
fig. 147

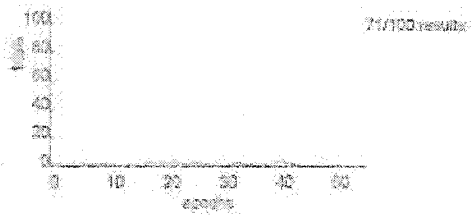






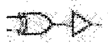
d19clean.jp
fig. 148



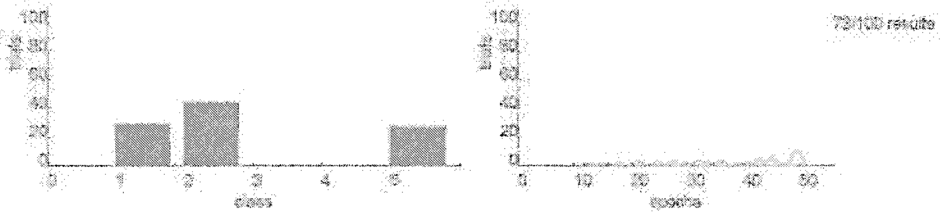


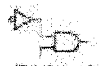
Analysis of Numerous Trials on Each Input Image

Input Image Accuracy Rating Distribution of Active Evolution

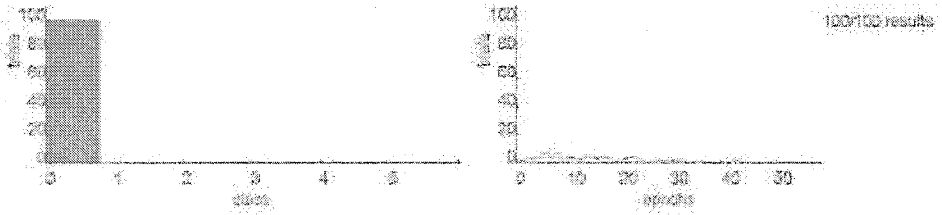


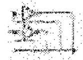
d20clean.jp
fig. 149



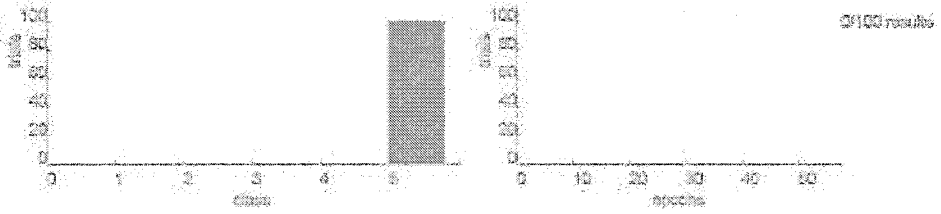


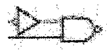
d21clean.jp
fig. 150



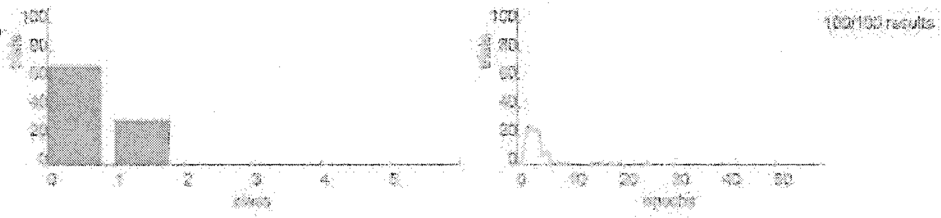


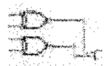
d22clean.jp
fig. 151



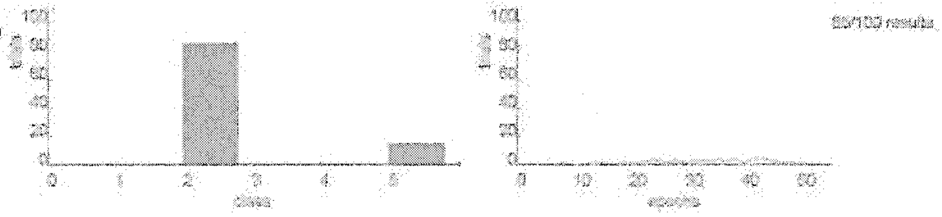


d23clean.jp
fig. 152





d24clean.jp
fig. 153

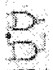


Analysis of Numerous Trials on Each Input Image

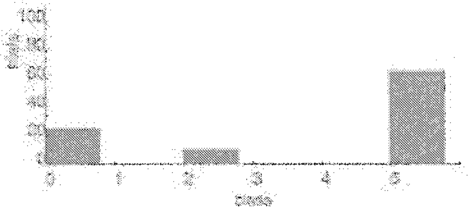
Input Image

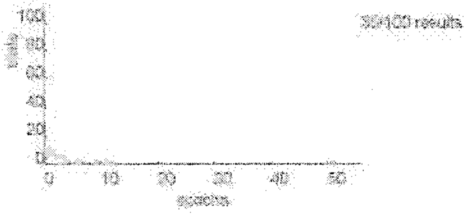
Accuracy Rating

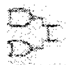
Distribution of Active Evolution



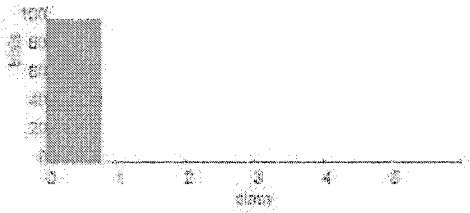
d25clean.jp
fig. 154

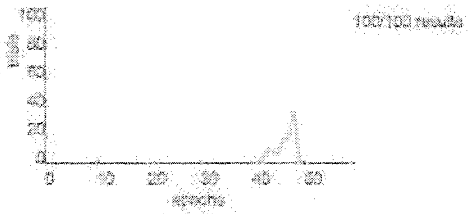


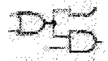




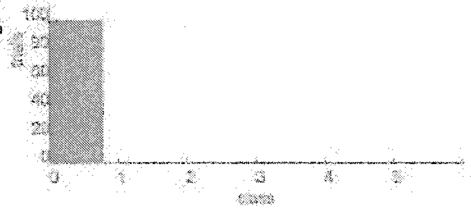
d26clean.jp
fig. 155

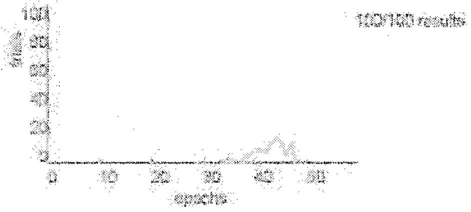


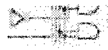




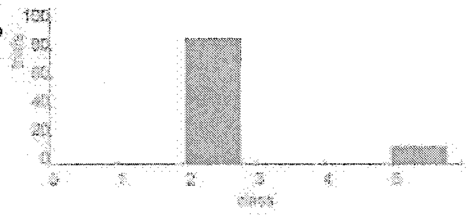
d27clean.jp
fig. 156

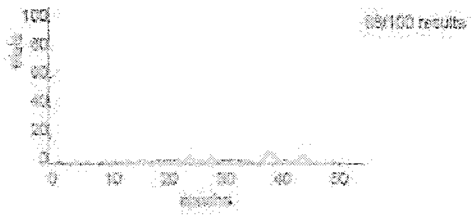






d28clean.jp
fig. 157



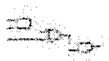


Analysis of Numerous Trials on Each Input Image

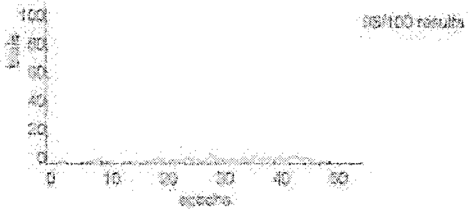
Input Image

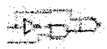
Accuracy Rating

Distribution of Active Evolution

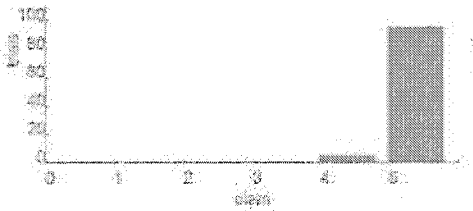


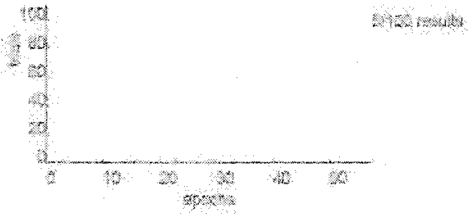
t0clean.jpg
fig. 158

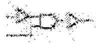




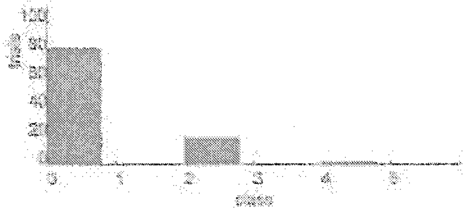
t1clean.jpg
fig. 159

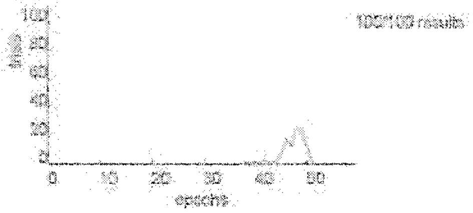


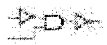




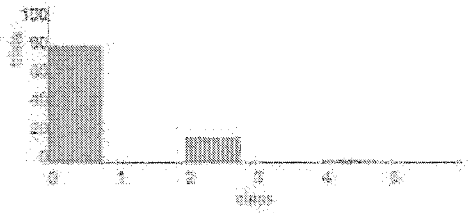
t2clean.jpg
fig. 160

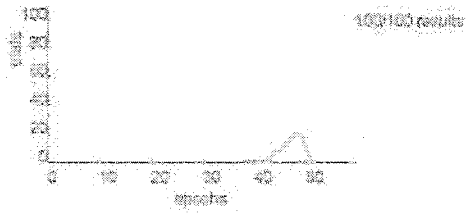


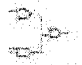




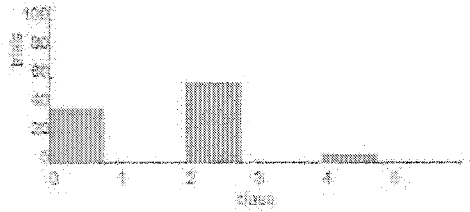
t3clean.jpg
fig. 161

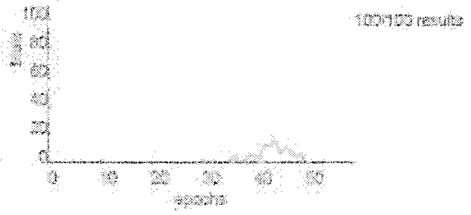






t4clean.jpg
fig. 162



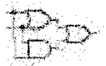


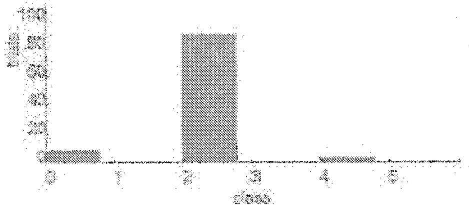
Analysis of Numerous Trials on Each Input Image

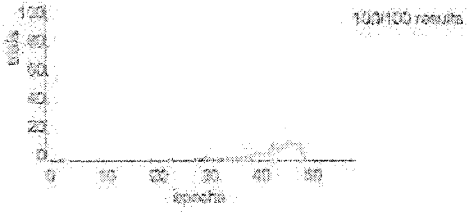
Input Image

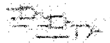
Accuracy Rating

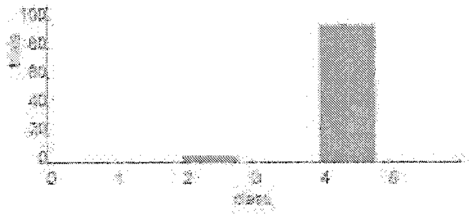
Distribution of Active Evolution

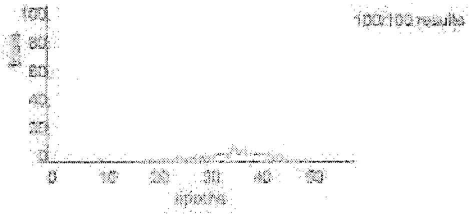

t5clean.jpg
fig. 163

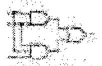


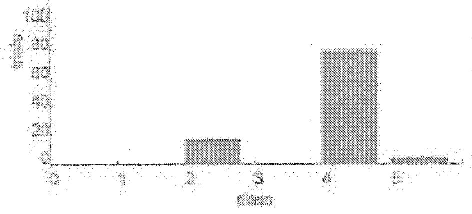


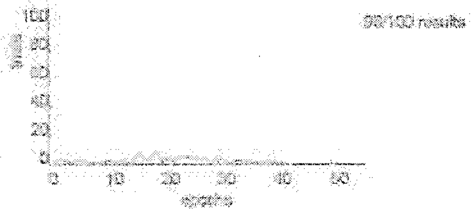

t6clean.jpg
fig. 164

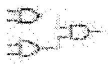


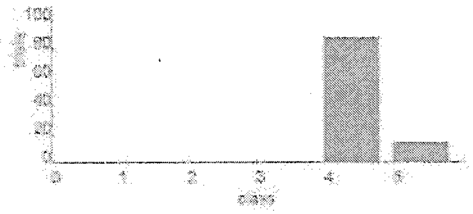


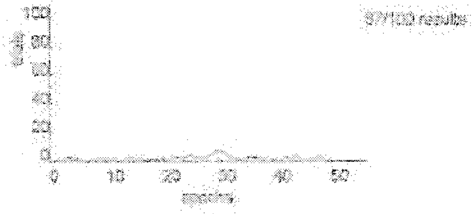

t7clean.jpg
fig. 165

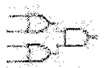


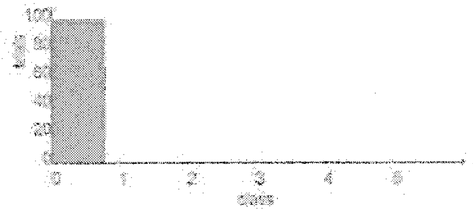


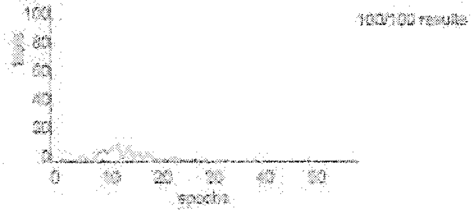

t8clean.jpg
fig. 166






t9clean.jpg
fig. 167



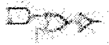


Analysis of Numerous Trials on Each Input Image

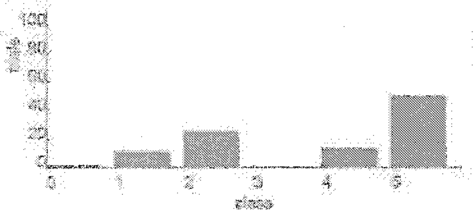
Input Image

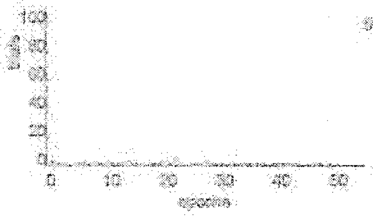
Accuracy Rating

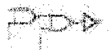
Distribution of Active Evolution



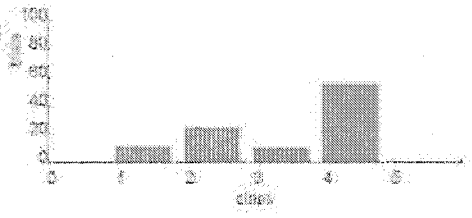
t10clean.jpg
fig. 168

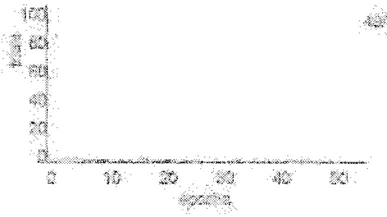


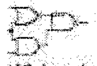




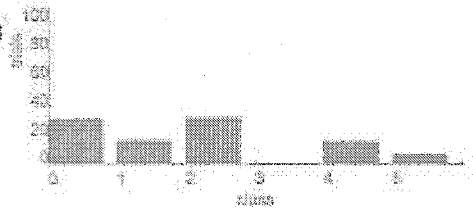
t11clean.jpg
fig. 169

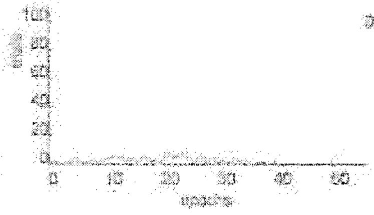


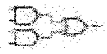




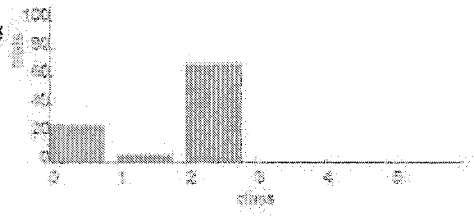
t12clean.jpg
fig. 170

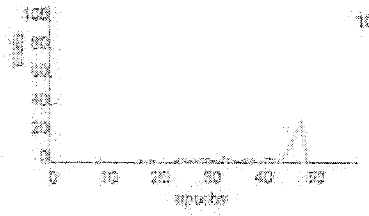


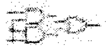




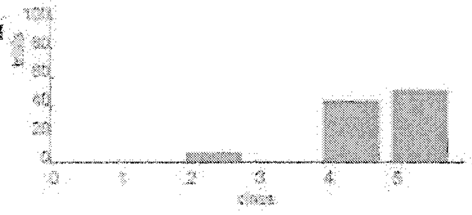
t13clean.jpg
fig. 171

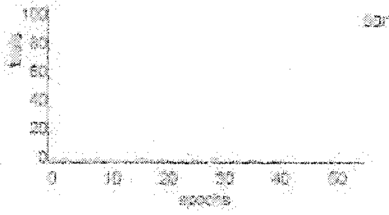






t14clean.jpg
fig. 172



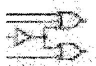


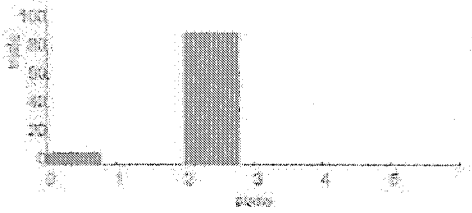
Analysis of Numerous Trials on Each Input Image

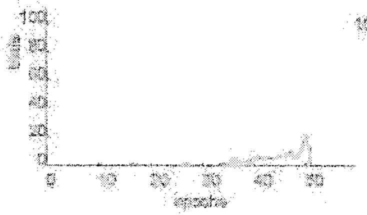
Input Image

Accuracy Rating

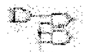
Distribution of Active Evolution

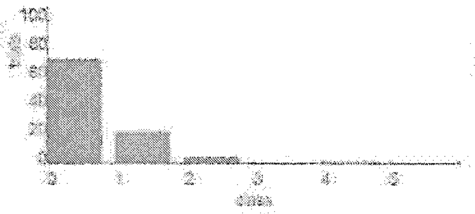

t15clean.jpg
fig. 173

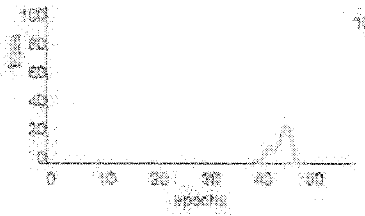




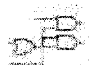
100/100 results

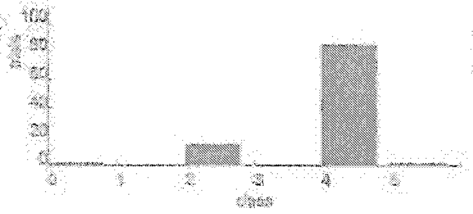

t16clean.jpg
fig. 174

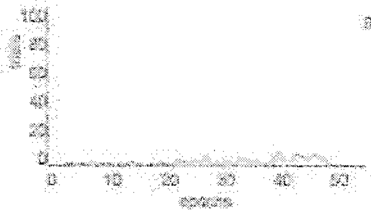




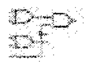
100/100 results

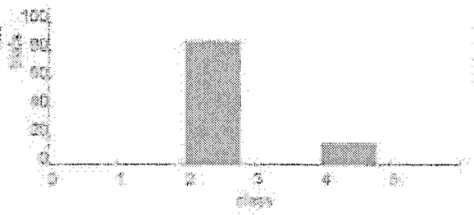

t17clean.jpg
fig. 175

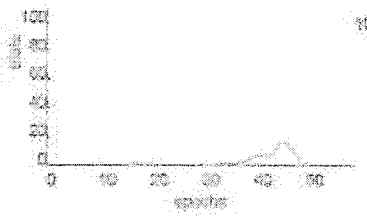




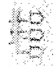
99/100 results

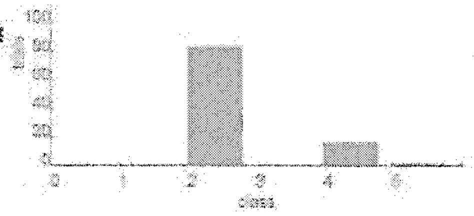

t18clean.jpg
fig. 176

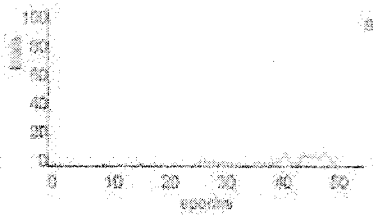




100/100 results


t19clean.jpg
fig. 177





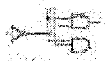
99/100 results

Analysis of Numerous Trials on Each Input Image

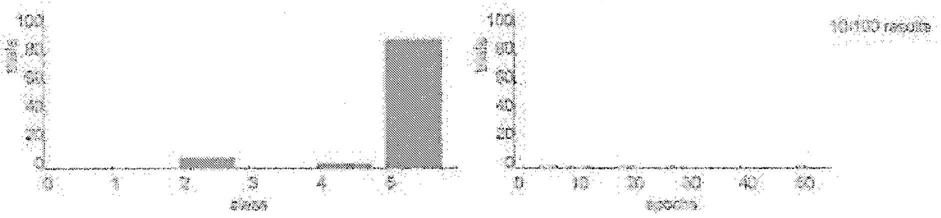
Input Image

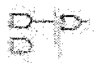
Accuracy Rating

Distribution of Active Evolution

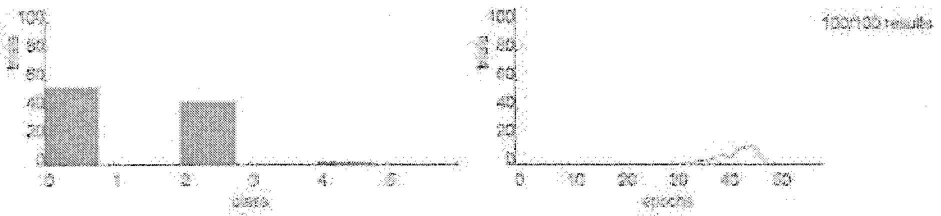


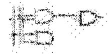
t20clean.jp
fig. 178



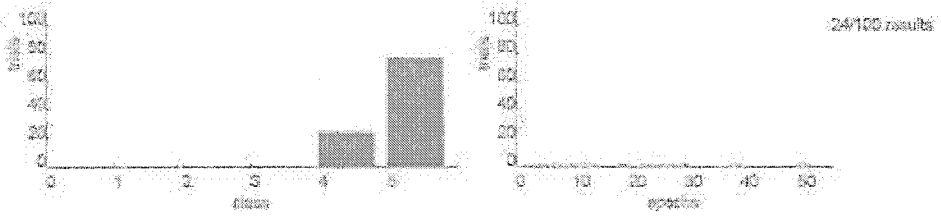


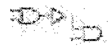
t21clean.jp
fig. 179



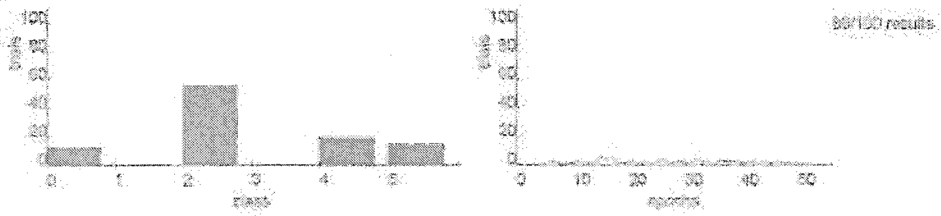


t22clean.jp
fig. 180

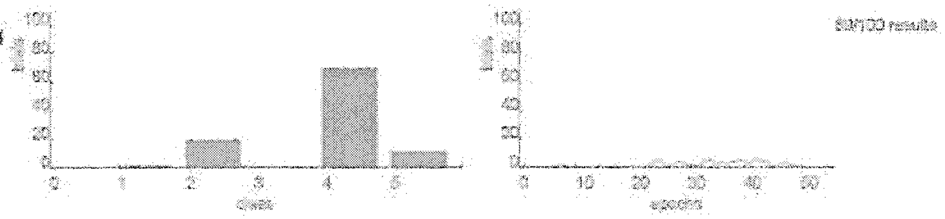




t23clean.jp
fig. 181



t24clean.jp
fig. 182



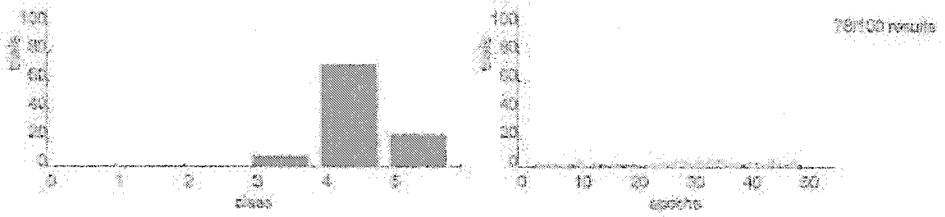
Analysis of Numerous Trials on Each Input Image

Input Image

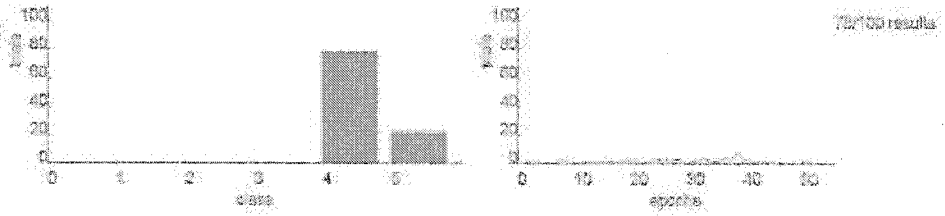
Accuracy Rating

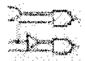
Distribution of Active Evolution

t25clean.jpg
fig. 183

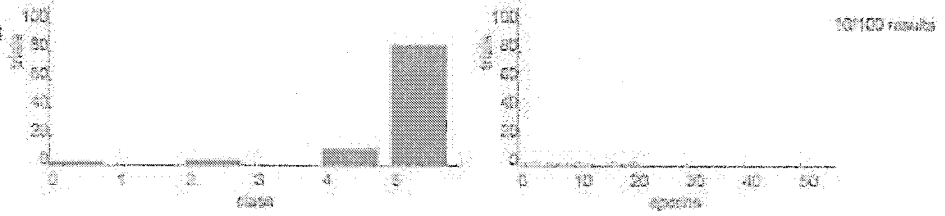


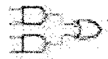
t26clean.jpg
fig. 184



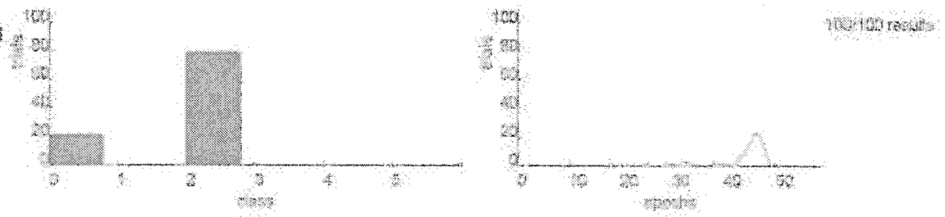


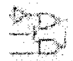
t27clean.jpg
fig. 185



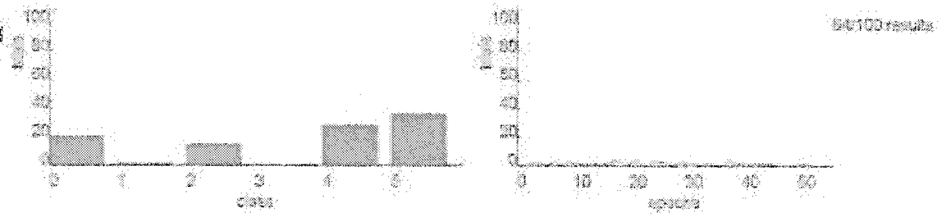


t28clean.jpg
fig. 186





t29clean.jpg
fig. 187



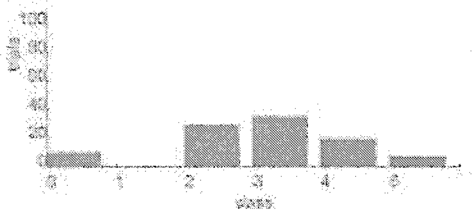
Analysis of Numerous Trials on Each Input Image

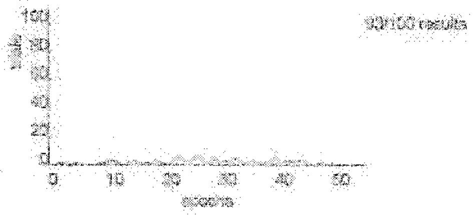
Input Image

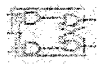
Accuracy Rating

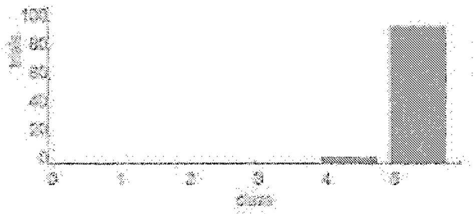
Distribution of Active Evolution

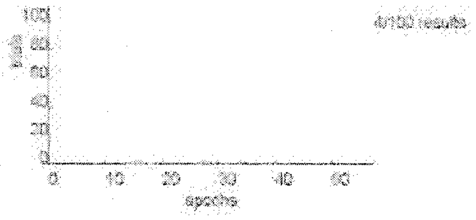
f0clean.jpg
fig. 188



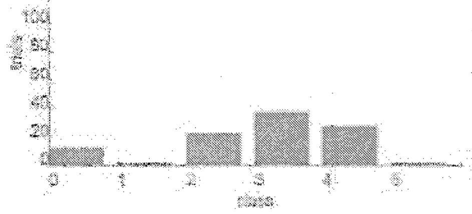


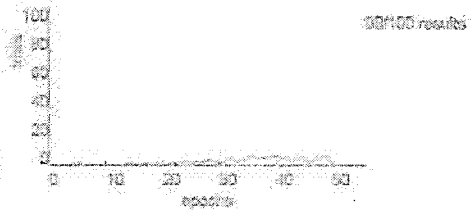

f1clean.jpg
fig. 189

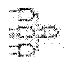


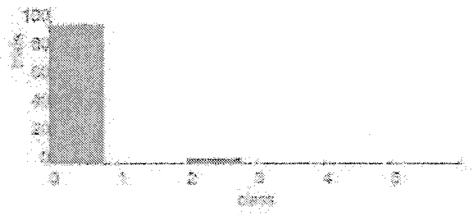


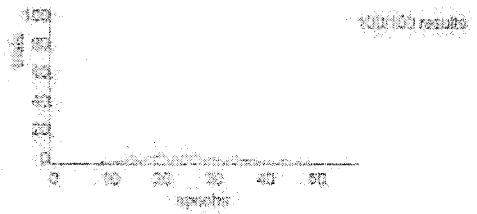
f2clean.jpg
fig. 190



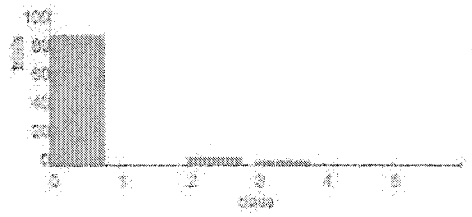


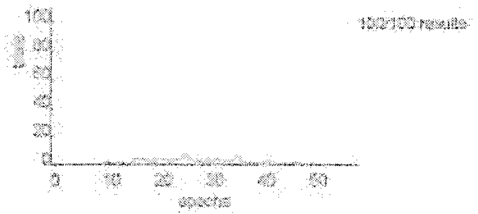

f3clean.jpg
fig. 191





f4clean.jpg
fig. 192



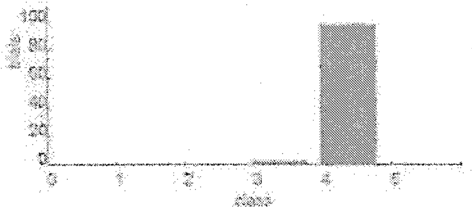


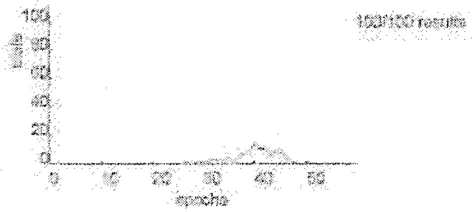
Analysis of Numerous Trials on Each Input Image

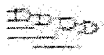
Input Image

Accuracy Rating

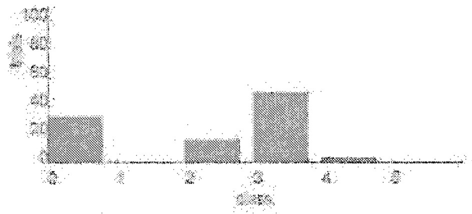
Distribution of Active Evolution

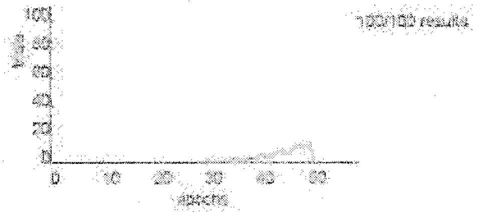


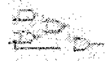




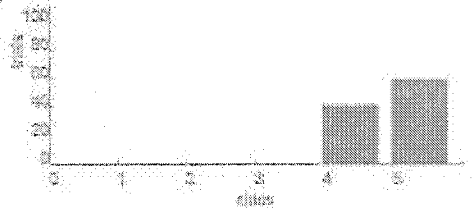
f6clean.jpg
fig. 194

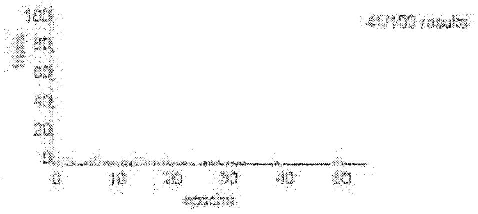


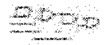




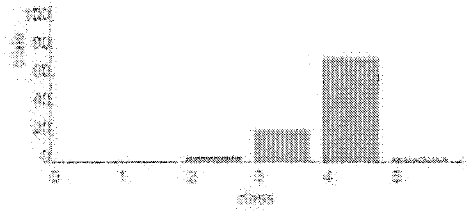
f7clean.jpg
fig. 195

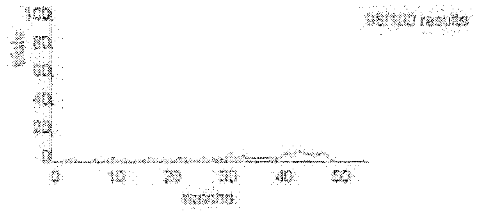


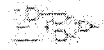




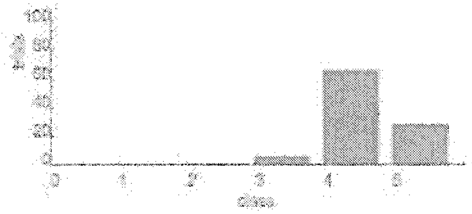
f8clean.jpg
fig. 196

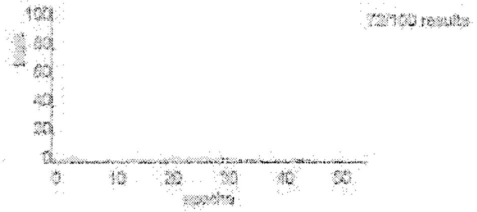






f9clean.jpg
fig. 197



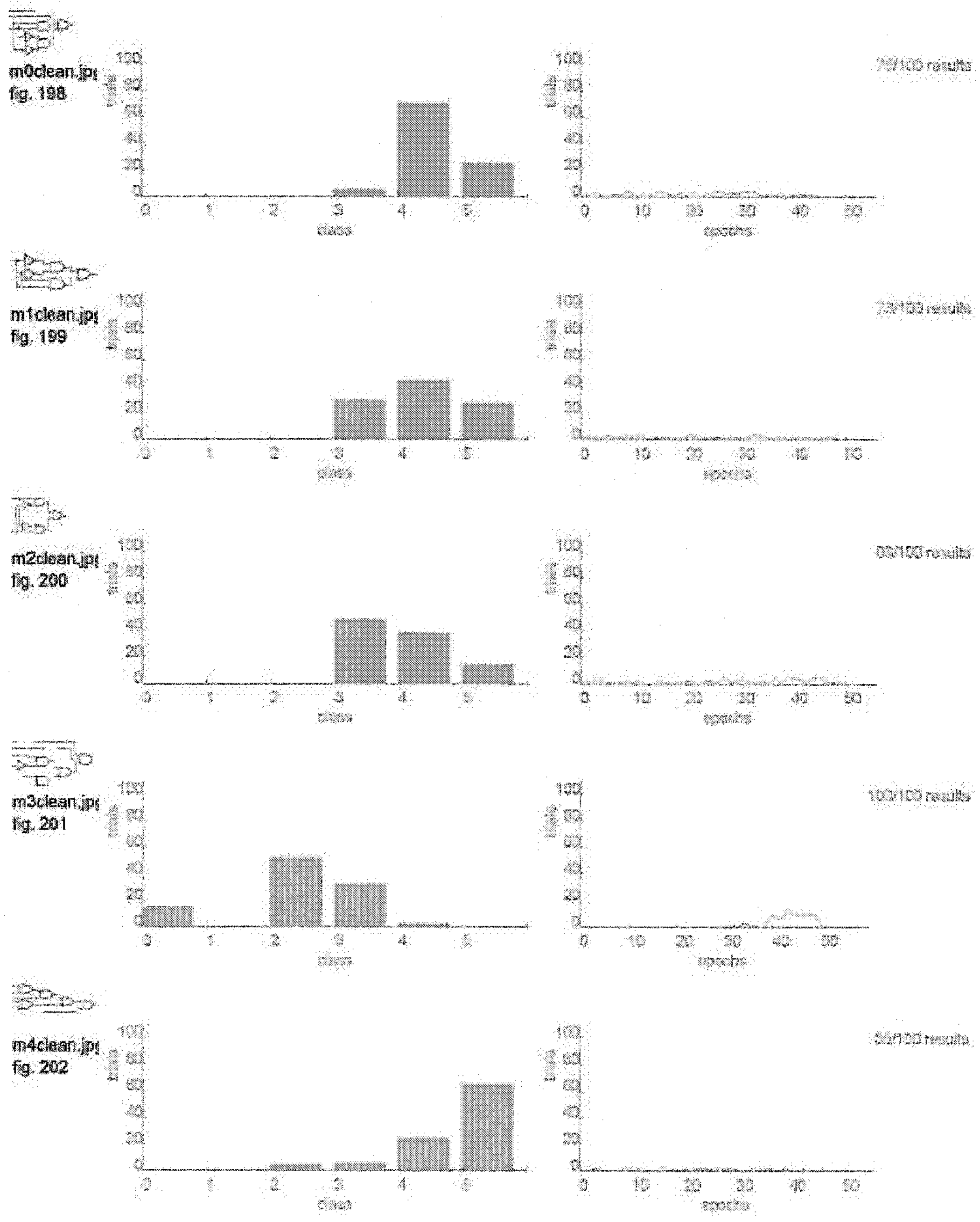


Analysis of Numerous Trials on Each Input Image

Input Image

Accuracy Rating

Distribution of Active Evolution

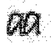


Analysis of Numerous Trials on Each Input Image

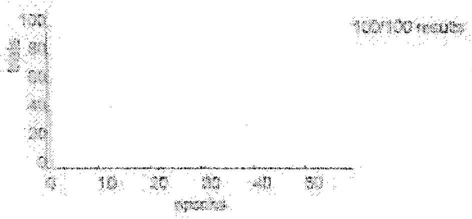
Input Image

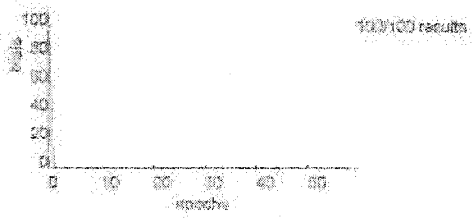
Distribution of First Symbol Found

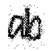
Distribution of Termination



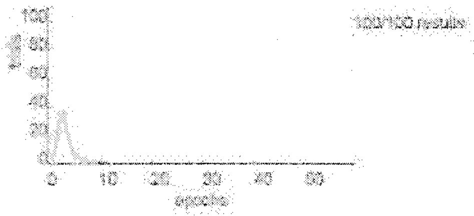
aa.jpg
fig. 203

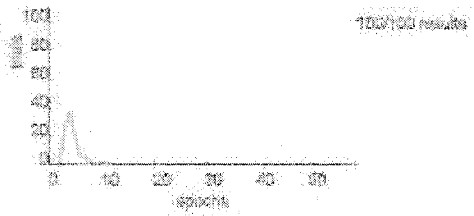




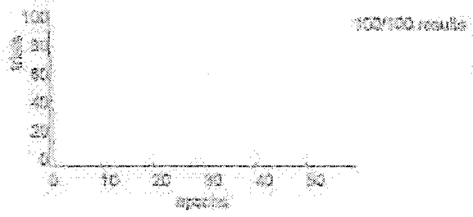


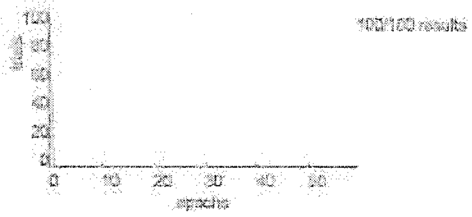
ab.jpg
fig. 204

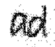




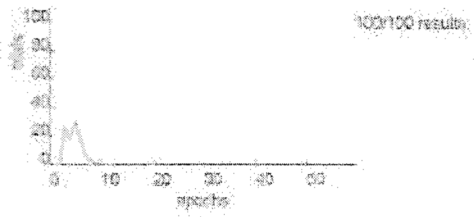
ac.jpg
fig. 205

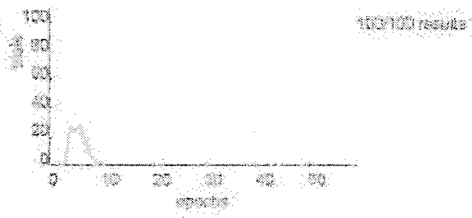


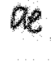




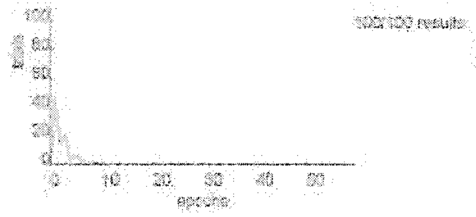
ad.jpg
fig. 206

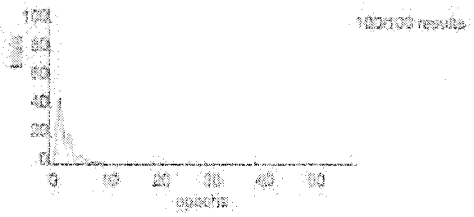




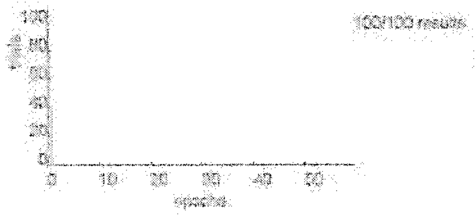
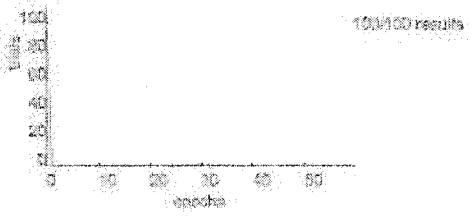
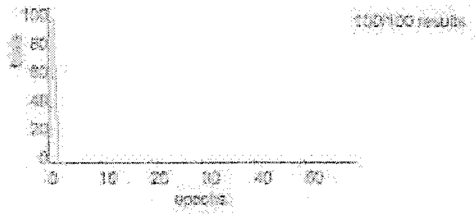
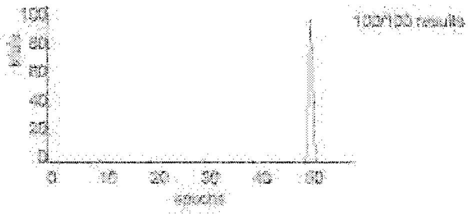
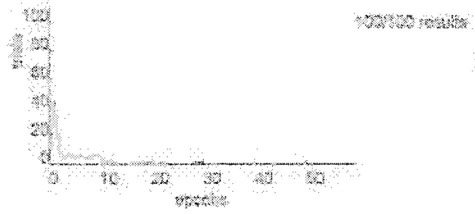
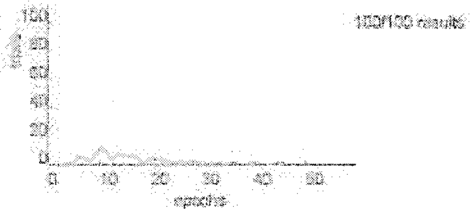
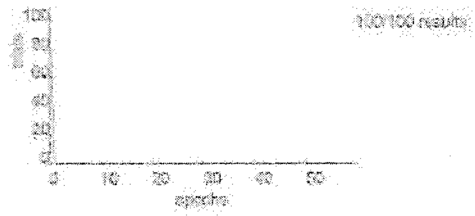
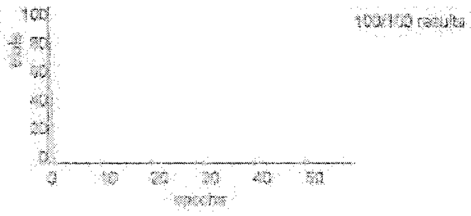
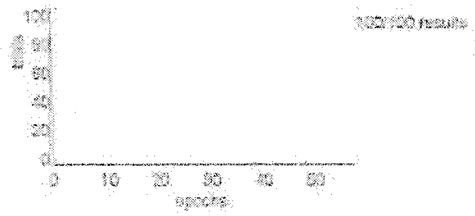
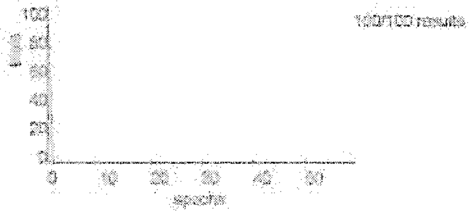


ae.jpg
fig. 207





Analysis of Numerous Trials on Each Input Image

Input Image	Distribution of First Symbol Found	Distribution of Termination
ba.jpg fig. 208		
bb.jpg fig. 209		
bc.jpg fig. 210		
bd.jpg fig. 211		
be.jpg fig. 212		

Analysis of Numerous Trials on Each Input Image

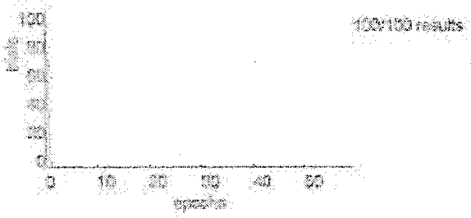
Input Image

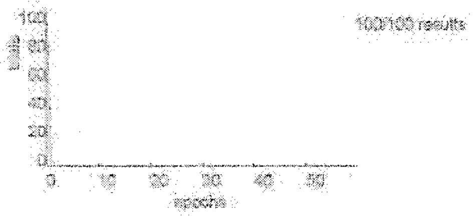
Distribution of First Symbol Found

Distribution of Termination

ca

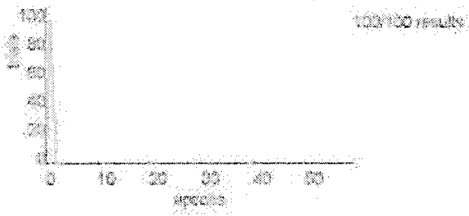
ca.jpg
fig. 213

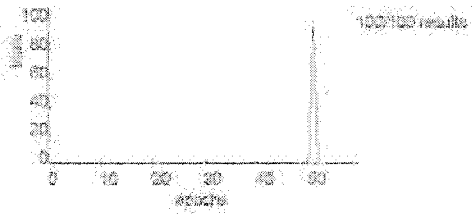




cb

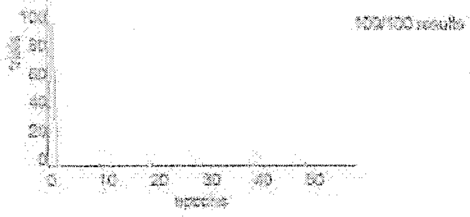
cb.jpg
fig. 214

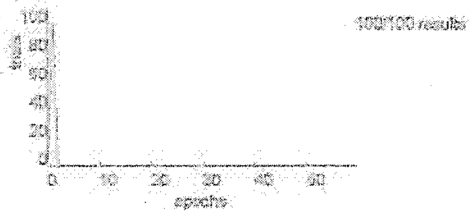




cc

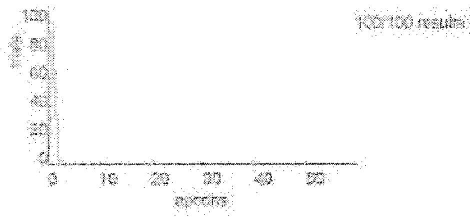
cc.jpg
fig. 215

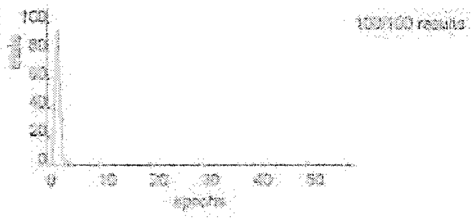




cd

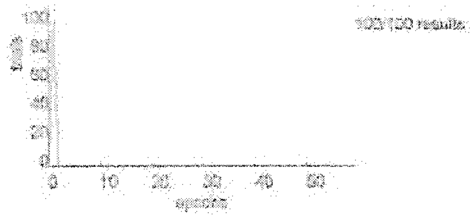
cd.jpg
fig. 216

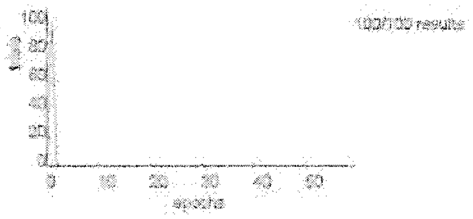




ce

ce.jpg
fig. 217



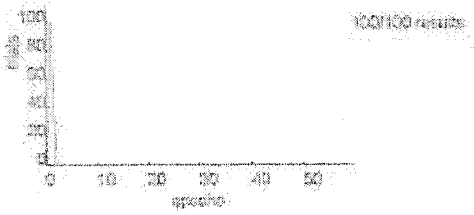


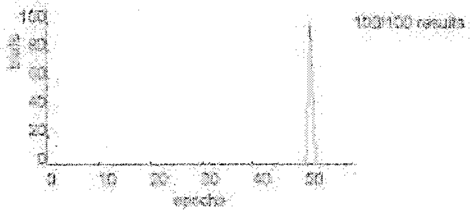
Analysis of Numerous Trials on Each Input Image

Input Image	Distribution of First Symbol Found	Distribution of Termination

da

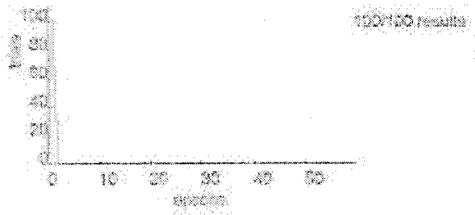
da.jpg
fig. 218

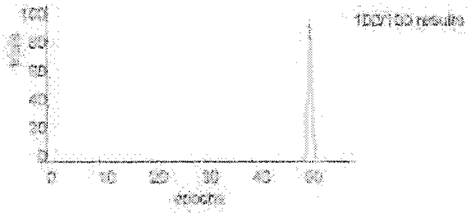




db

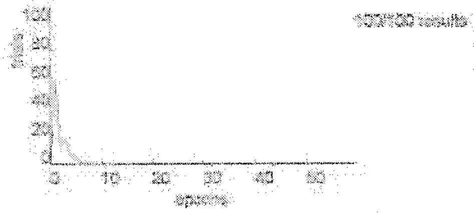
db.jpg
fig. 219

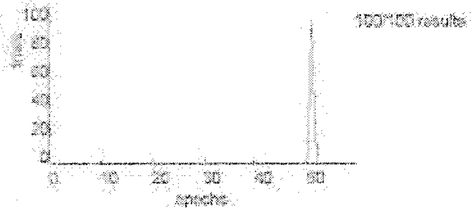




dc

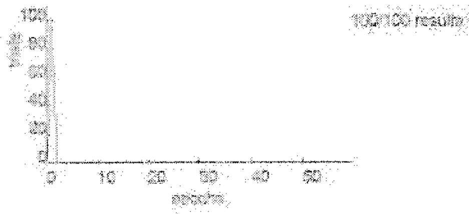
dc.jpg
fig. 220

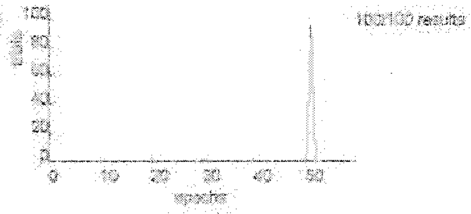




dd

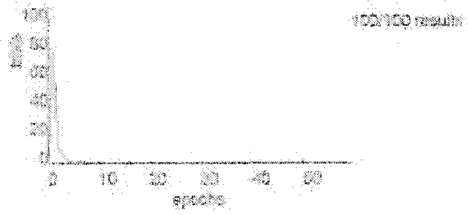
dd.jpg
fig. 221

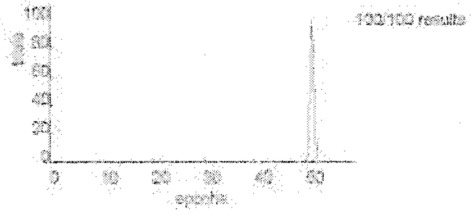




de

de.jpg
fig. 222

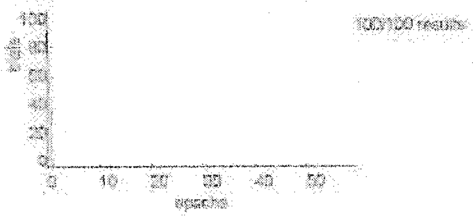


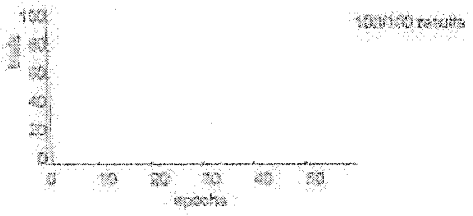


Analysis of Numerous Trials on Each Input Image

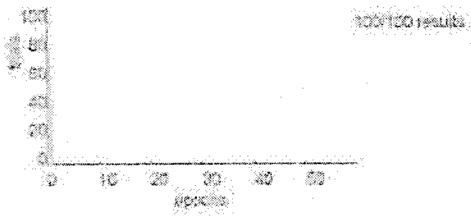
Input Image	Distribution of First Symbol Found	Distribution of Termination

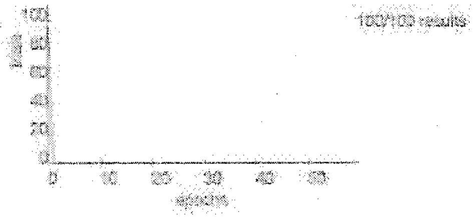
ea.jpg
fig. 223



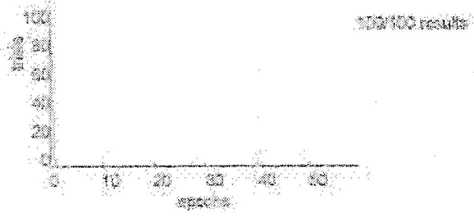


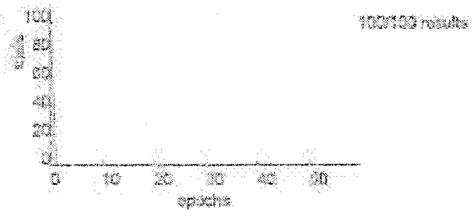
eb.jpg
fig. 224



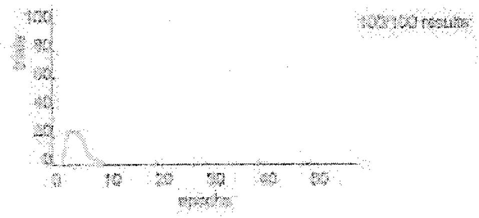


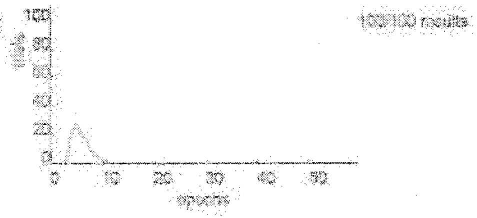
ec.jpg
fig. 225



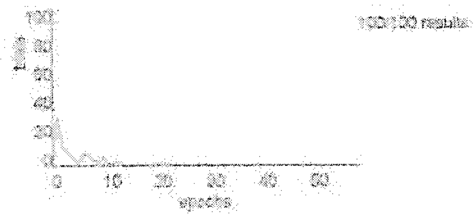


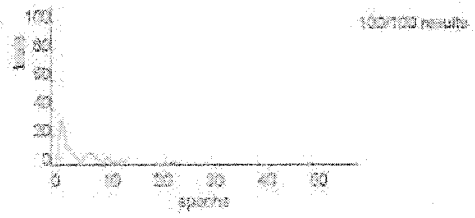
ed.jpg
fig. 226





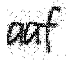
ee.jpg
fig. 227

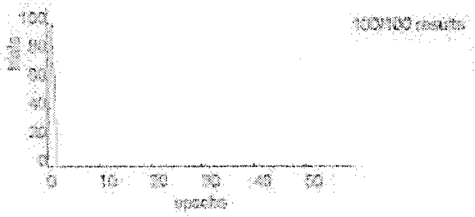


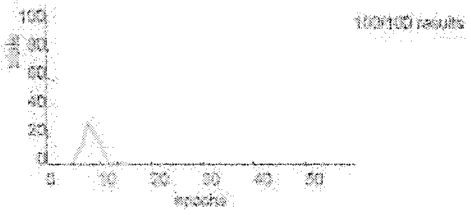


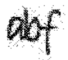
Analysis of Numerous Trials on Each Input Image

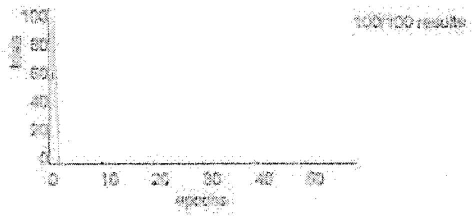
Input Image Distribution of First Symbol Found Distribution of Termination

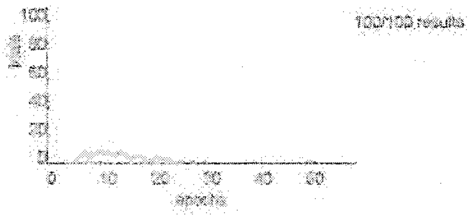

aaf.jpg
fig. 228

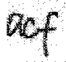


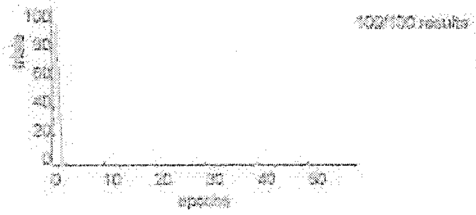


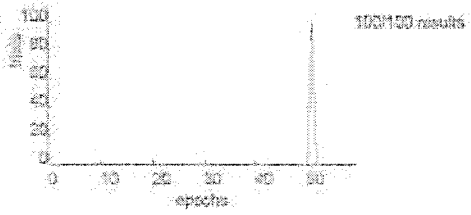

abf.jpg
fig. 229

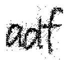


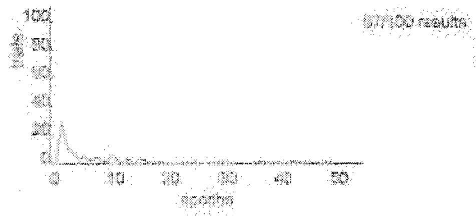


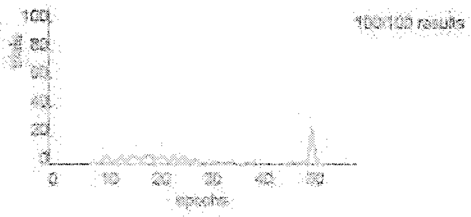

acf.jpg
fig. 230

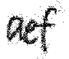


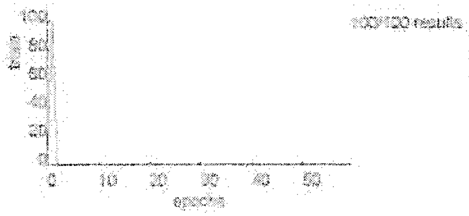


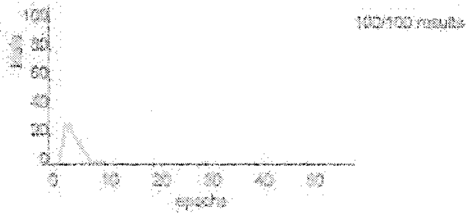

adf.jpg
fig. 231





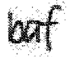

aef.jpg
fig. 232



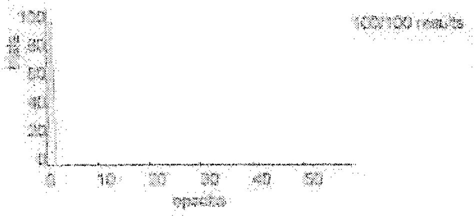


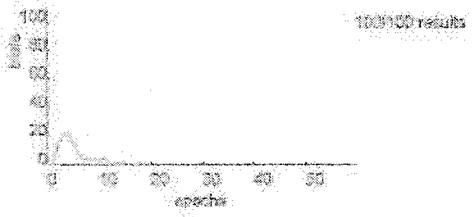
Analysis of Numerous Trials on Each Input Image

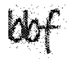
Input Image **Distribution of First Symbol Found** **Distribution of Termination**



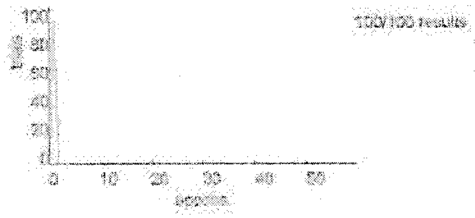
baf.jpg
fig. 233

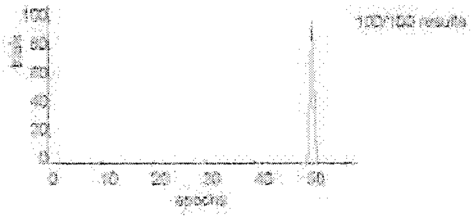


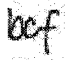




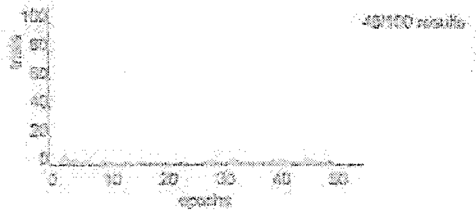
bbf.jpg
fig. 234

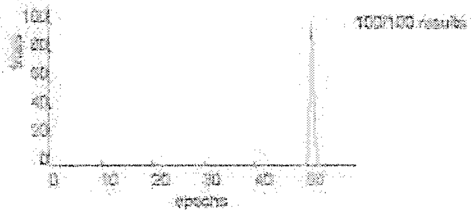


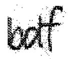




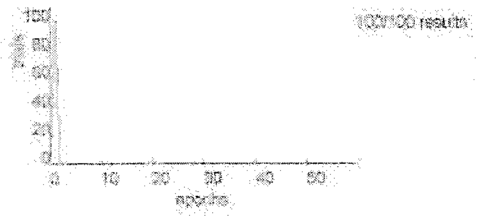
bcf.jpg
fig. 235

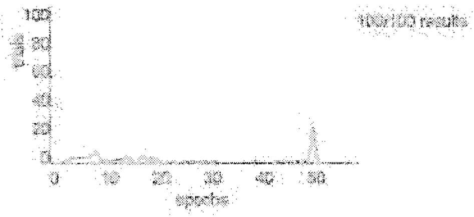


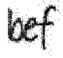




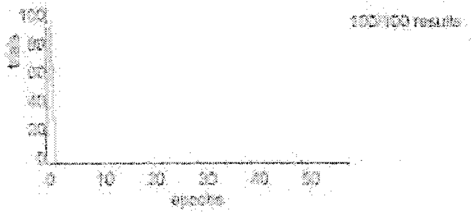
bdf.jpg
fig. 236

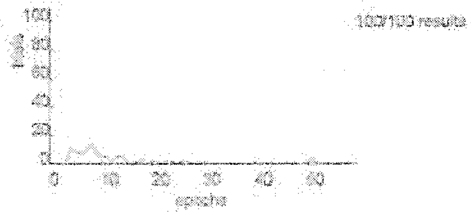






bef.jpg
fig. 237



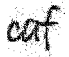


Analysis of Numerous Trials on Each Input Image

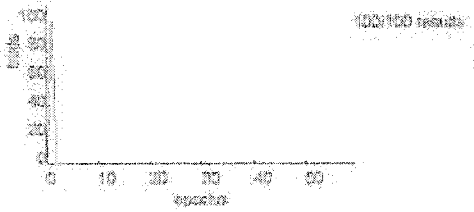
Input Image

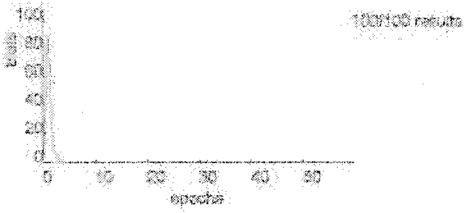
Distribution of First Symbol Found

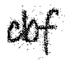
Distribution of Termination



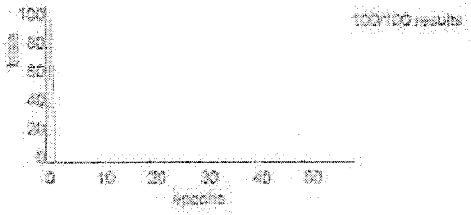
caf.jpg
fig. 238

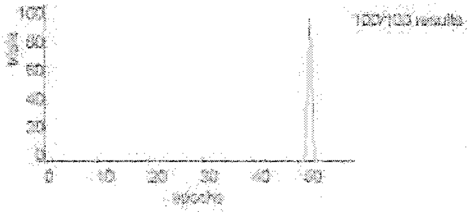


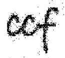




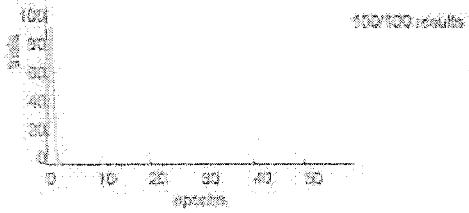
cbf.jpg
fig. 239

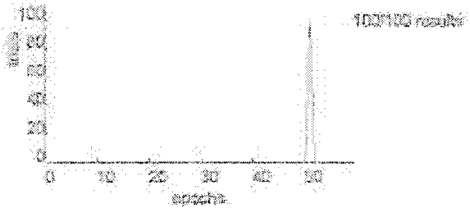


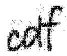




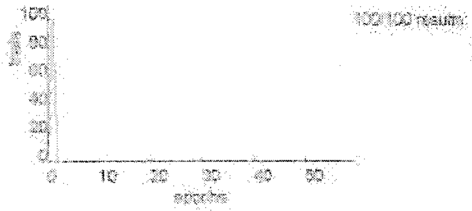
ccf.jpg
fig. 240

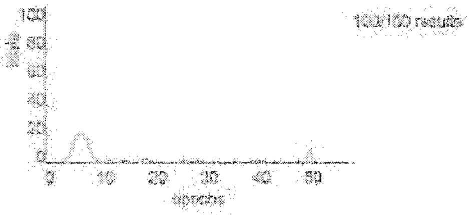


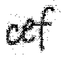




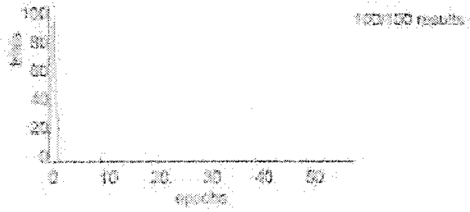
cdf.jpg
fig. 241

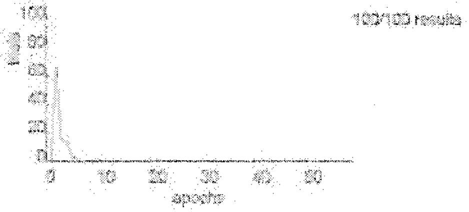






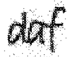
cef.jpg
fig. 242

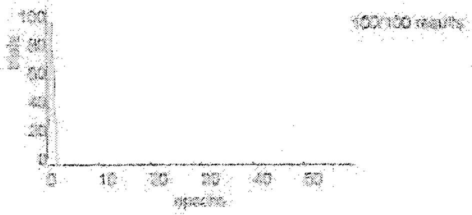


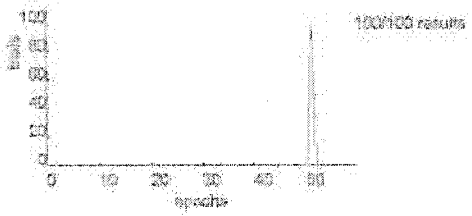


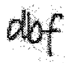
Analysis of Numerous Trials on Each Input Image

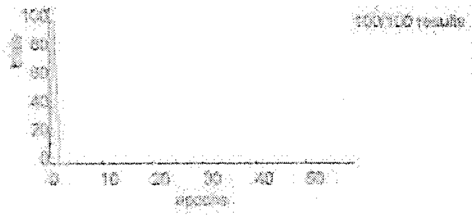
Input Image **Distribution of First Symbol Found** **Distribution of Termination**

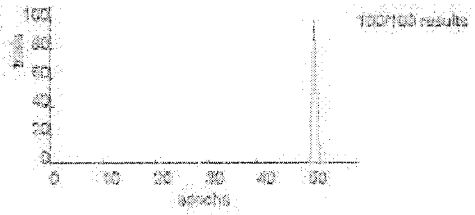

daf.jpg
fig. 243

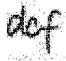


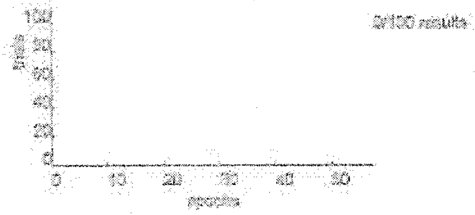


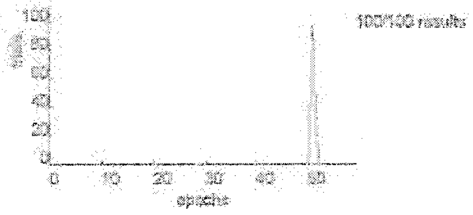

dbf.jpg
fig. 244

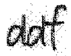


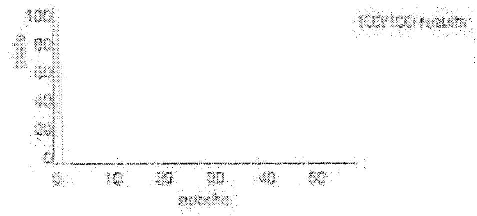


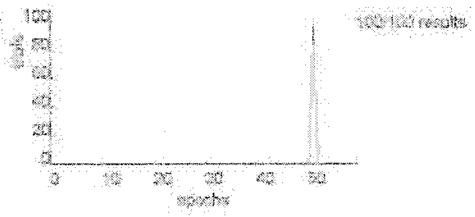

dcf.jpg
fig. 245

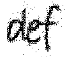


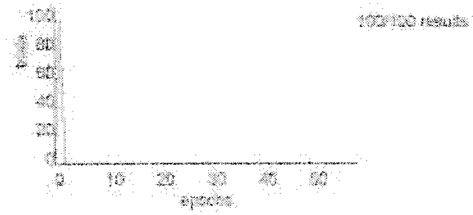


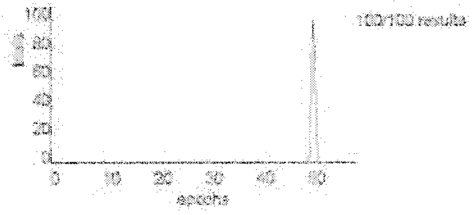

ddf.jpg
fig. 246






def.jpg
fig. 247



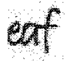


Analysis of Numerous Trials on Each Input Image

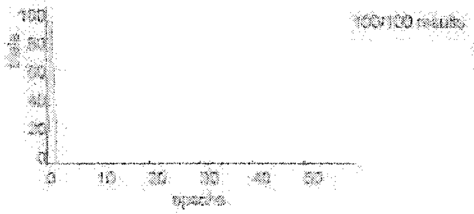
Input Image

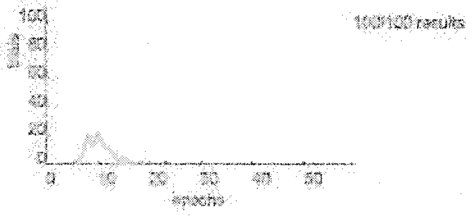
Distribution of First Symbol Found

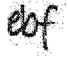
Distribution of Termination



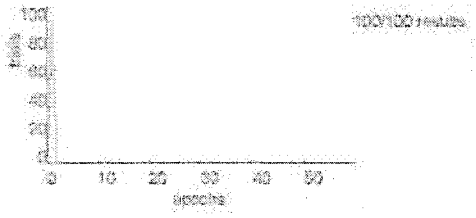
eaf.jpg
fig. 248

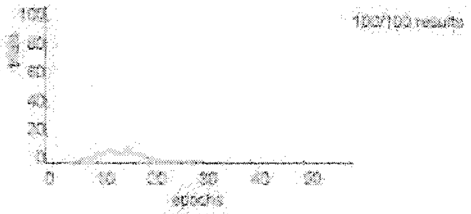


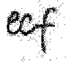




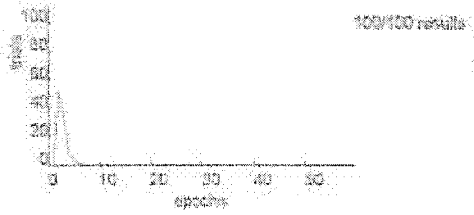
ebf.jpg
fig. 249

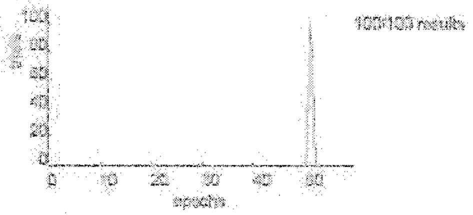


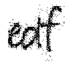




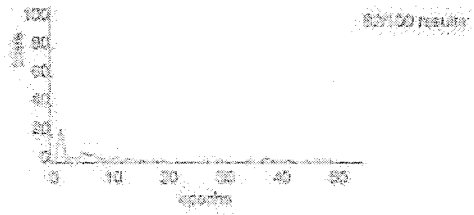
ecf.jpg
fig. 250

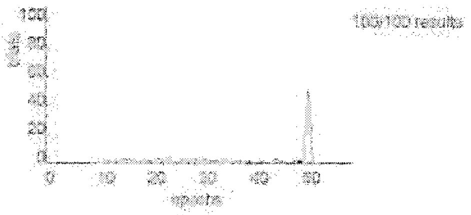


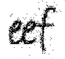




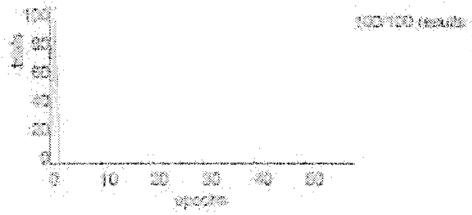
edf.jpg
fig. 251

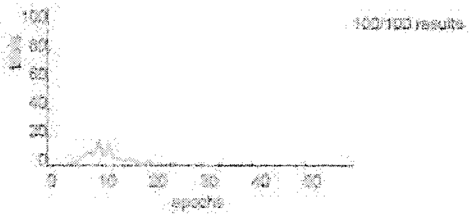






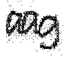
eef.jpg
fig. 252



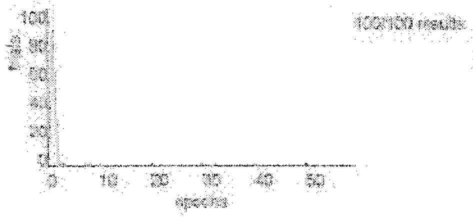


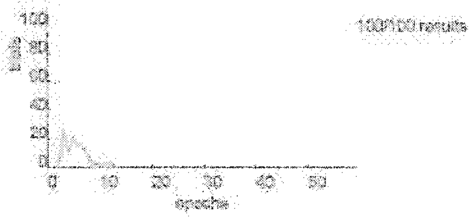
Analysis of Numerous Trials on Each Input Image

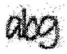
Input Image	Distribution of First Symbol Found	Distribution of Termination



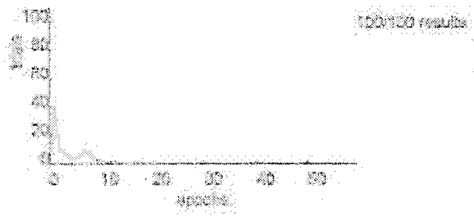
aag.jpg
fig. 253

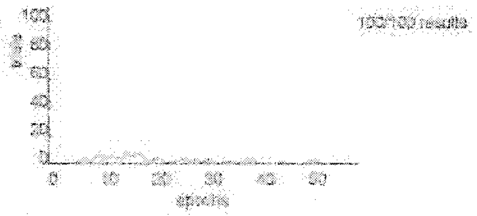


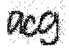




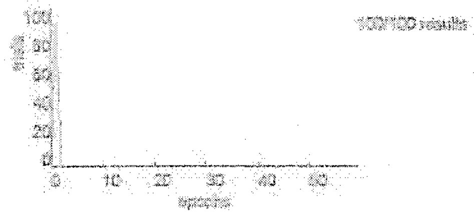
abg.jpg
fig. 254

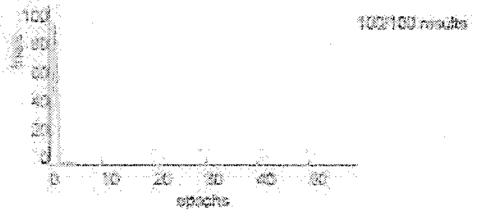


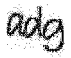




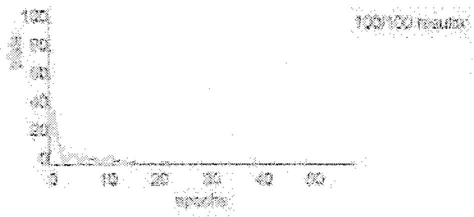
acg.jpg
fig. 255

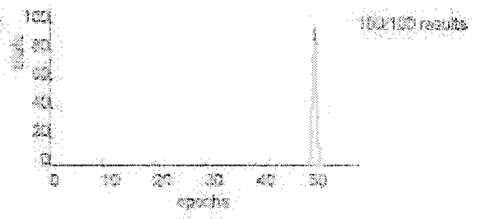


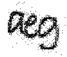




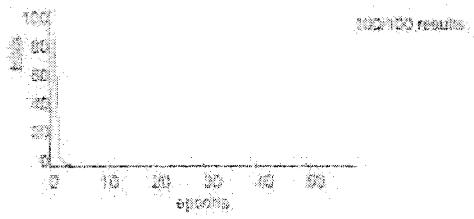
adg.jpg
fig. 256

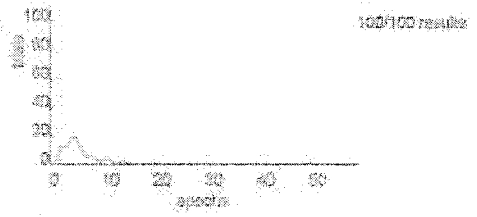






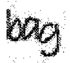
aeg.jpg
fig. 257

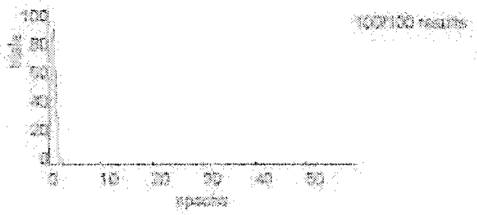


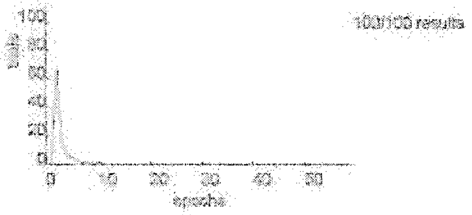


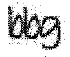
Analysis of Numerous Trials on Each Input Image

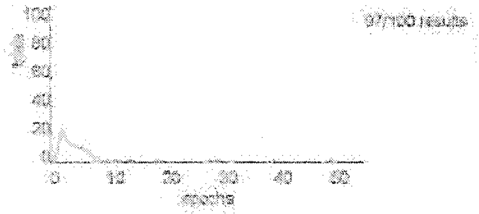
Input Image	Distribution of First Symbol Found	Distribution of Termination

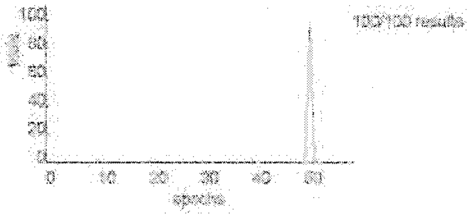

bag.jpg
fig. 258

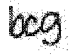


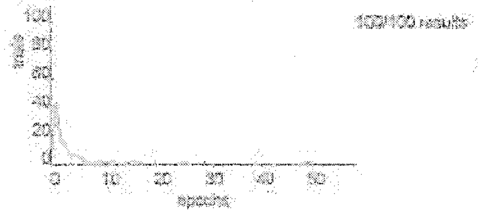


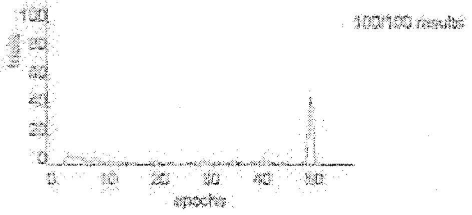

bbg.jpg
fig. 259

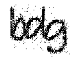


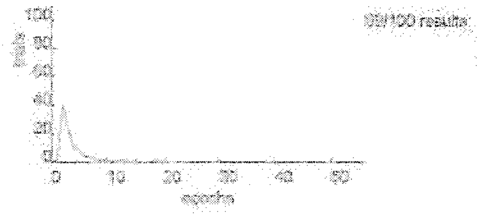


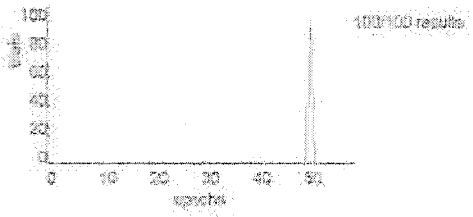

bcg.jpg
fig. 260

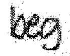


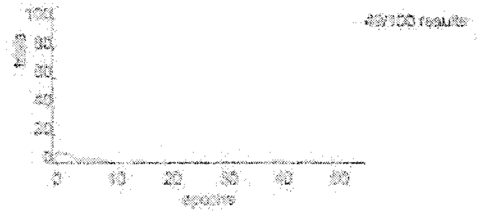


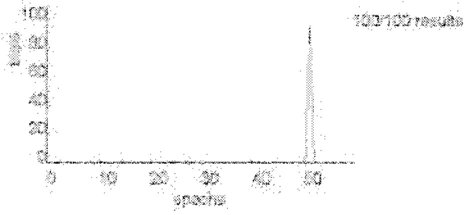

bdg.jpg
fig. 261






beg.jpg
fig. 262



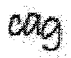


Analysis of Numerous Trials on Each Input Image

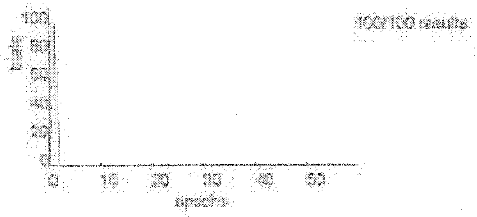
Input Image

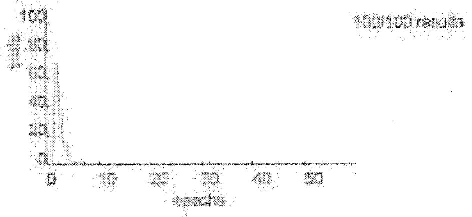
Distribution of First Symbol Found

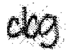
Distribution of Termination



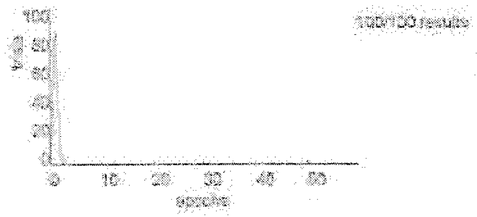
cag.jpg
fig. 263

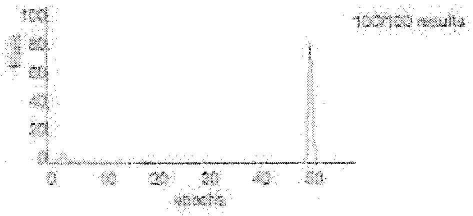


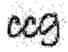




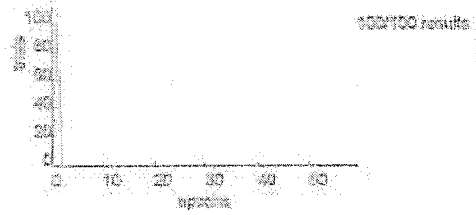
cbg.jpg
fig. 264

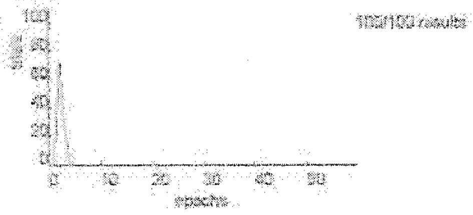


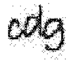




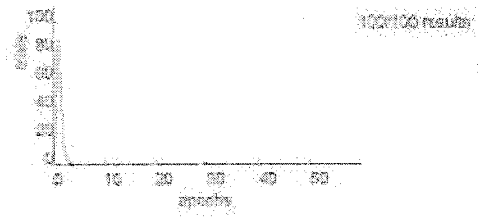
ccg.jpg
fig. 265

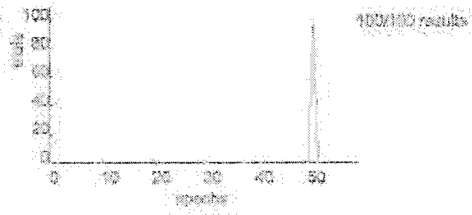


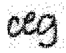




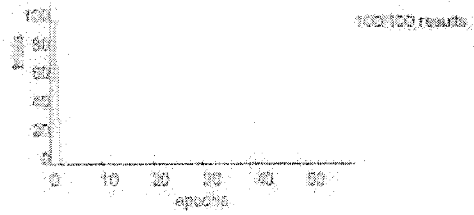
cdg.jpg
fig. 266

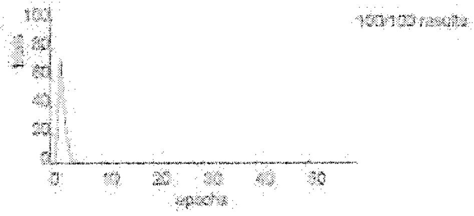






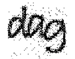
ceg.jpg
fig. 267



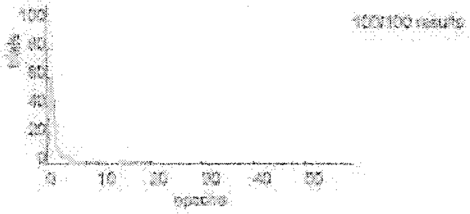


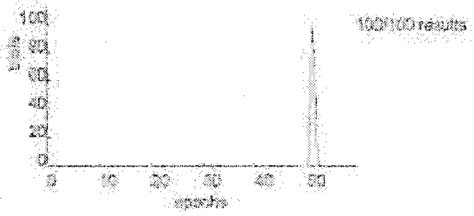
Analysis of Numerous Trials on Each Input Image

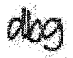
Input Image Distribution of First Symbol Found Distribution of Termination



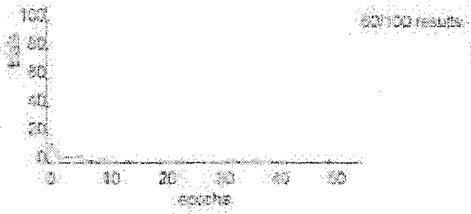
dag.jpg
fig. 268

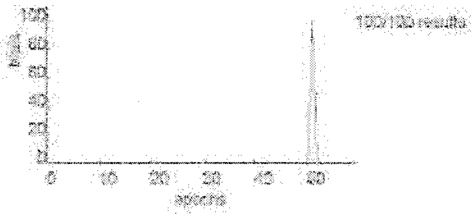


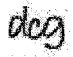




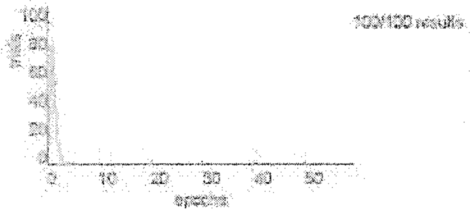
dbg.jpg
fig. 269

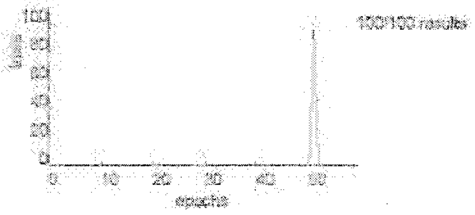


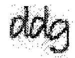




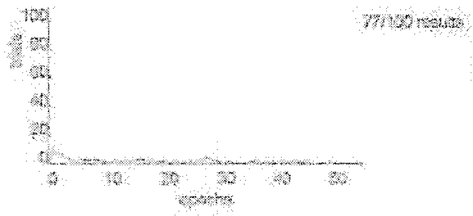
dcg.jpg
fig. 270

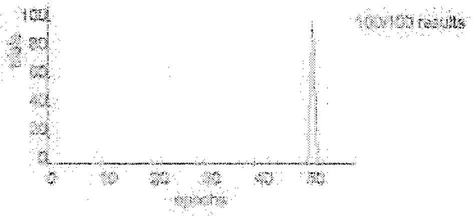


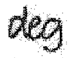




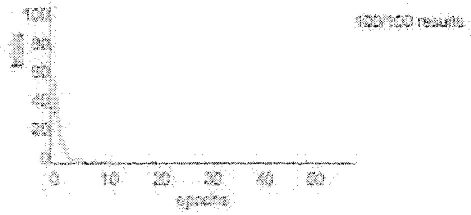
ddg.jpg
fig. 271

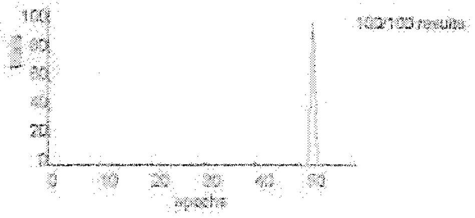






deg.jpg
fig. 272



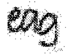


Analysis of Numerous Trials on Each Input Image

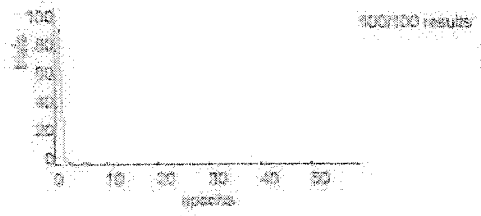
Input Image

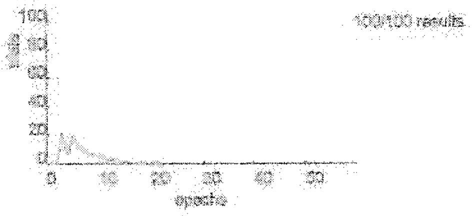
Distribution of First Symbol Found

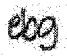
Distribution of Termination



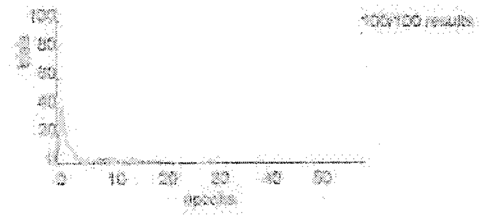
eag.jpg
fig. 273

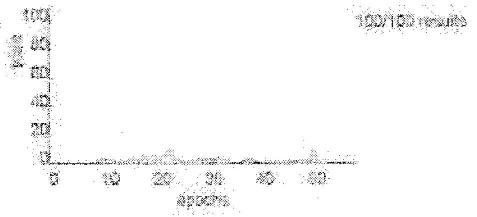


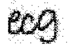




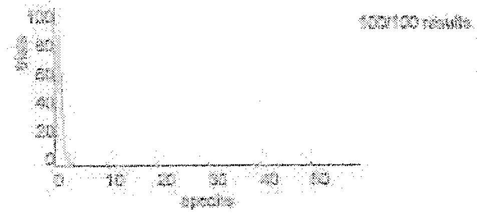
ebg.jpg
fig. 274

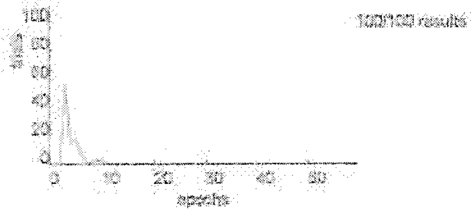


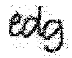




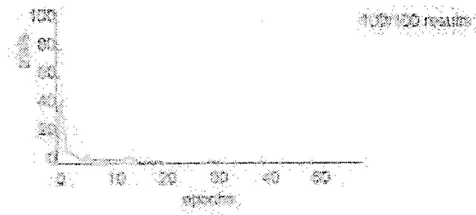
ecg.jpg
fig. 275

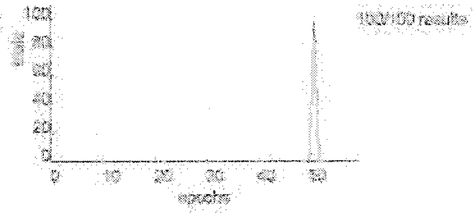


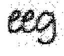




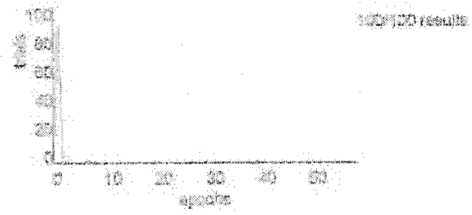
edg.jpg
fig. 276

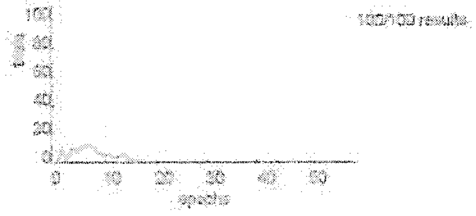






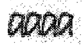
eeg.jpg
fig. 277



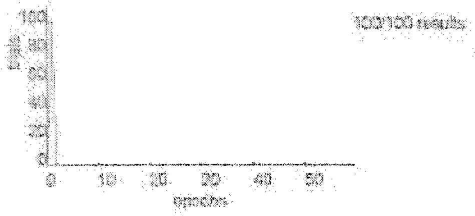


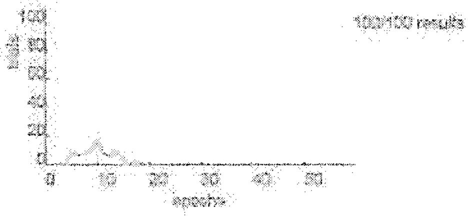
Analysis of Numerous Trials on Each Input Image

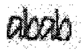
Input Image	Distribution of First Symbol Found	Distribution of Termination



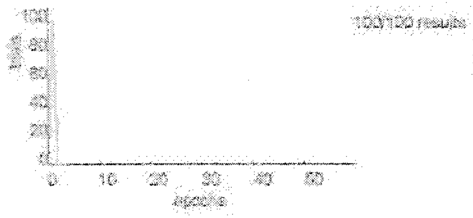
aaaa.jpg
fig. 278

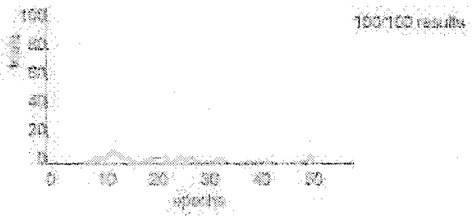


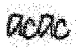




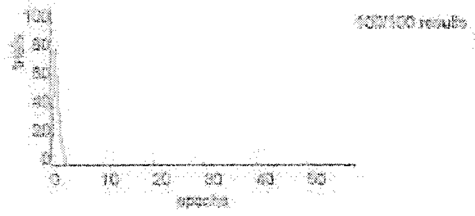
abab.jpg
fig. 279

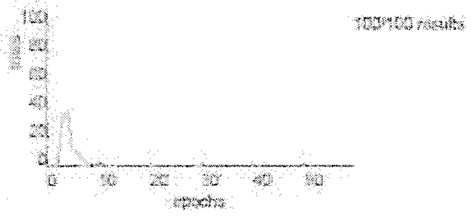


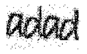




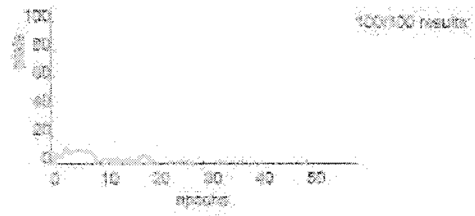
acac.jpg
fig. 280

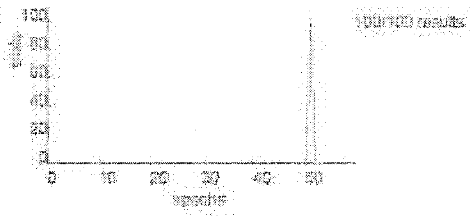


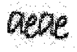




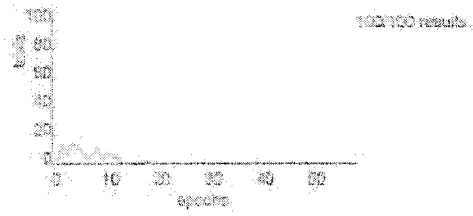
adad.jpg
fig. 281

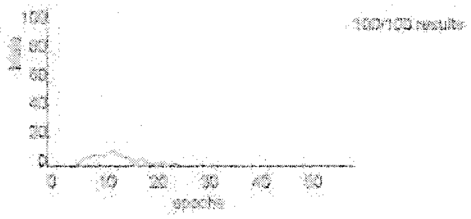






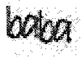
aeae.jpg
fig. 282



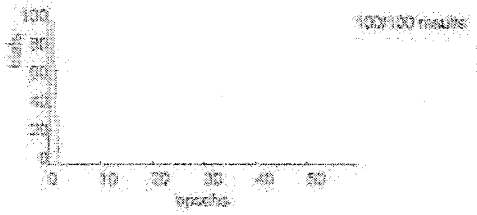


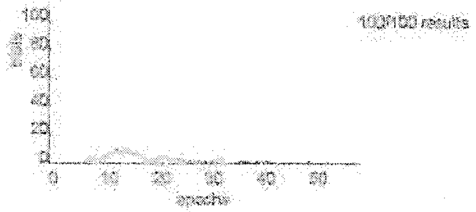
Analysis of Numerous Trials on Each Input Image

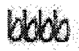
Input Image	Distribution of First Symbol Found	Distribution of Termination



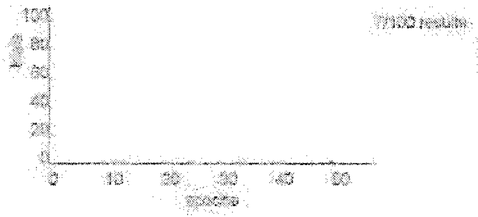
baba.jpg
fig. 283

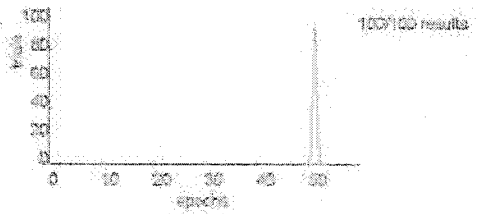


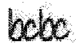




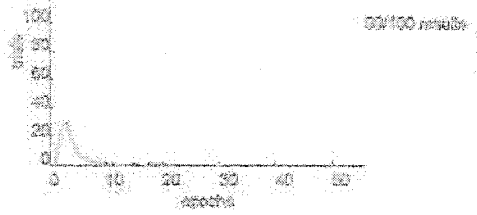
bbbb.jpg
fig. 284

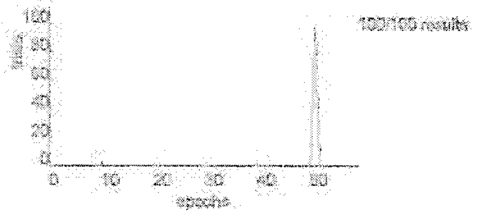


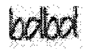




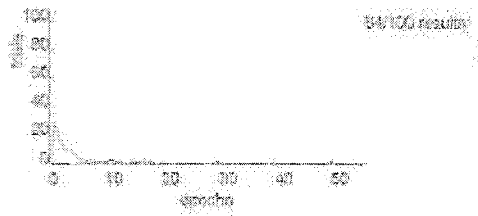
bcbc.jpg
fig. 285

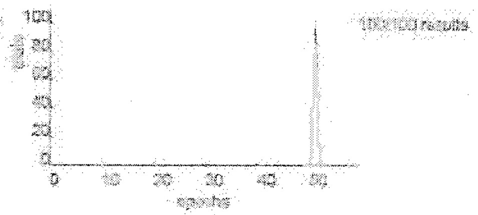


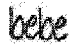




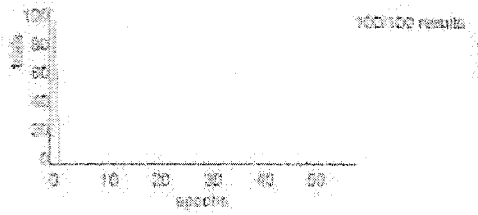
bdbd.jpg
fig. 286

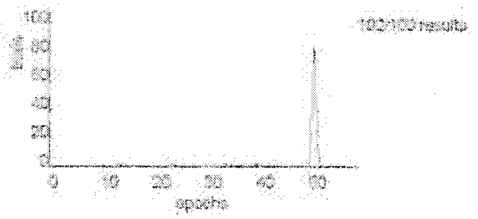






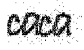
bebe.jpg
fig. 287



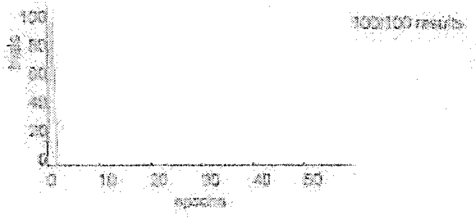


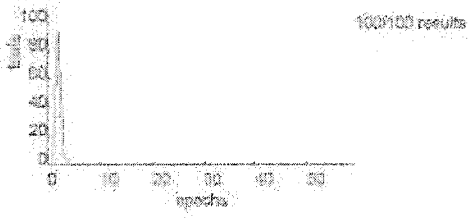
Analysis of Numerous Trials on Each Input Image

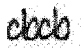
Input Image   Distribution of First Symbol Found   Distribution of Termination



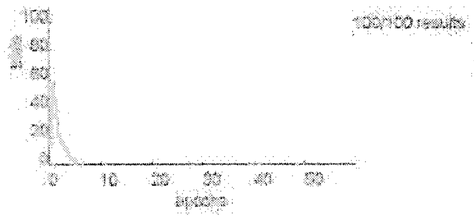
caca.jpg
fig. 288

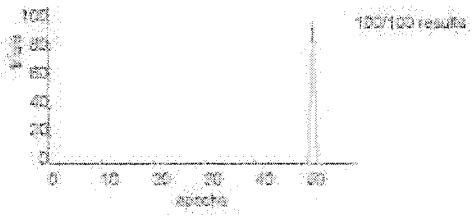


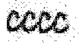




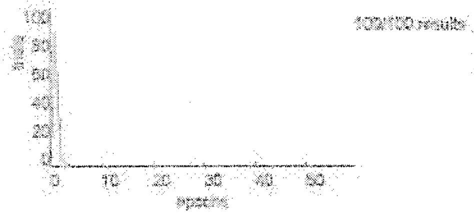
cbcb.jpg
fig. 289

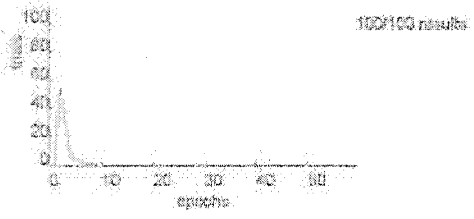


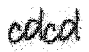




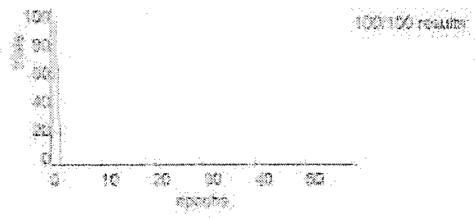
cccc.jpg
fig. 290

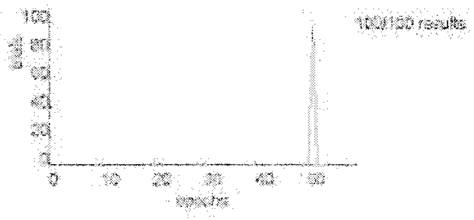


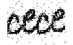




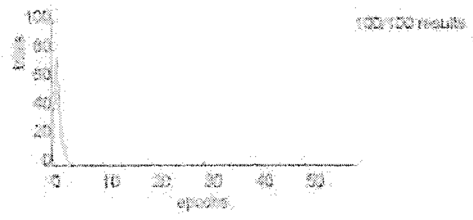
cdcd.jpg
fig. 291

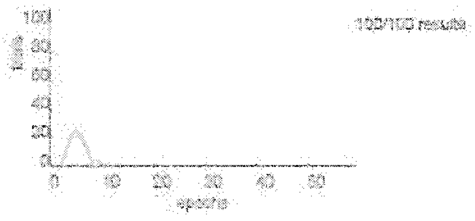






ceca.jpg
fig. 292



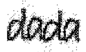


Analysis of Numerous Trials on Each Input Image

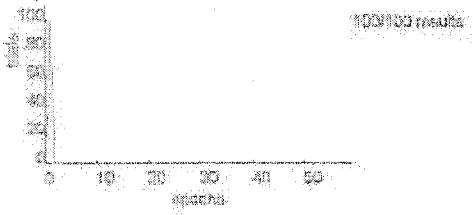
Input Image

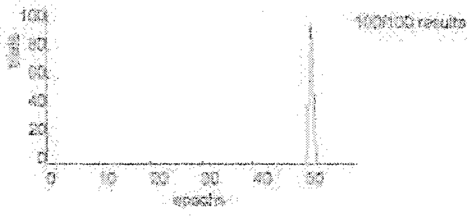
Distribution of First Symbol Found

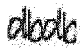
Distribution of Termination



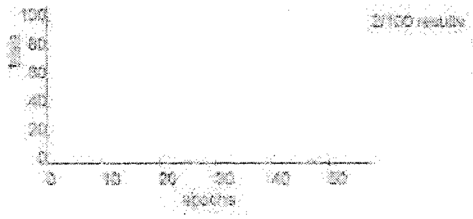
dada.jpg
fig. 293

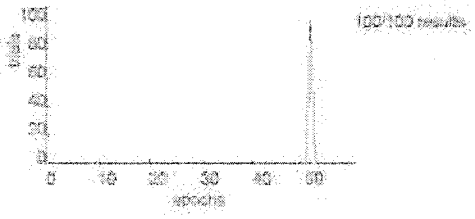


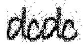




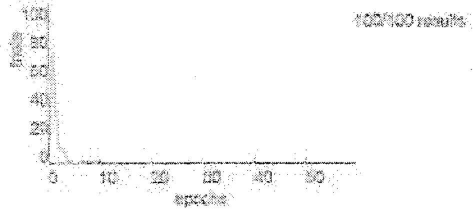
dbdb.jpg
fig. 294

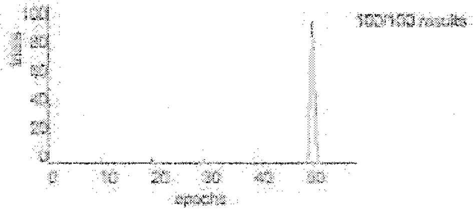


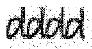




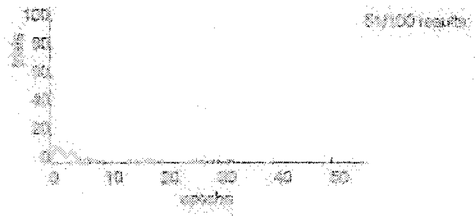
dcdc.jpg
fig. 295

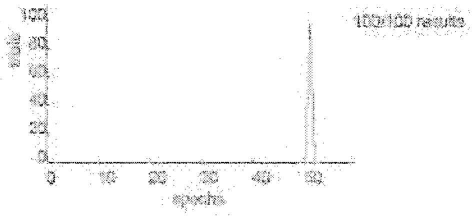


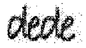




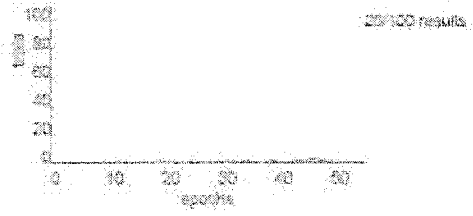
dddd.jpg
fig. 296

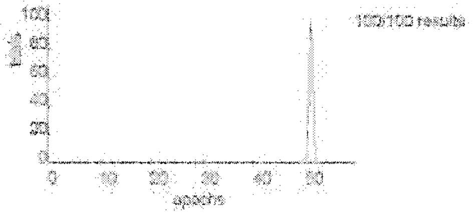






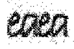
dede.jpg
fig. 297



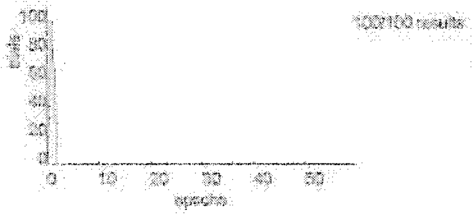


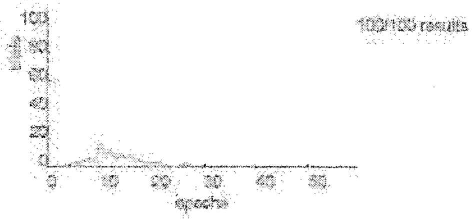
Analysis of Numerous Trials on Each Input Image

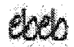
Input Image Distribution of First Symbol Found Distribution of Termination



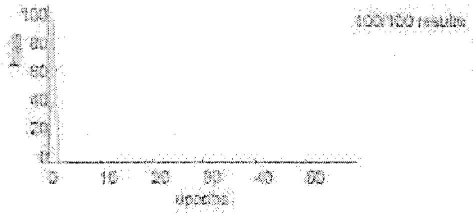
eaea.jpg
fig. 298

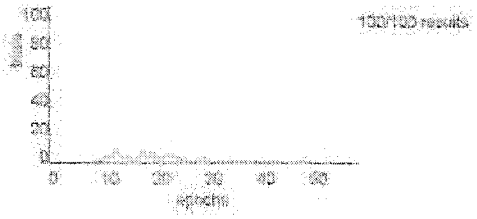


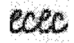




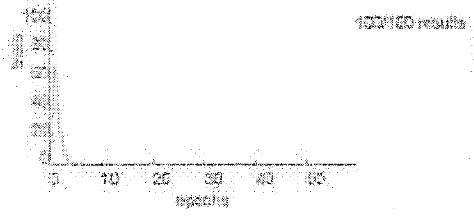
ebeb.jpg
fig. 299

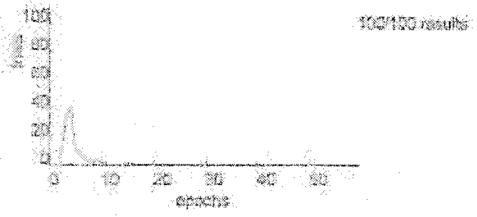


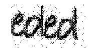




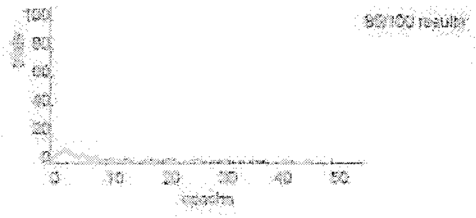
ecec.jpg
fig. 300

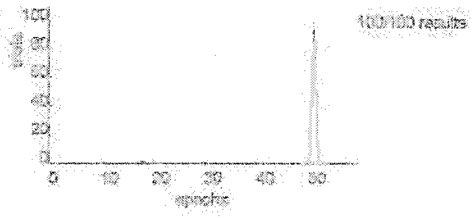


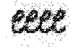




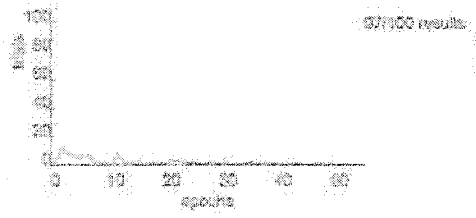
eded.jpg
fig. 301

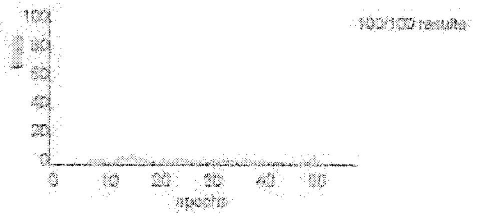


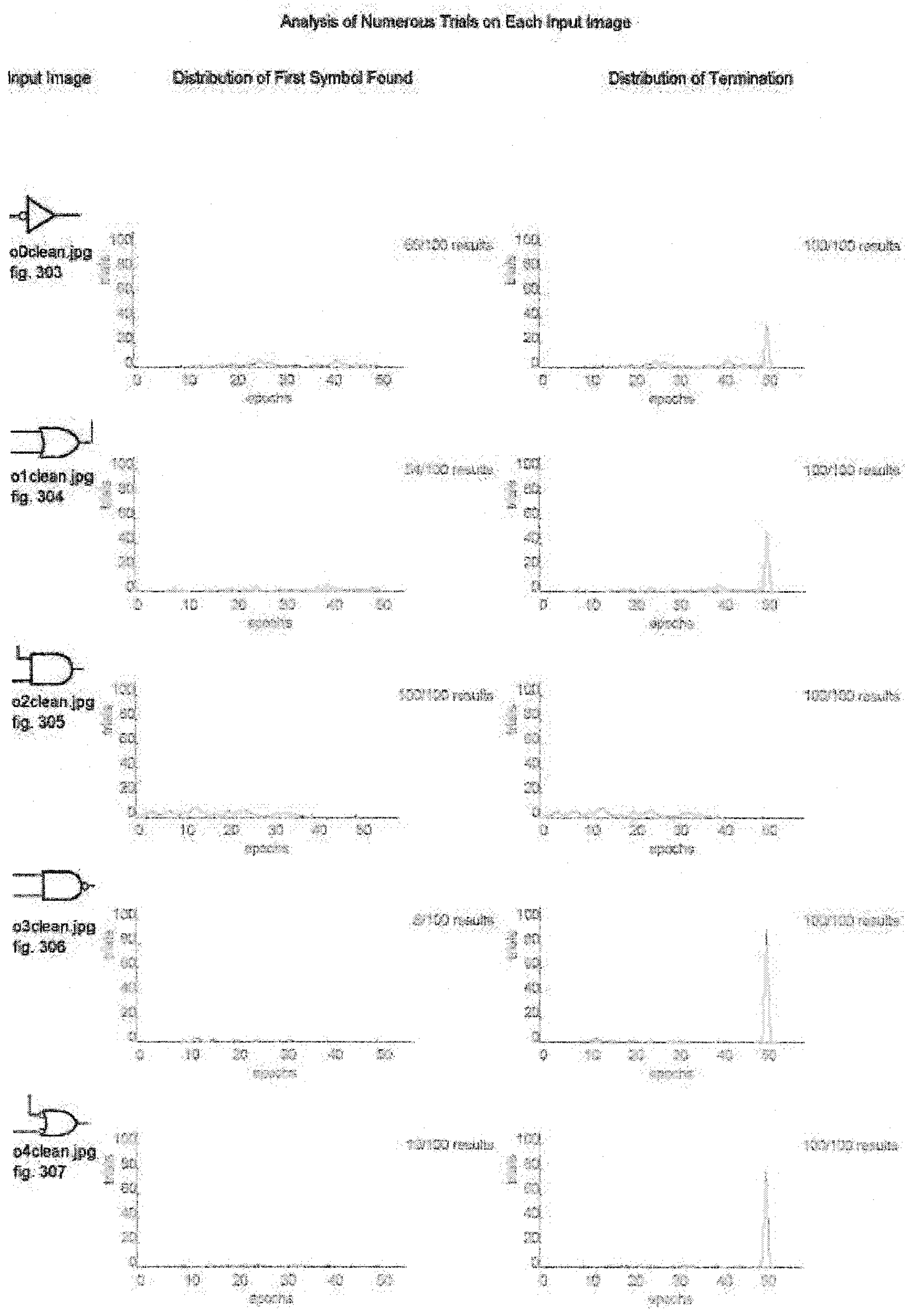




eeee.jpg
fig. 302

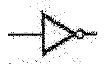


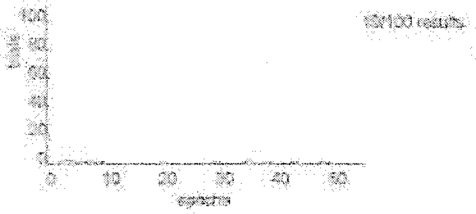


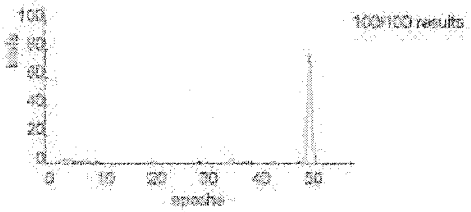

Analysis of Numerous Trials on Each Input Image
Input Image
Distribution of First Symbol Found
Distribution of Termination
o0clean.jpg
fig. 303
66/100 results
100/100 results
o1clean.jpg
fig. 304
54/100 results
100/100 results
o2clean.jpg
fig. 305
100/100 results
100/100 results
o3clean.jpg
fig. 306
8/100 results
100/100 results
o4clean.jpg
fig. 307
13/100 results
100/100 results
trials
epochs
0
10
20
30
40
50
60
80
100

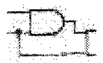
Analysis of Numerous Trials on Each Input Image

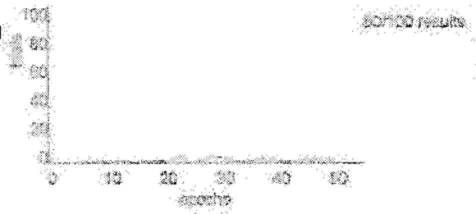
Input Image **Distribution of First Symbol Found** **Distribution of Termination**

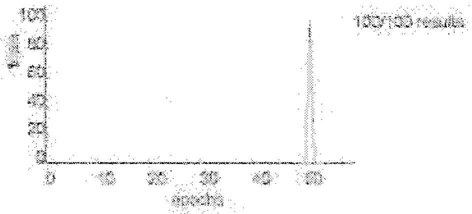

o5clean.jpg
fig. 308

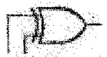


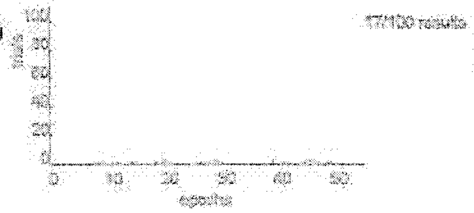


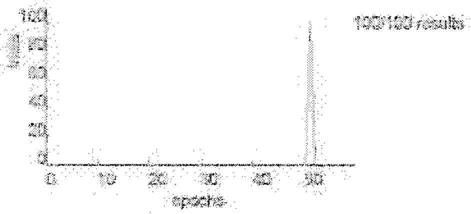

o6clean.jpg
fig. 309

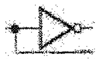


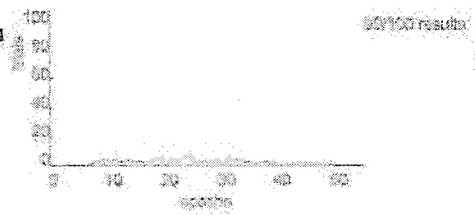


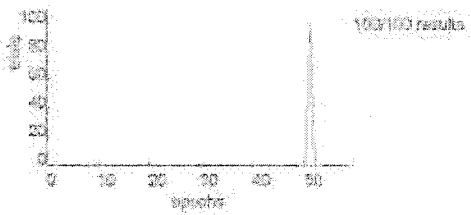

o7clean.jpg
fig. 310

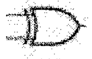


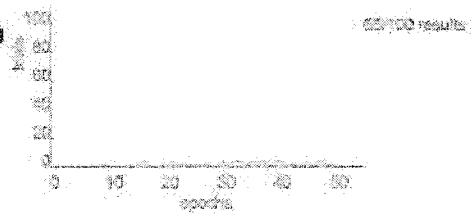


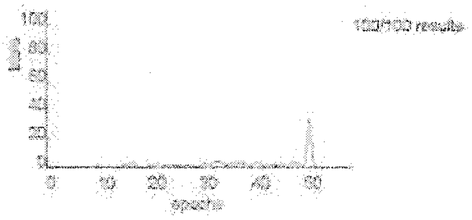

o8clean.jpg
fig. 311





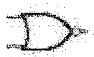

o9clean.jpg
fig. 312



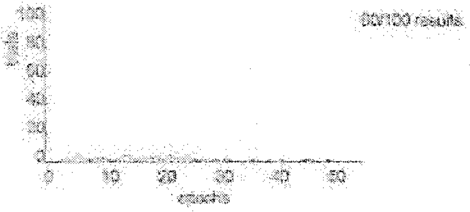


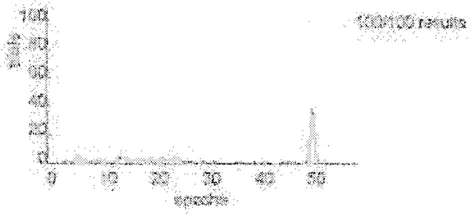
Analysis of Numerous Trials on Each Input Image

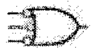
Input Image Distribution of First Symbol Found Distribution of Termination



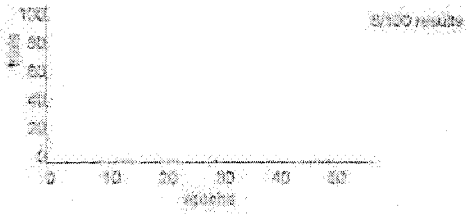
o10clean.jp
fig. 313

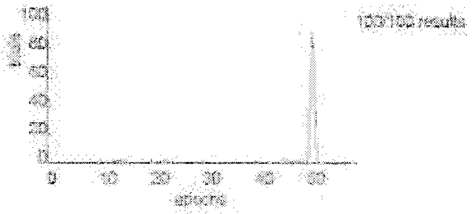


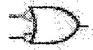




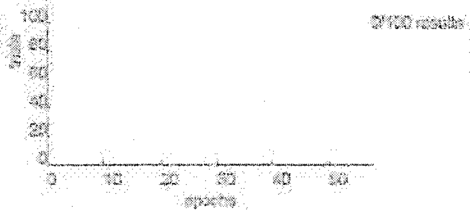
o11clean.jp
fig. 314

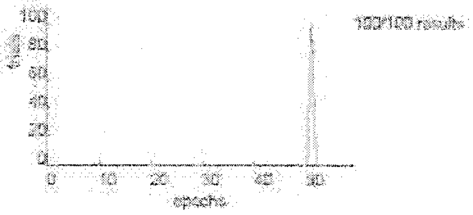


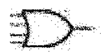




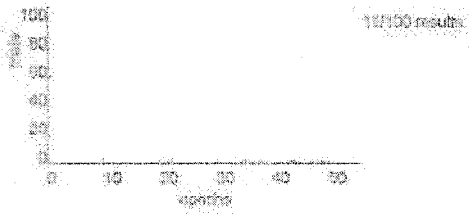
o12clean.jp
fig. 315

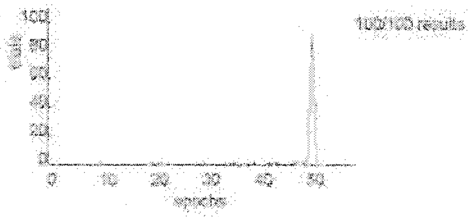


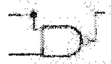




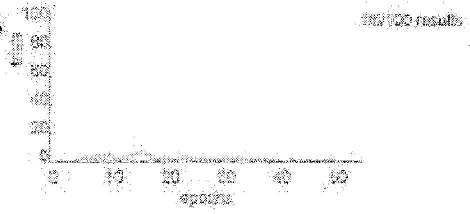
o14clean.jp
fig. 316

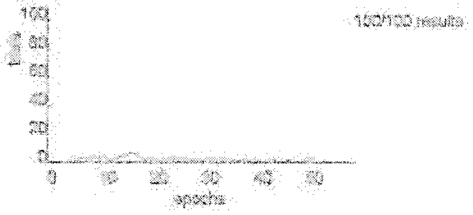






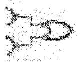
o15clean.jp
fig. 317



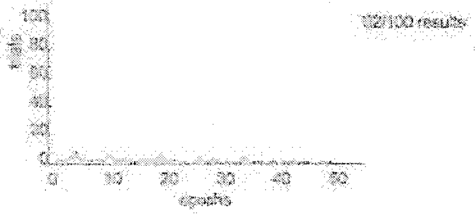


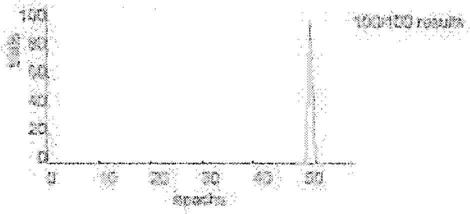
Analysis of Numerous Trials on Each Input Image

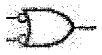
Input Image Distribution of First Symbol Found Distribution of Termination



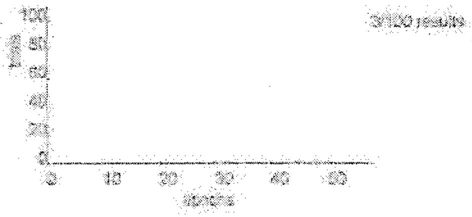
o16clean.jp
fig. 318

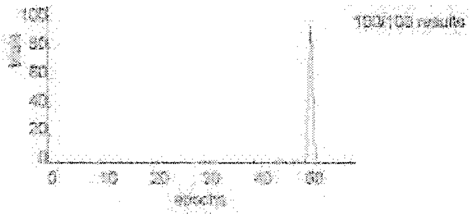


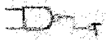




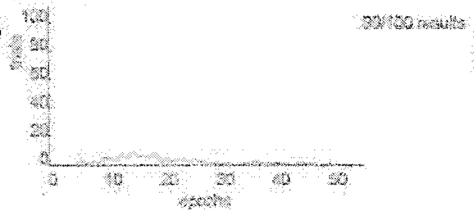
o17clean.jp
fig. 319

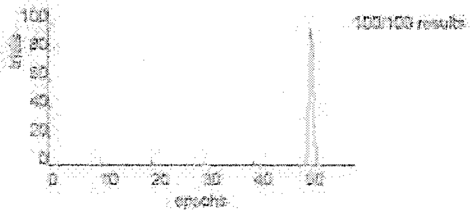


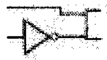




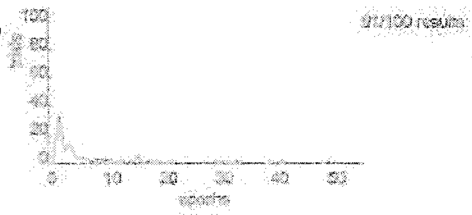
o18clean.jp
fig. 320

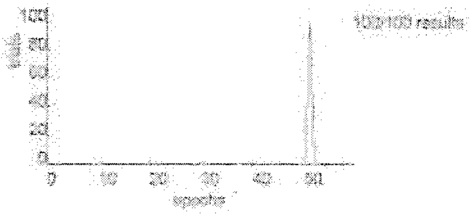


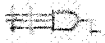




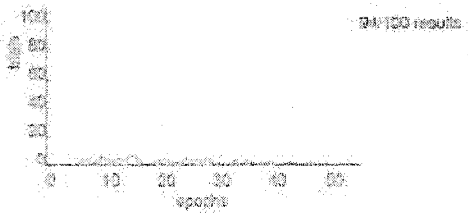
o19clean.jp
fig. 321

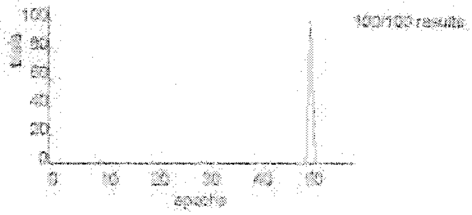






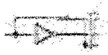
o20clean.jp
fig. 322



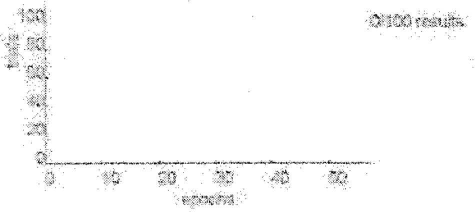


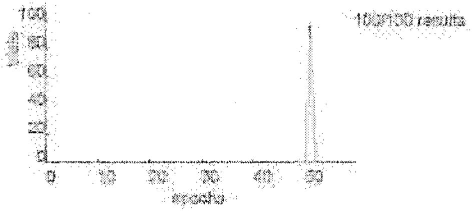
Analysis of Numerous Trials on Each Input Image

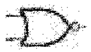
Input Image **Distribution of First Symbol Found** **Distribution of Termination**



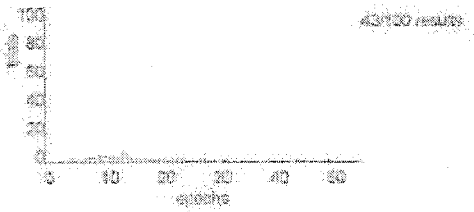
o21clean.jp
fig. 323

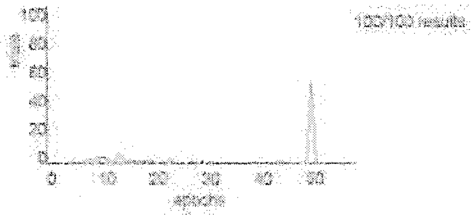


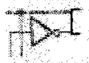




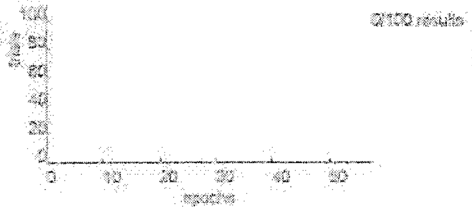
o22clean.jp
fig. 324

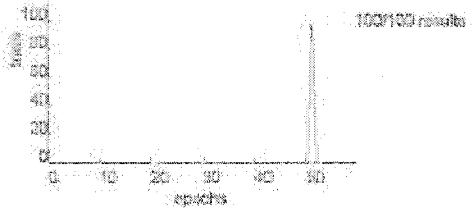


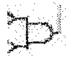




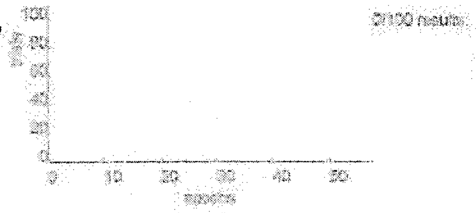
o23clean.jp
fig. 325

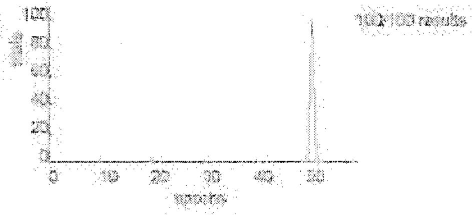


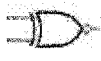




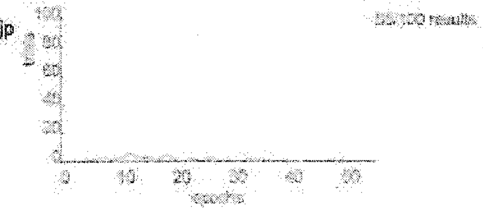
o24clean.jp
fig. 326

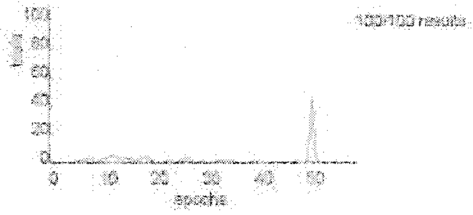






o25clean.jp
fig. 327



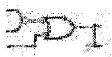


Analysis of Numerous Trials on Each Input Image

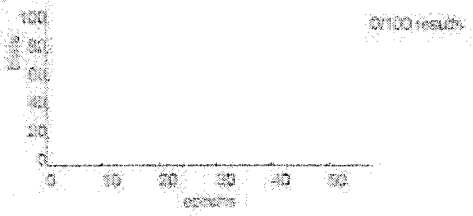
Input Image

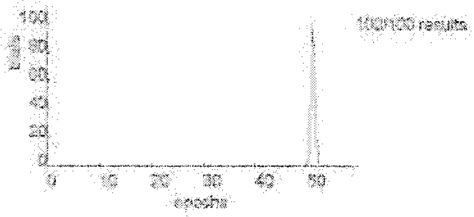
Distribution of First Symbol Found

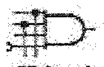
Distribution of Termination



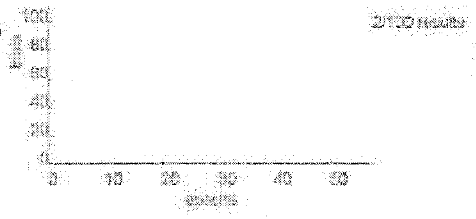
o26clean.jp
fig. 328

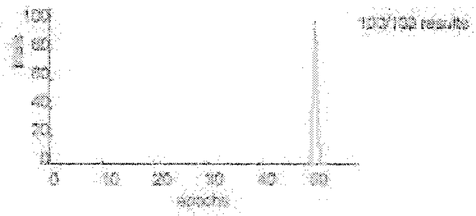


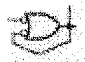




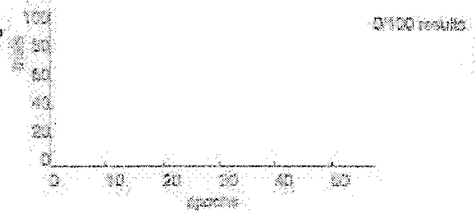
o27clean.jp
fig. 329

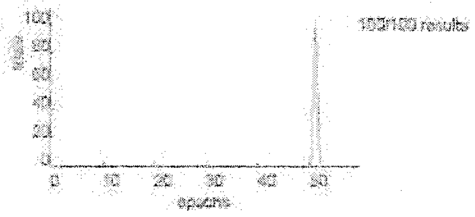


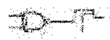




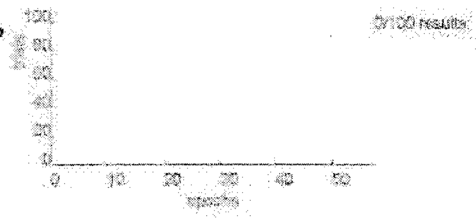
o28clean.jp
fig. 330

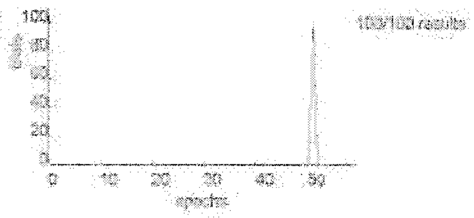






o29clean.jp
fig. 331



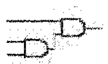


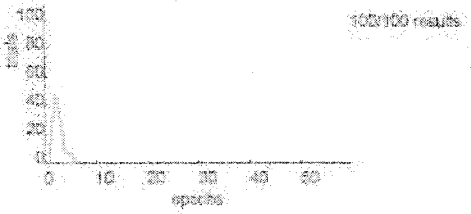
Analysis of Numerous Trials on Each Input Image

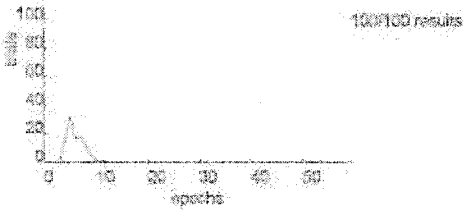
Input Image

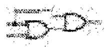
Distribution of First Symbol Found

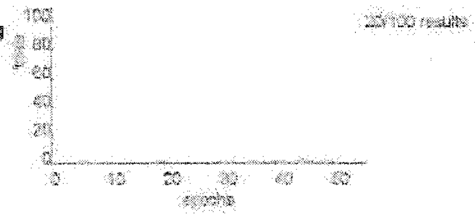
Distribution of Termination

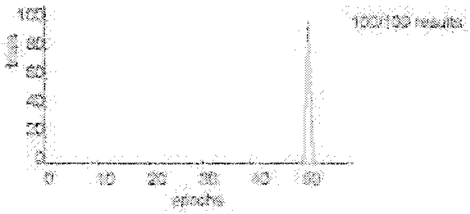

d0clean.jpg
fig. 332

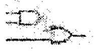


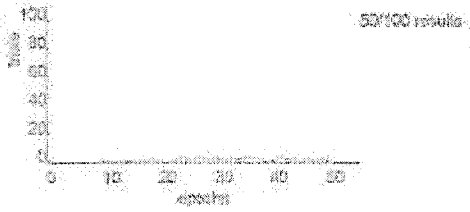


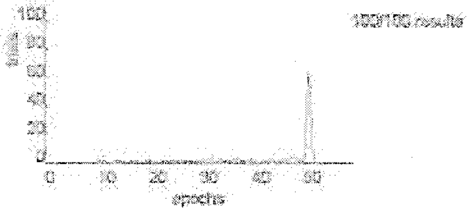

d1clean.jpg
fig. 333

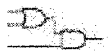


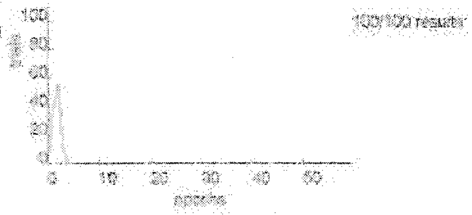


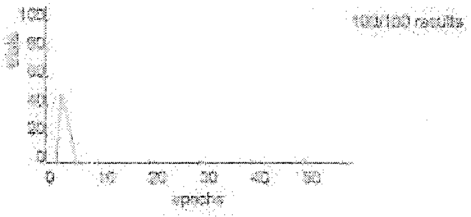

d2clean.jpg
fig. 334



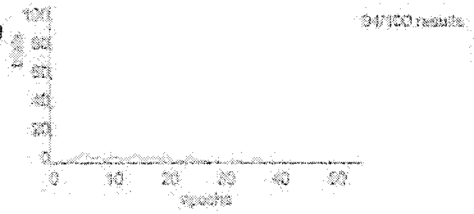


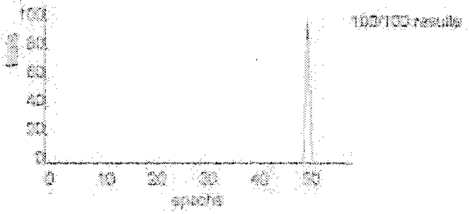

d3clean.jpg
fig. 335





d4clean.jpg
fig. 336



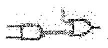


Analysis of Numerous Trials on Each Input Image

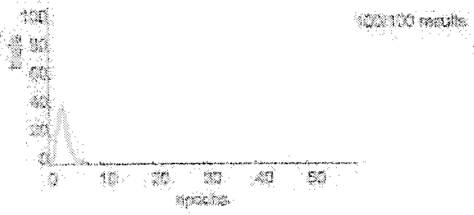
Input Image

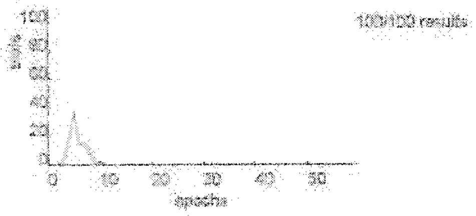
Distribution of First Symbol Found

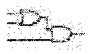
Distribution of Termination



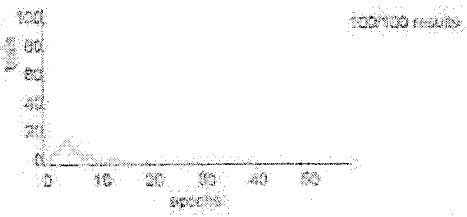
d5clean.jpg
fig. 337

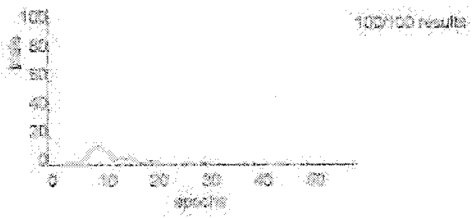


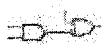




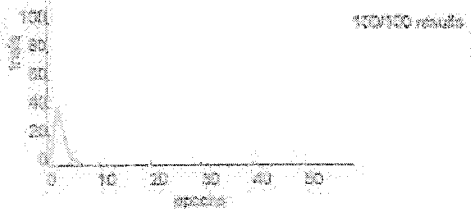
d6clean.jpg
fig. 338

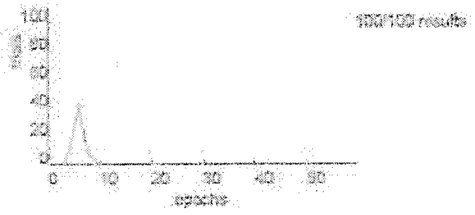


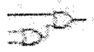




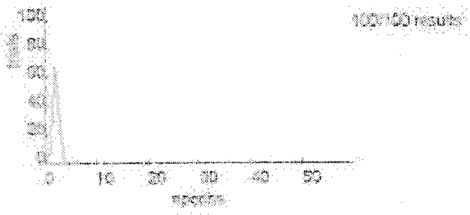
d7clean.jpg
fig. 339

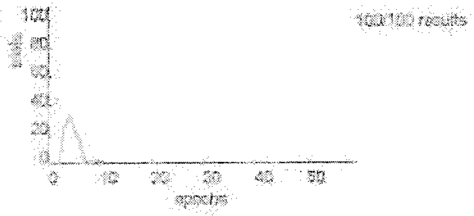


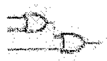




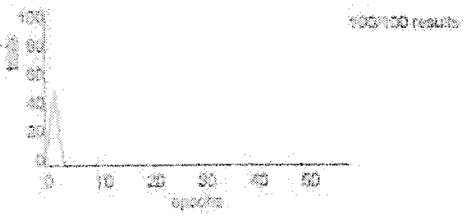
d8clean.jpg
fig. 340

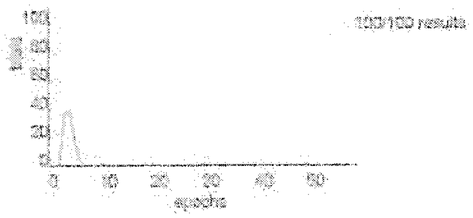




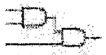
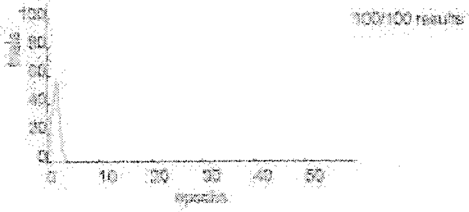
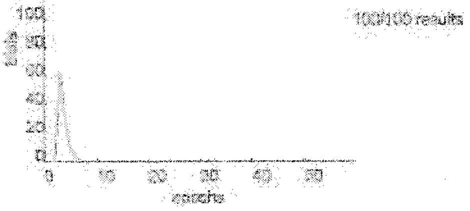
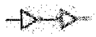
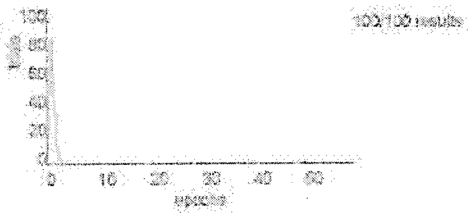
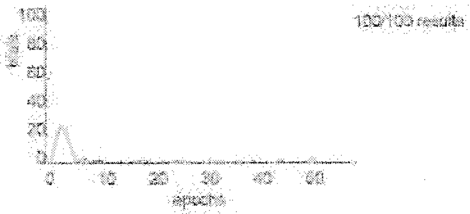
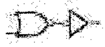
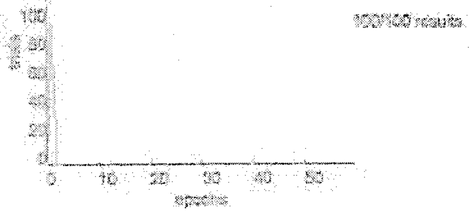
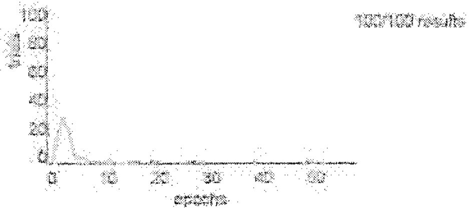
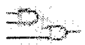
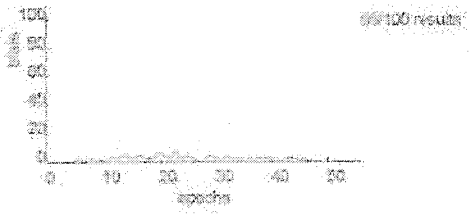
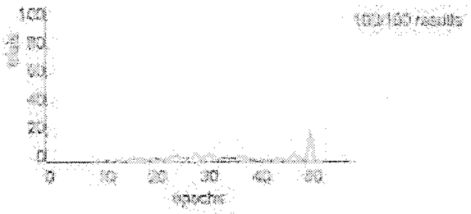
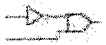
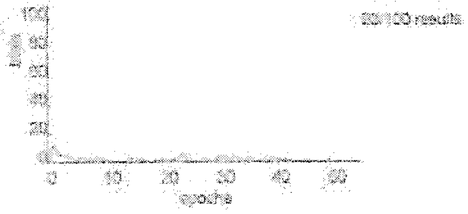
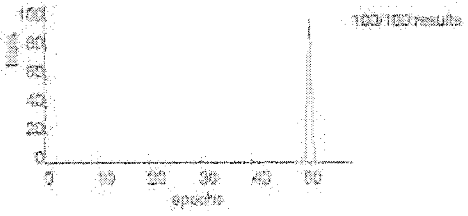


d9clean.jpg
fig. 341





Analysis of Numerous Trials on Each Input Image

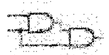
Input Image	Distribution of First Symbol Found	Distribution of Termination
 **d10clean.jp fig. 342**		
 **d11clean.jp fig. 343**		
 **d13clean.jp fig. 344**		
 **d12clean.jp fig. 345**		
 **d14clean.jp fig. 346**		

Analysis of Numerous Trials on Each Input Image

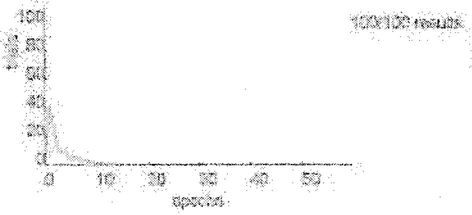
Input Image

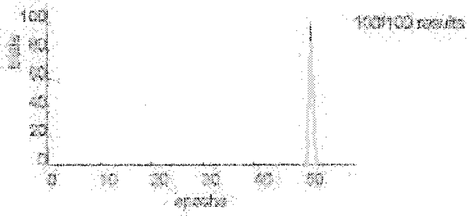
Distribution of First Symbol Found

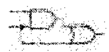
Distribution of Termination



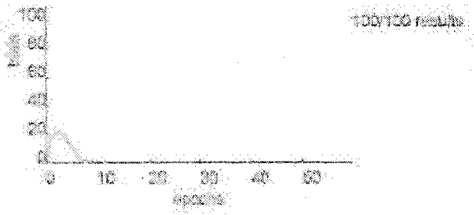
d15clean.jp
fig. 347

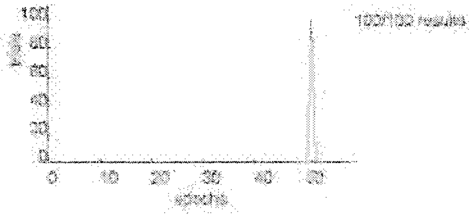


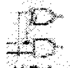




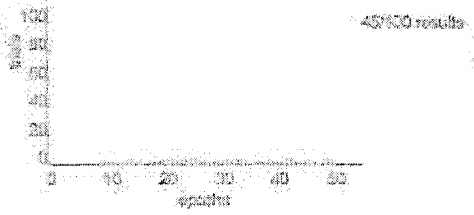
d16clean.jp
fig. 348

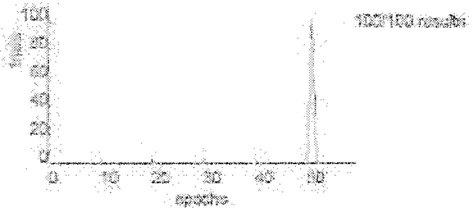


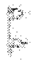




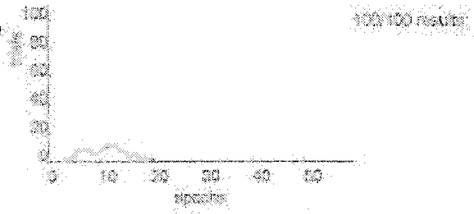
d17clean.jp
fig. 349

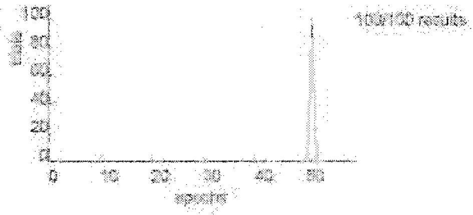


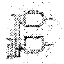




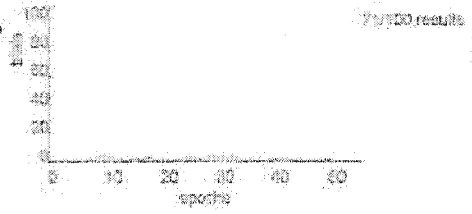
d18clean.jp
fig. 350

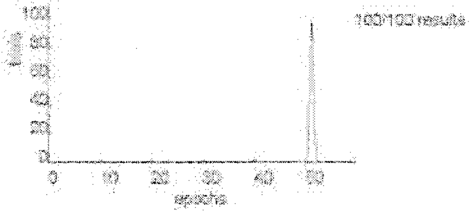






d19clean.jp
fig. 351



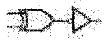


Analysis of Numerous Trials on Each Input Image

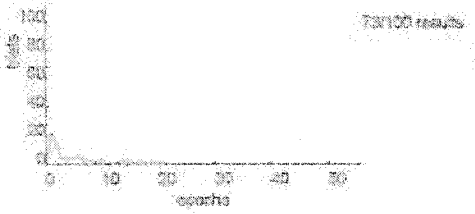
Input Image

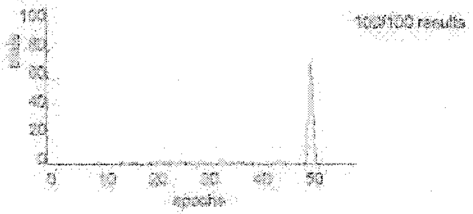
Distribution of First Symbol Found

Distribution of Termination

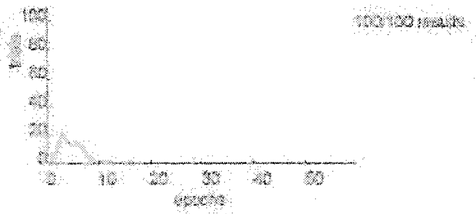


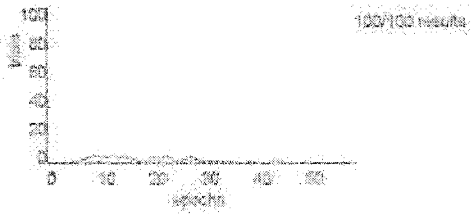
d20clean.jp
fig. 352

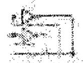




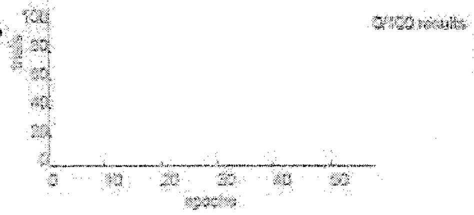
d21clean.jp
fig. 353

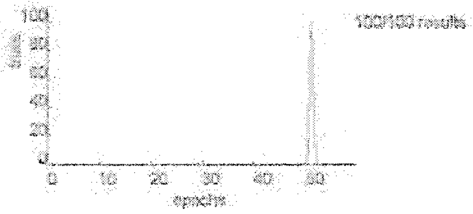


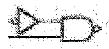




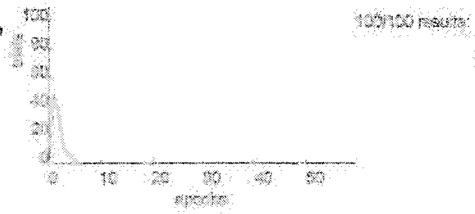
d22clean.jp
fig. 354

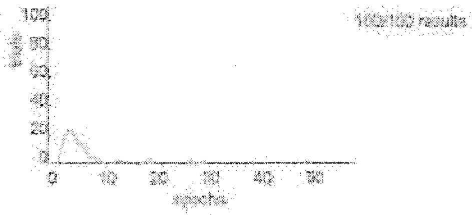


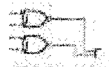




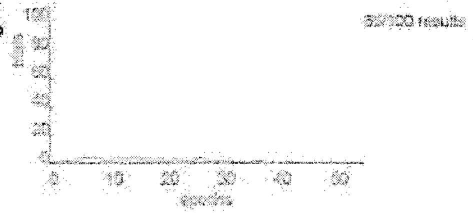
d23clean.jp
fig. 355

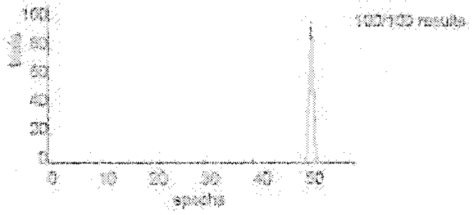






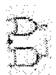
d24clean.jp
fig. 356



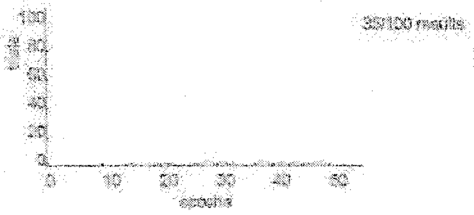


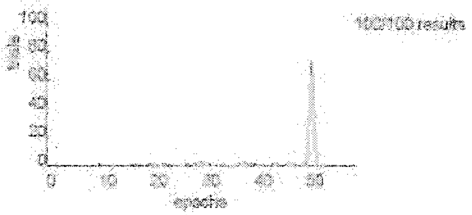
Analysis of Numerous Trials on Each Input Image

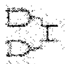
Input Image **Distribution of First Symbol Found** **Distribution of Termination**



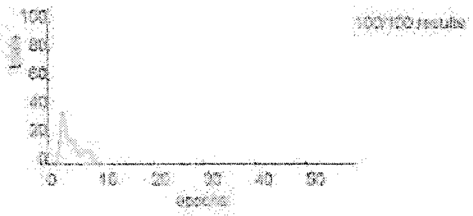
d25clean.jp
fig. 357

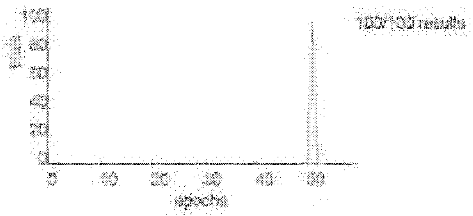


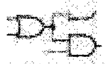




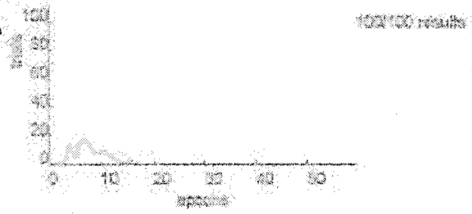
d26clean.jp
fig. 358

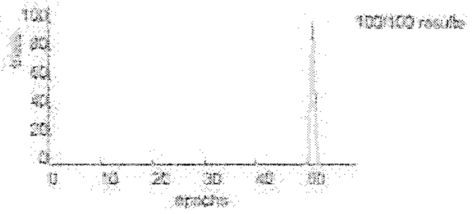


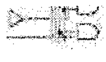




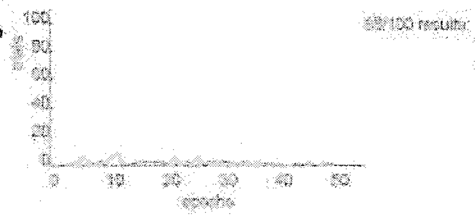
d27clean.jp
fig. 359

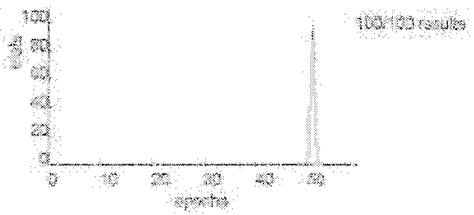






d28clean.jp
fig. 360



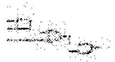


Analysis of Numerous Trials on Each Input Image

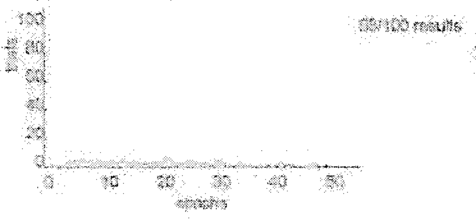
Input Image

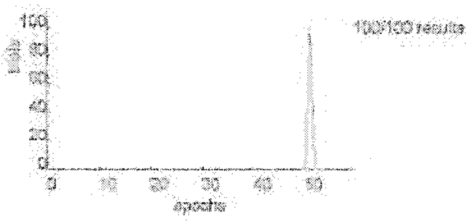
Distribution of First Symbol Found

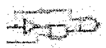
Distribution of Termination



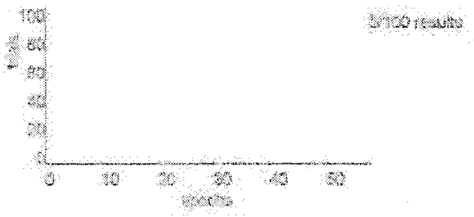
t0clean.jpg
fig. 361

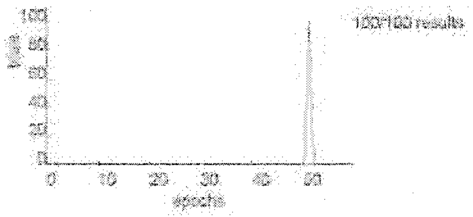


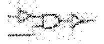




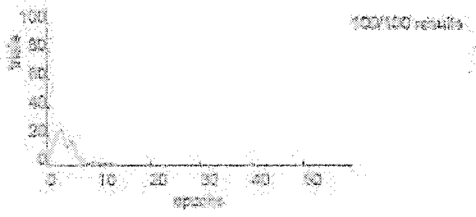
t1clean.jpg
fig. 362

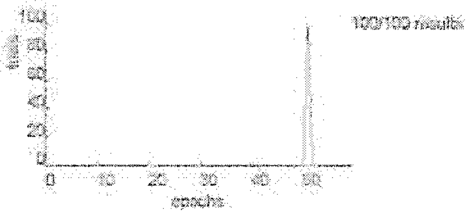


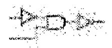




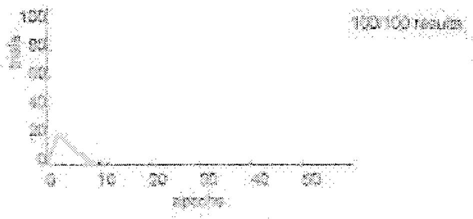
t2clean.jpg
fig. 363

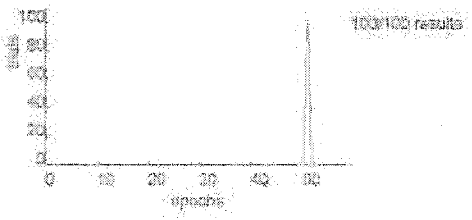


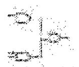




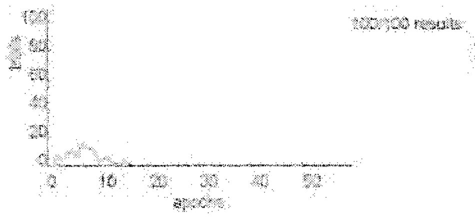
t3clean.jpg
fig. 364

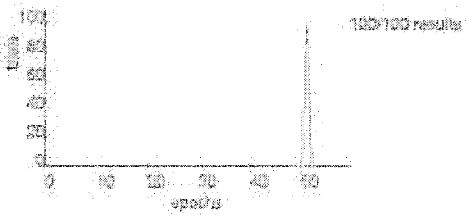




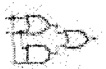
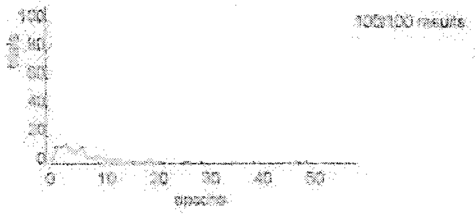
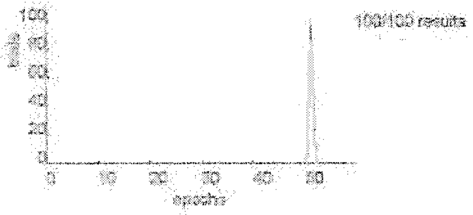
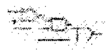
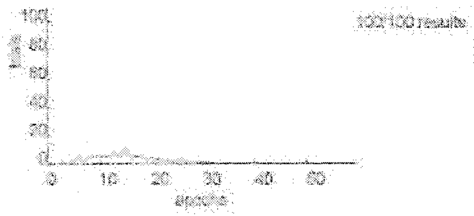
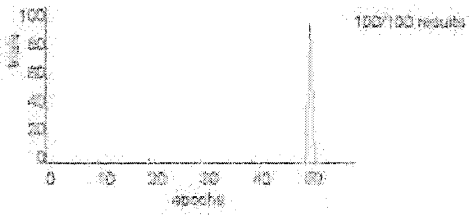
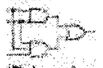
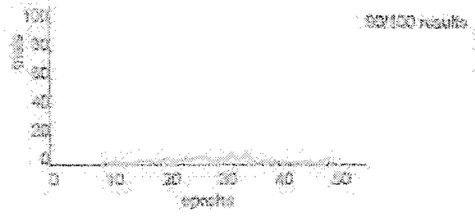
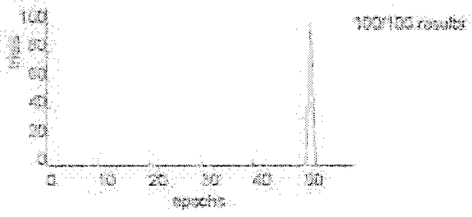
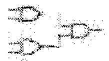
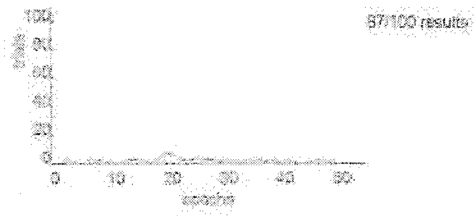
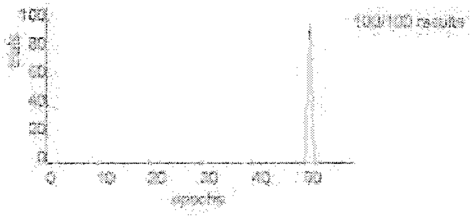
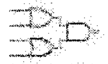
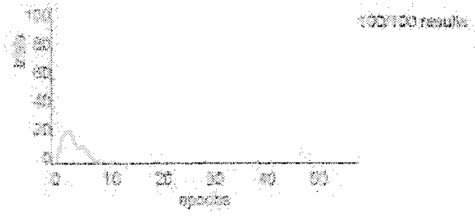
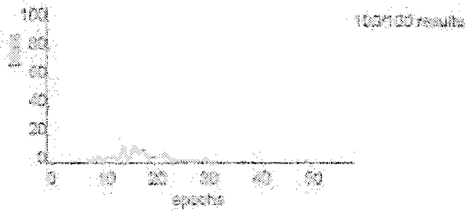


t4clean.jpg
fig. 365





Analysis of Numerous Trials on Each Input Image

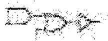
Input Image	Distribution of First Symbol Found	Distribution of Termination
 t5clean.jpg fig. 366		
 t6clean.jpg fig. 367		
 t7clean.jpg fig. 368		
 t8clean.jpg fig. 369		
 t9clean.jpg fig. 370		

Analysis of Numerous Trials on Each Input Image

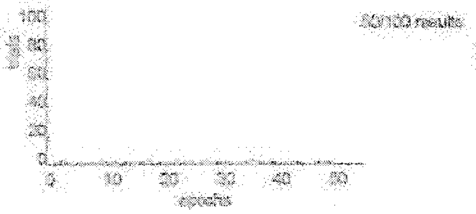
Input Image

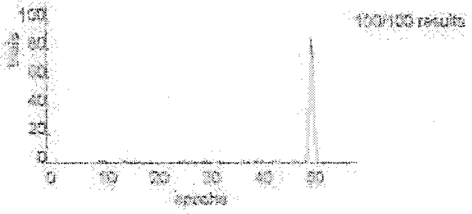
Distribution of First Symbol Found

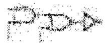
Distribution of Termination



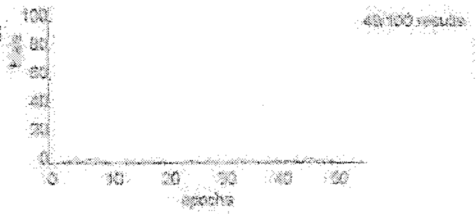
t10clean.jpg
fig. 371

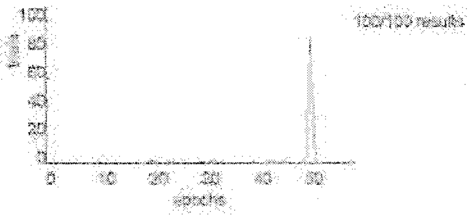


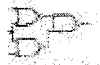




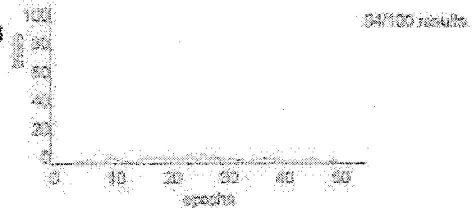
t11clean.jpg
fig. 372

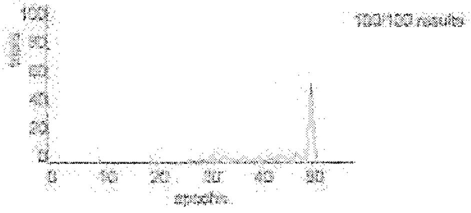


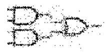




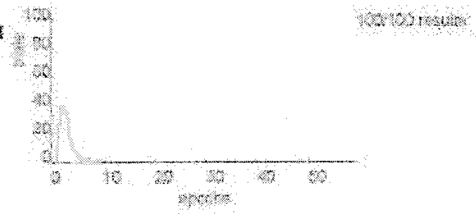
t12clean.jpg
fig. 373

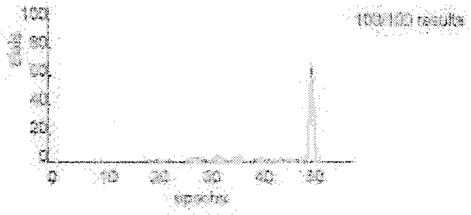


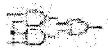




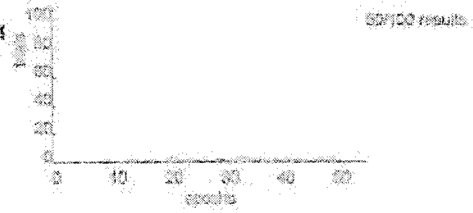
t13clean.jpg
fig. 374

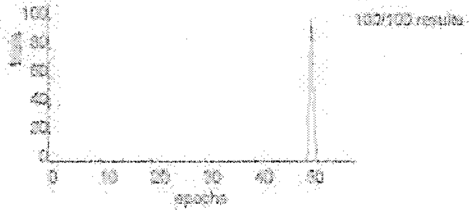






t14clean.jpg
fig. 375



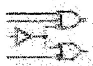


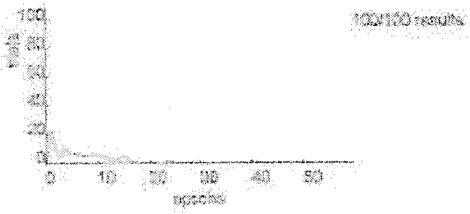
Analysis of Numerous Trials on Each Input Image

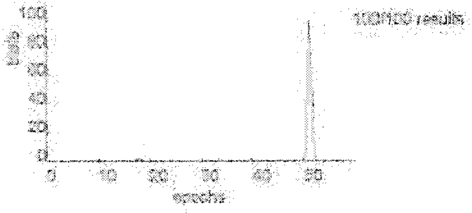
Input Image

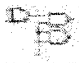
Distribution of First Symbol Found

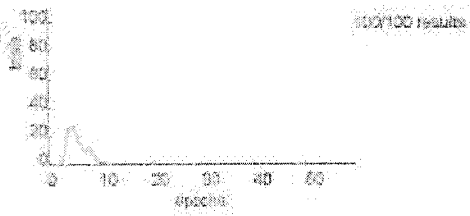
Distribution of Termination

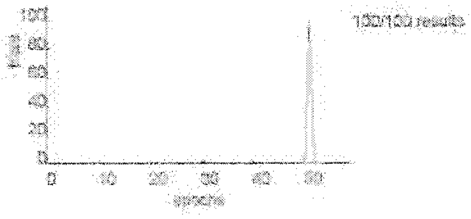

t15clean.jpg
fig. 376

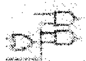


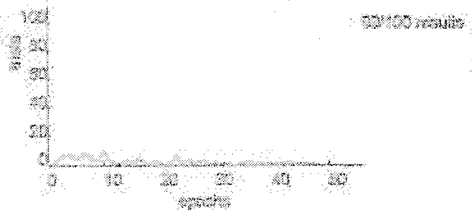


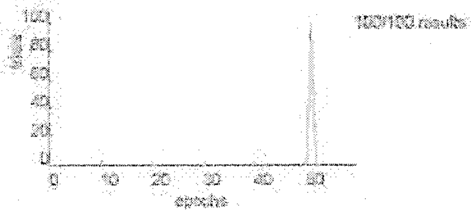

t16clean.jpg
fig. 377

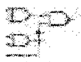


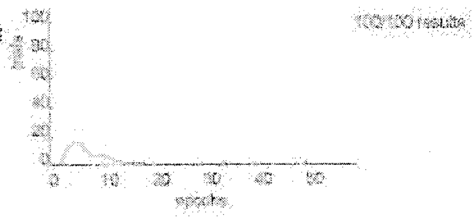


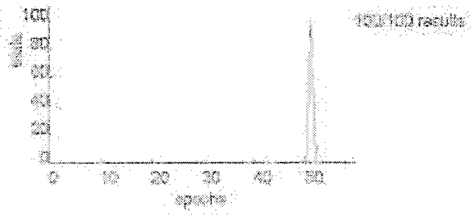

t17clean.jpg
fig. 378

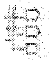


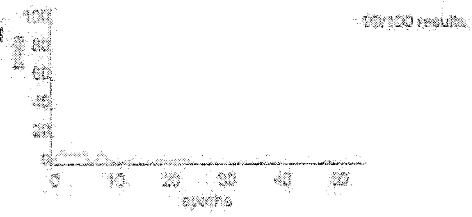


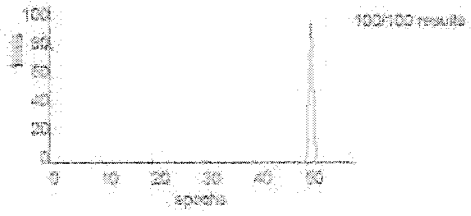

t18clean.jpg
fig. 379





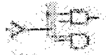

t19clean.jpg
fig. 380



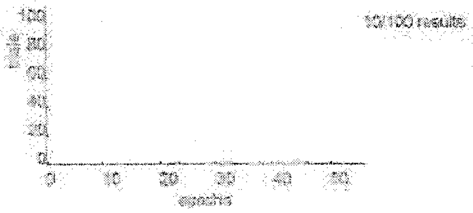


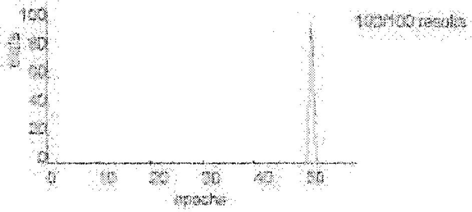
Analysis of Numerous Trials on Each Input Image

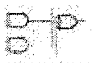
Input Image **Distribution of First Symbol Found** **Distribution of Termination**



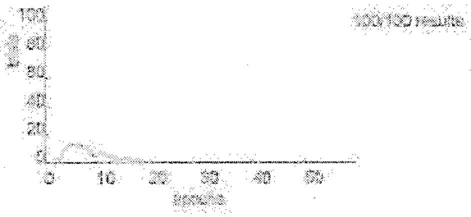
t20clean.jpg
fig. 381

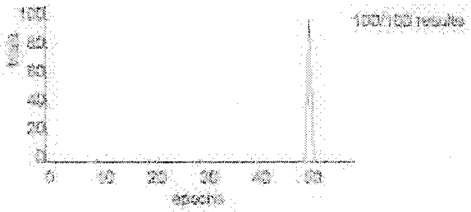


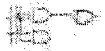




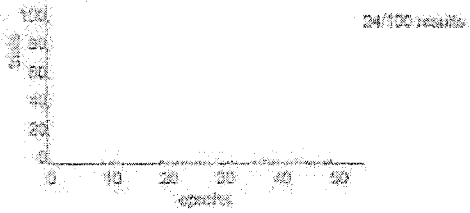
t21clean.jpg
fig. 382

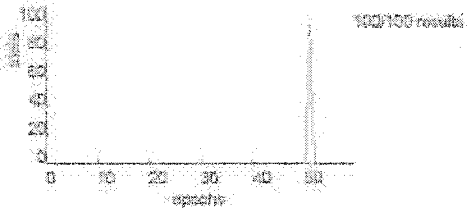


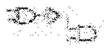




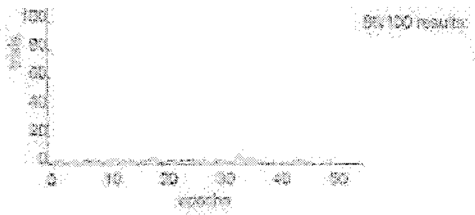
t22clean.jpg
fig. 383

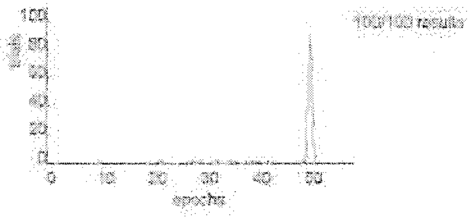


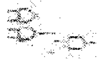




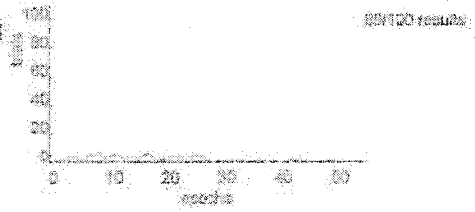
t23clean.jpg
fig. 384

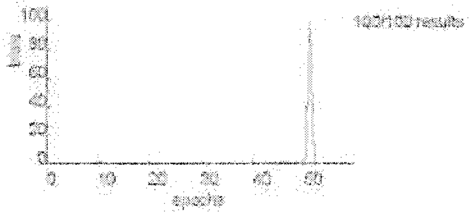






t24clean.jpg
fig. 385





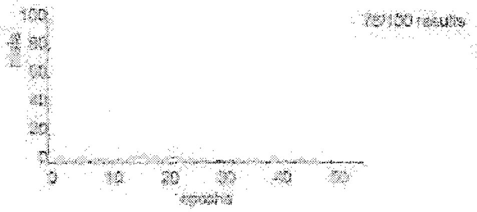
Analysis of Numerous Trials on Each input Image

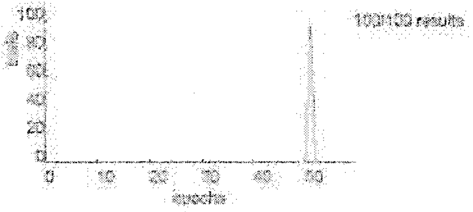
Input Image

Distribution of First Symbol Found

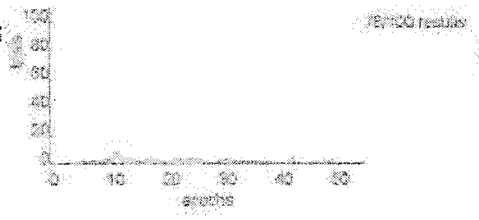
Distribution of Termination

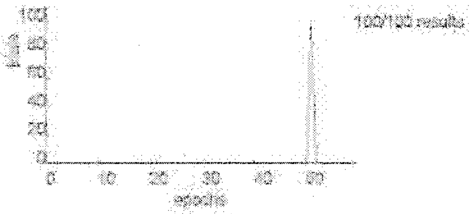
t25clean.jpg
fig. 386

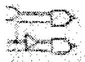




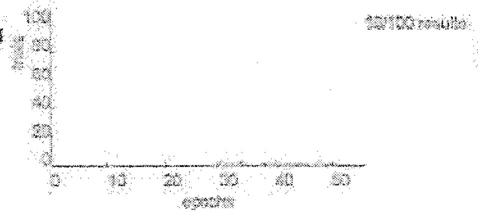
t26clean.jpg
fig. 387

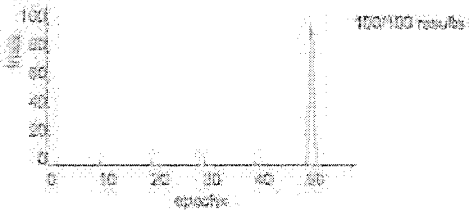


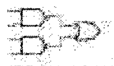




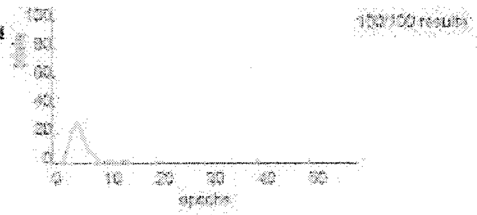
t27clean.jpg
fig. 388

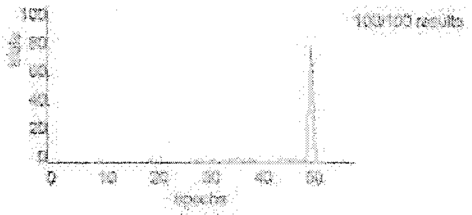


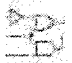




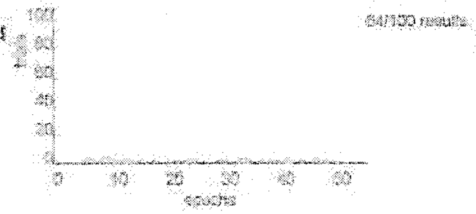
t28clean.jpg
fig. 389

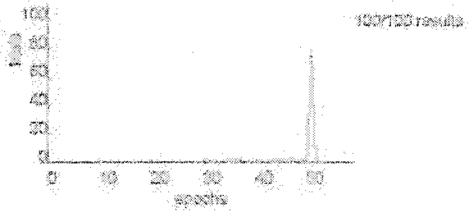






t29clean.jpg
fig. 390

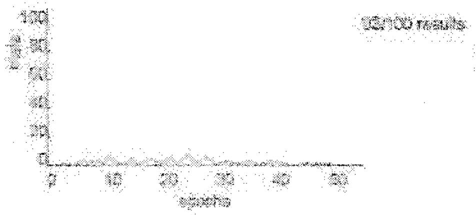


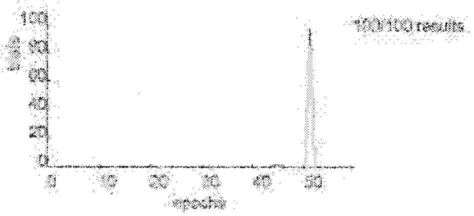


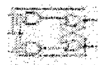
Analysis of Numerous Trials on Each Input Image

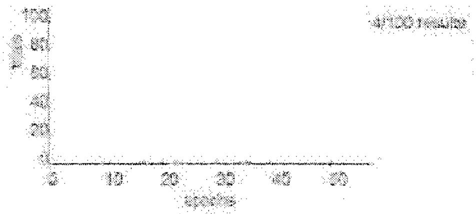
Input Image	Distribution of First Symbol Found	Distribution of Termination

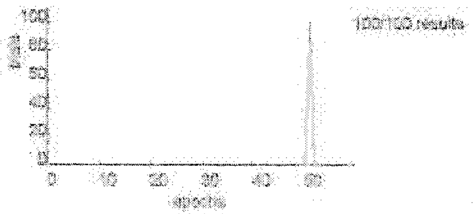
f0clean.jpg
fig. 391

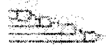


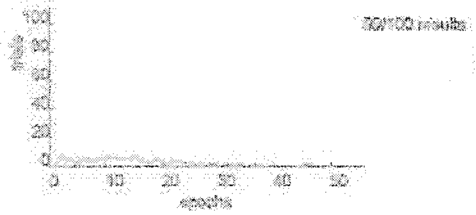


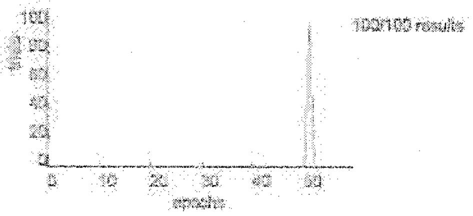

f1clean.jpg
fig. 392

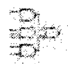


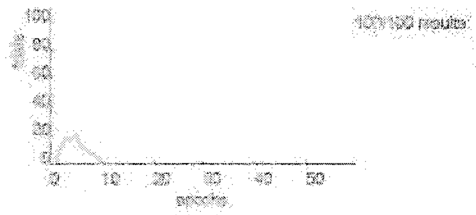


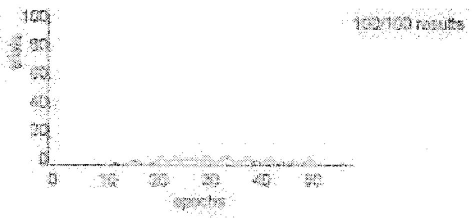

f2clean.jpg
fig. 393



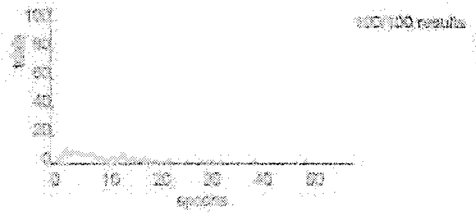


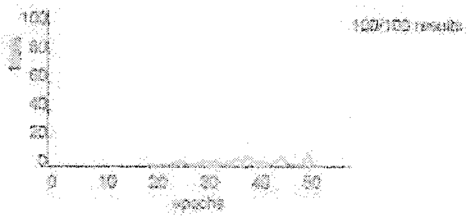

f3clean.jpg
fig. 394





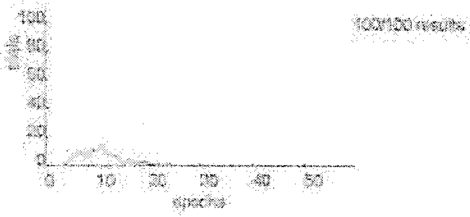
f4clean.jpg
fig. 395

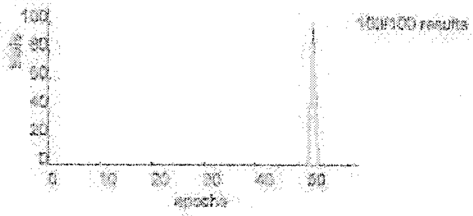


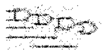


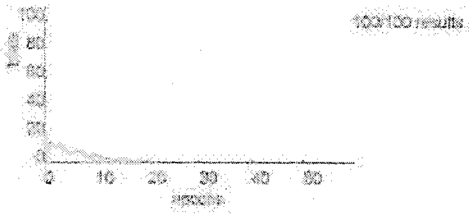
Analysis of Numerous Trials on Each Input Image

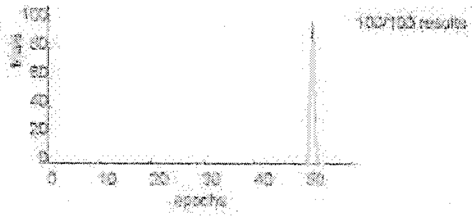
Input Image   Distribution of First Symbol Found   Distribution of Termination

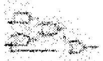


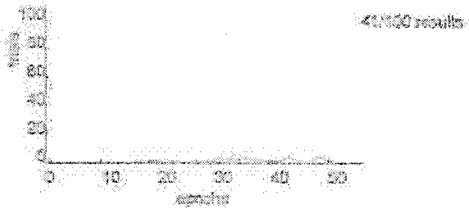


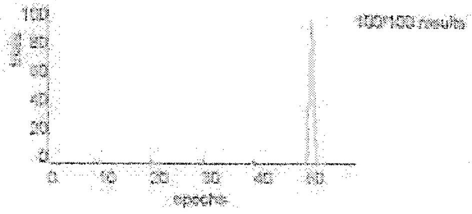

f6clean.jpg
fig. 397

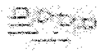


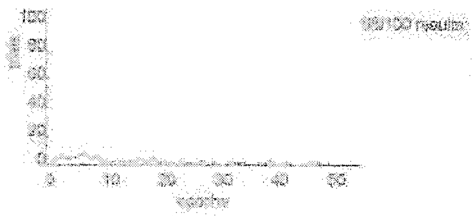


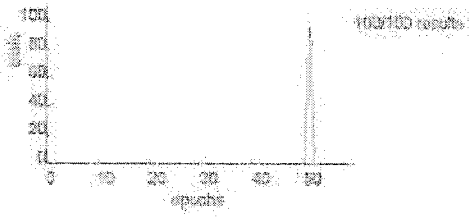

f7clean.jpg
fig. 398

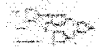


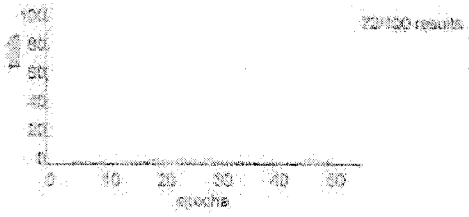


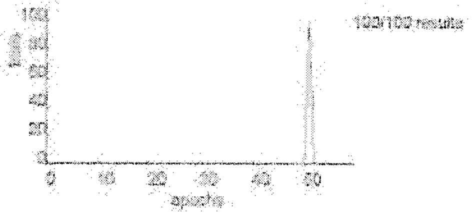

f8clean.jpg
fig. 399






f9clean.jpg
fig. 400





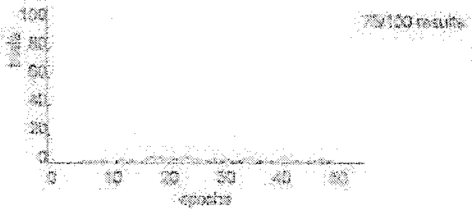
Analysis of Numerous Trials on Each Input Image

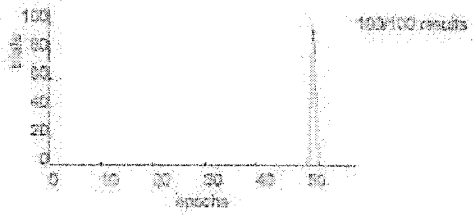
Input Image

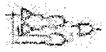
Distribution of First Symbol Found

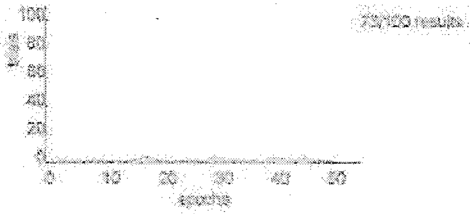
Distribution of Termination

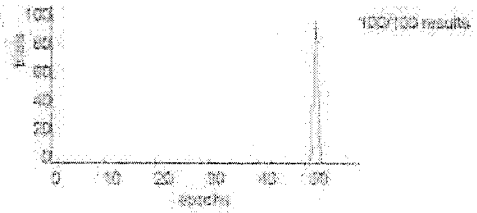
m0clean.jpg
fig. 401



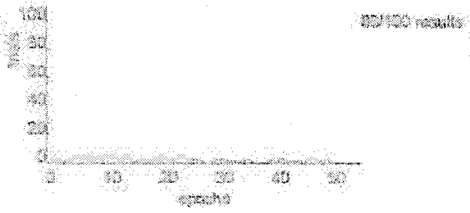


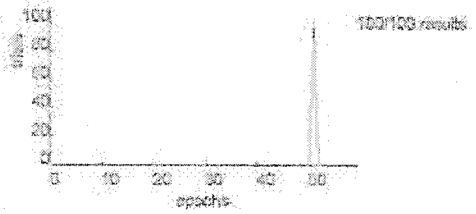

m1clean.jpg
fig. 402

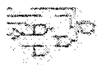


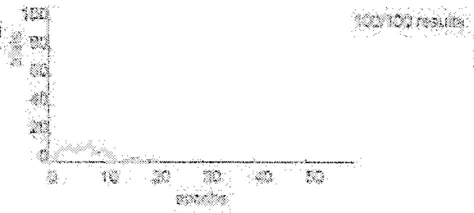


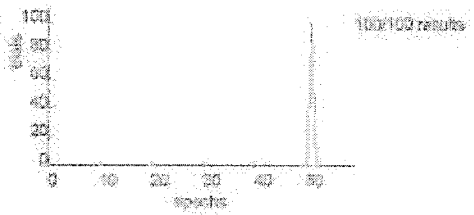
m2clean.jpg
fig. 403

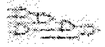


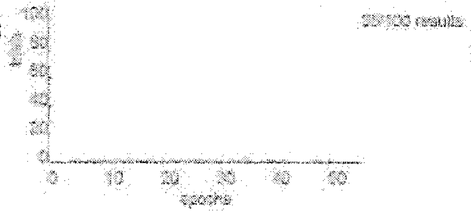


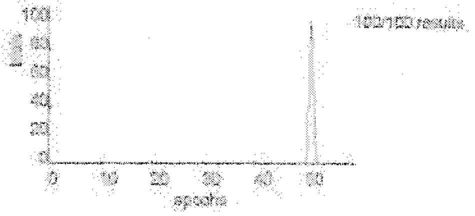

m3clean.jpg
fig. 404






m4clean.jpg
fig. 405





Appendix C:

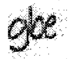
This appendix contains the results for the experiments with various algoirthm configurations, on a strict set of input images. For all experiments, evolution was terminated after 50 epochs. The Appendix is separated into three sections:

- The first seciton presents the accuracy and duration of active evolution.
- The second section presents the epoch of first symbol found, and the epoch of termination data.
- The third section presents the duration of time in seconds for which the algoirhtm was run.

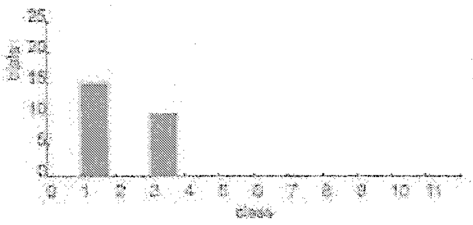
The separation is necessary to present the charts clearly on standard size paper.

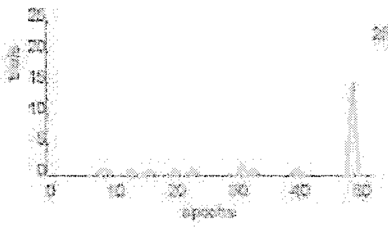
Configuration: Seeded initialization, full parent search, 50 epoch limit, 5 individuals.

Input Image Accuracy Rating Distribution of Active Evolution

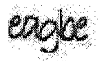


c3.jpg
fig. 0

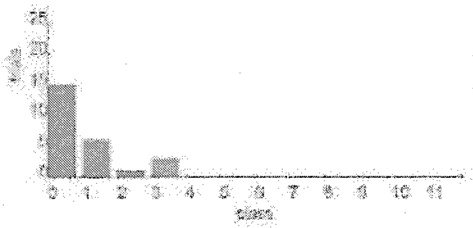


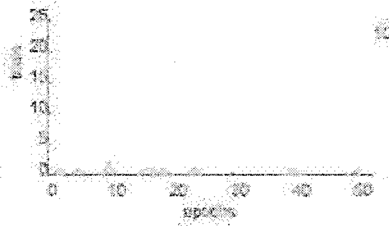


25/25 results

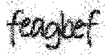


c5.jpg
fig. 1

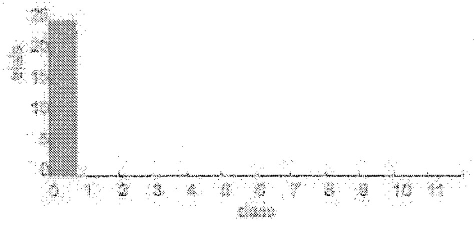


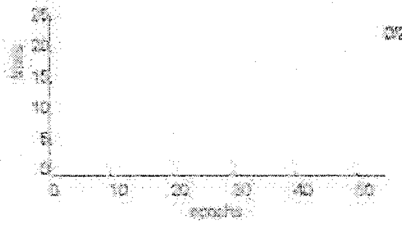


12/25 results

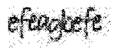


c7.jpg
fig. 2

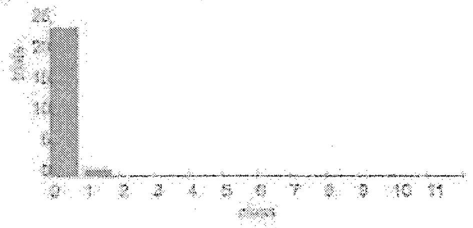


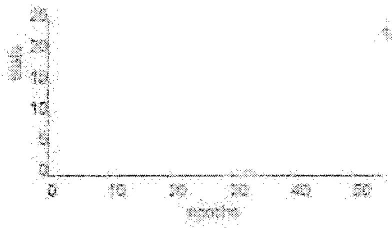


3/25 results

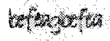


c9.jpg
fig. 3

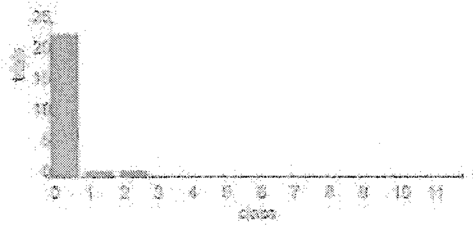


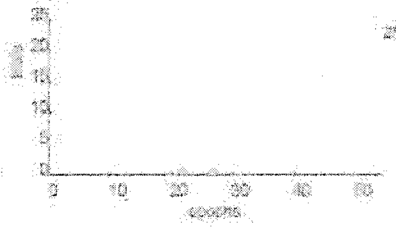


1/25 results



c11.jpg
fig. 4

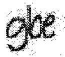




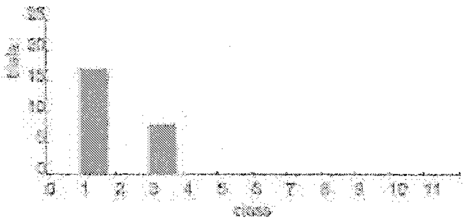
2/25 results

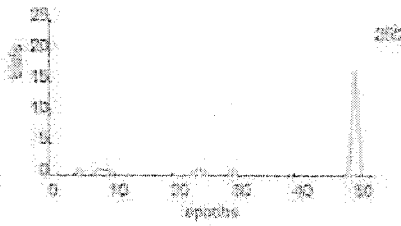
Configuration: Seeded initialization, random parent search, 50 epoch limit, 5 individuals.

Input Image Accuracy Rating Distribution of Active Evolution

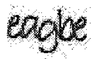


c3.jpg
fig. 5

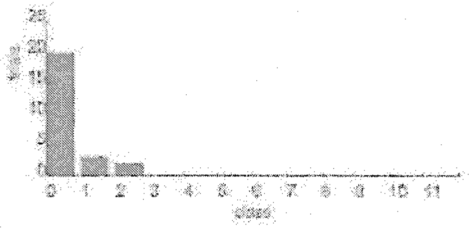


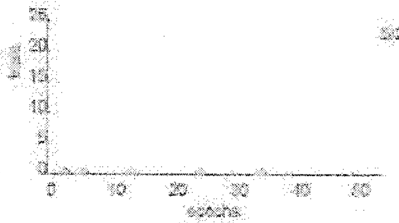


20/25 results

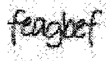


c5.jpg
fig. 6

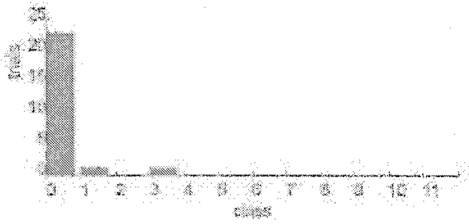


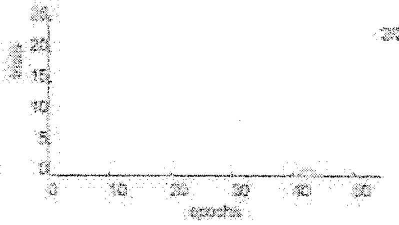


3/25 results

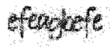


c7.jpg
fig. 7

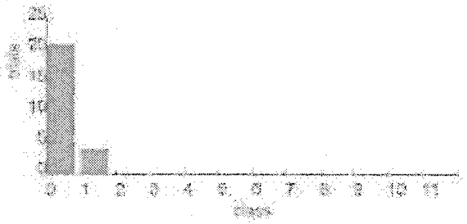


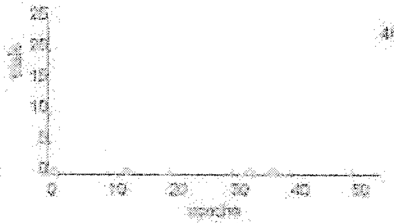


2/25 results

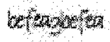


c9.jpg
fig. 8

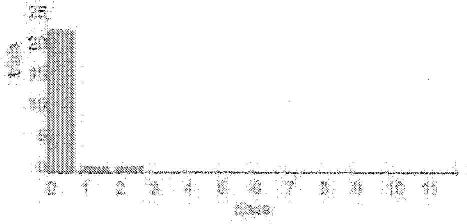


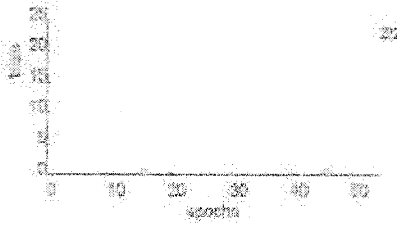


4/25 results



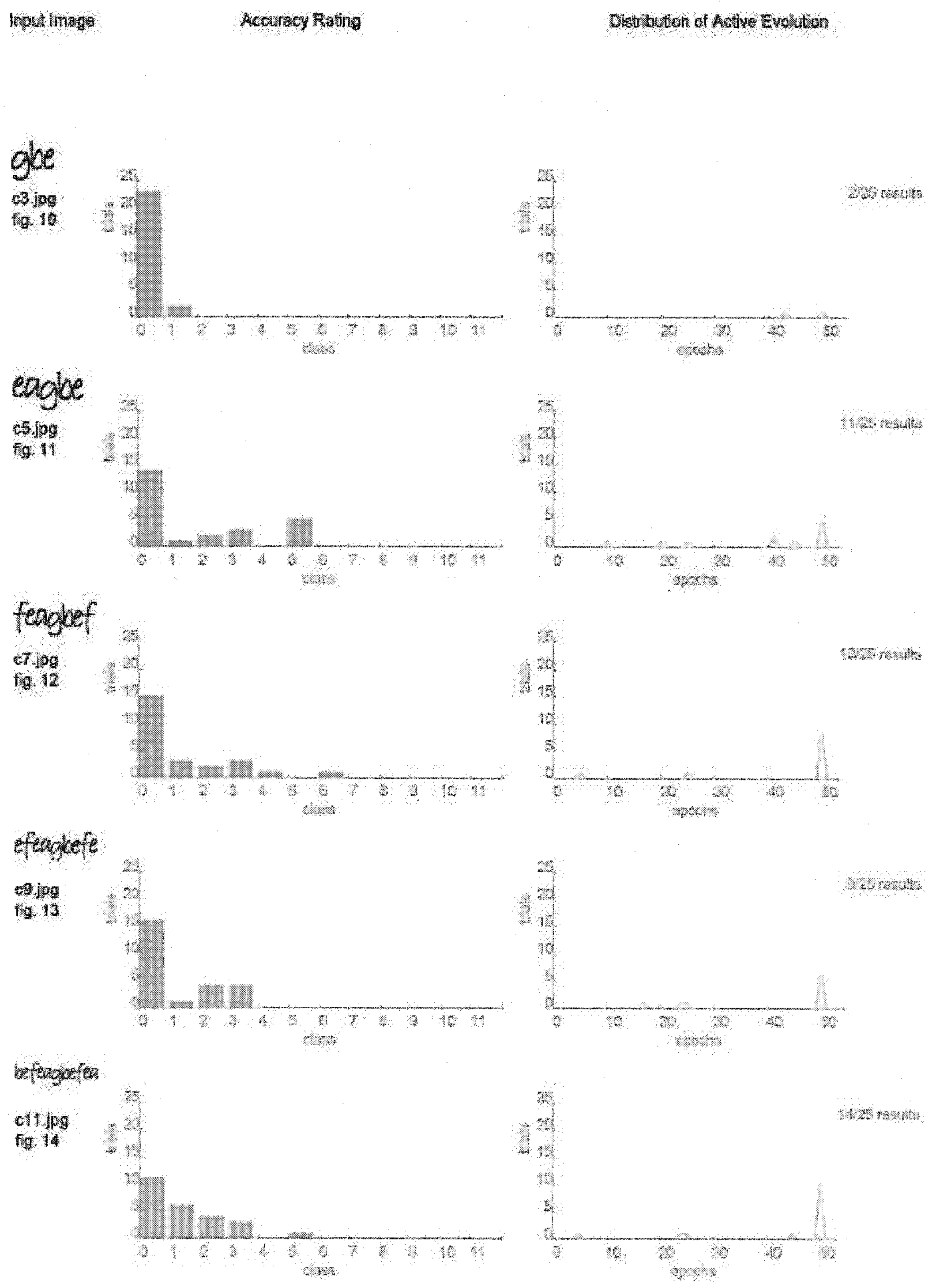
c11.jpg
fig. 9





2/25 results

Configuration: Random initialization, full parent search, 50 epoch limit, 5 individuals.

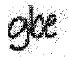


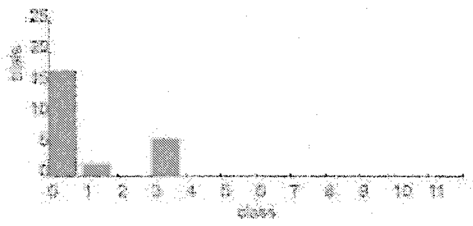
Configuration: Random initialization, random parent search, 50 epoch limit, 5 individuals.

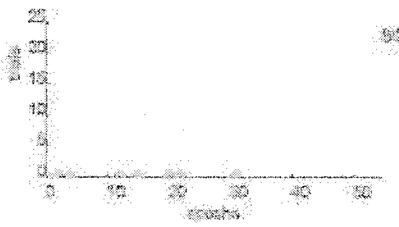
Input Image

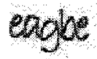
Accuracy Rating

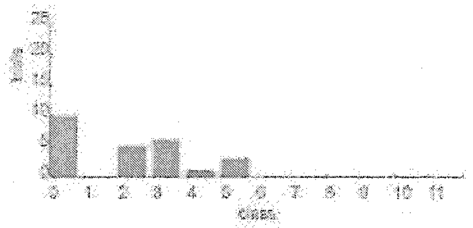
Distribution of Active Evolution

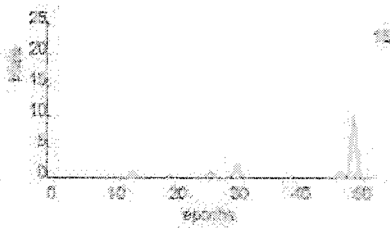

c3.jpg
fig. 15

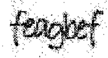


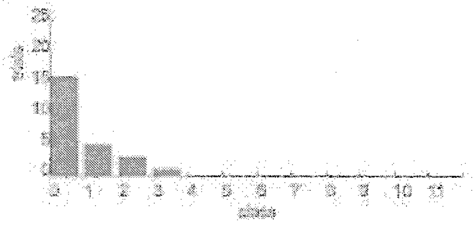


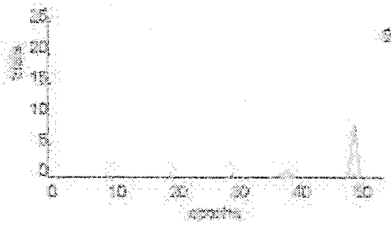

c5.jpg
fig. 16

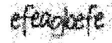


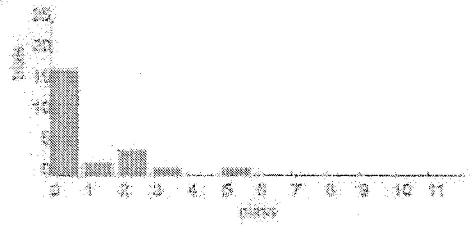


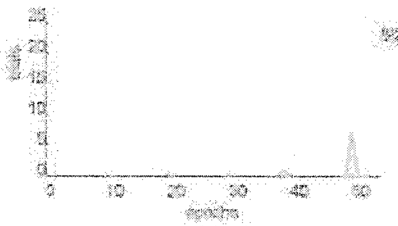

c7.jpg
fig. 17

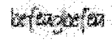


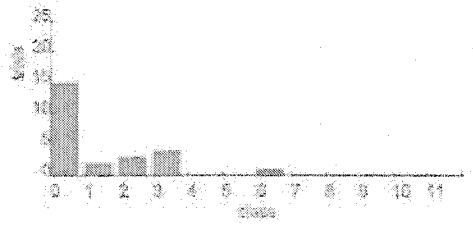


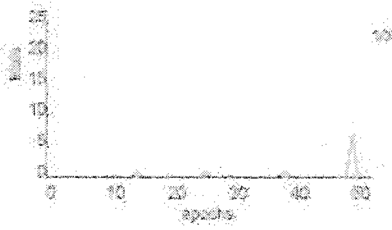

c9.jpg
fig. 18






c11.jpg
fig. 19



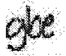


Configuration: Seeded initialization, full parent search, 50 epoch limit, 10 individuals.

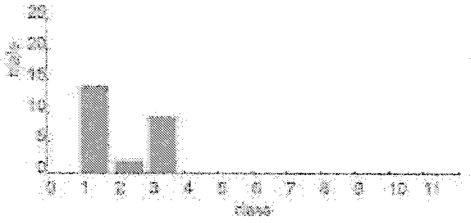
Input Image

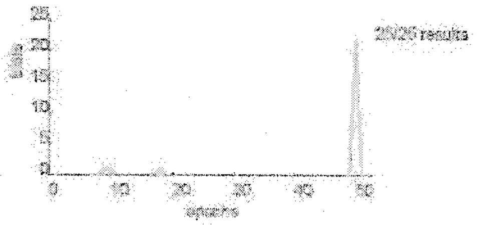
Accuracy Rating

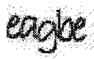
Distribution of Active Evolution



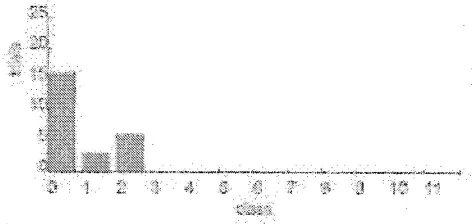
c3.jpg
fig. 20

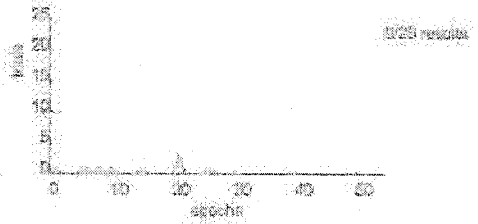


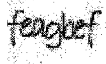




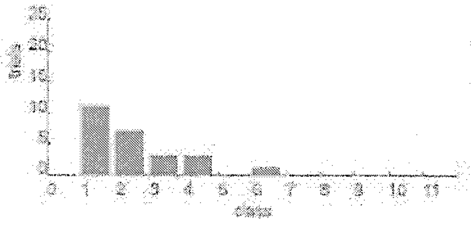
c5.jpg
fig. 21

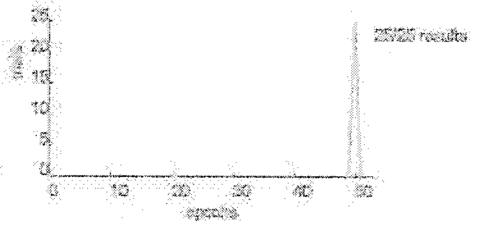


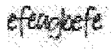




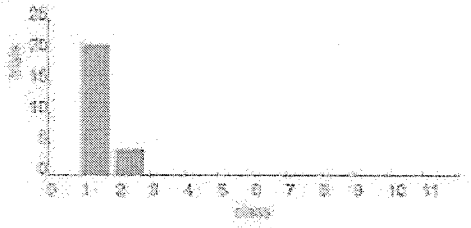
c7.jpg
fig. 22

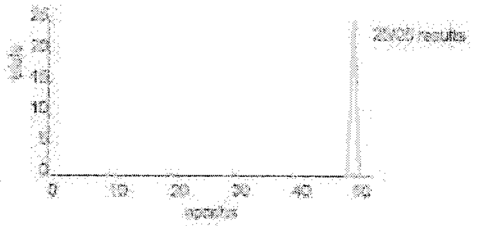


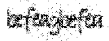




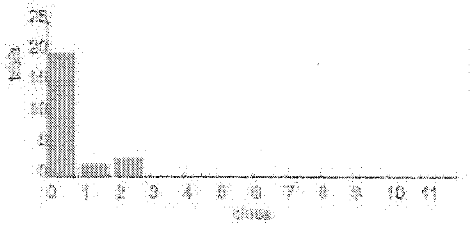
c9.jpg
fig. 23

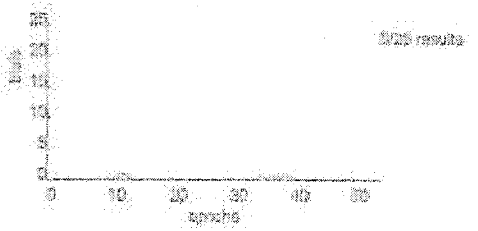






c11.jpg
fig. 24



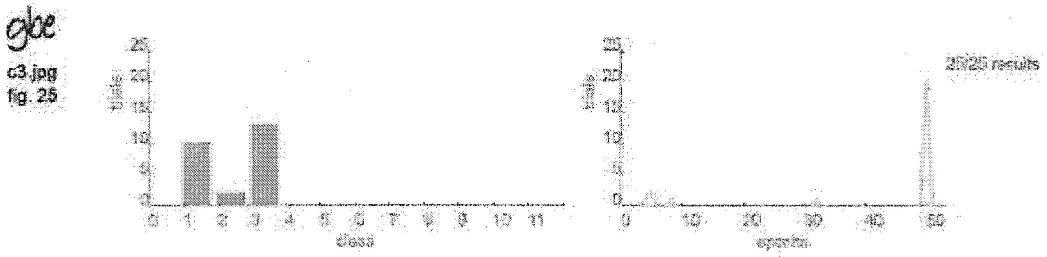


Configuration: Seeded initialization, random parent search, 50 epoch limit, 10 individuals.

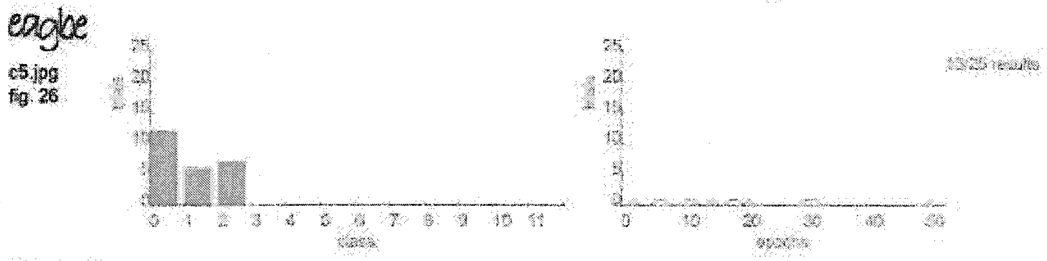
Input Image

Accuracy Rating

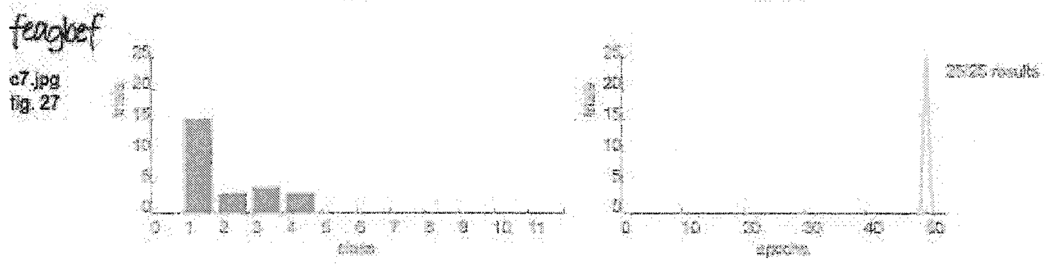
Distribution of Active Evolution



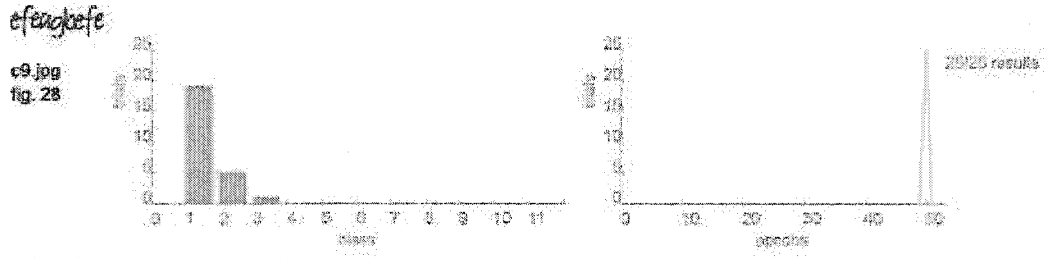
c3.jpg
fig. 25



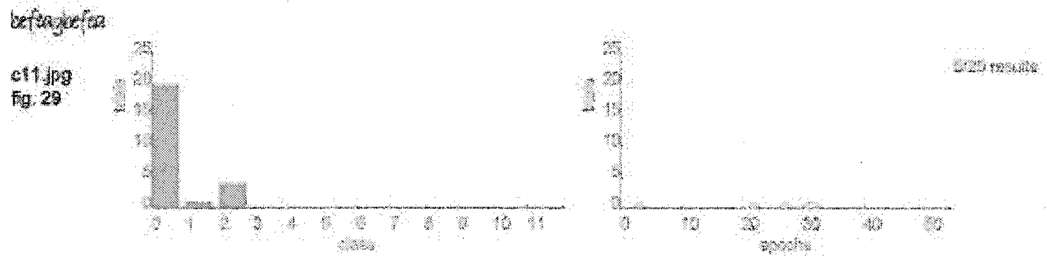
c5.jpg
fig. 26



c7.jpg
fig. 27

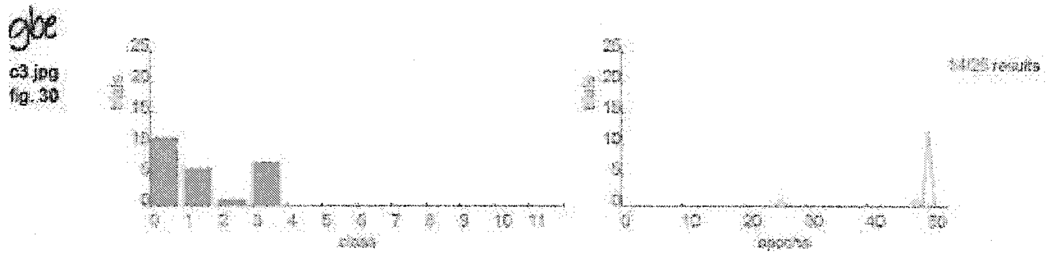


c9.jpg
fig. 28

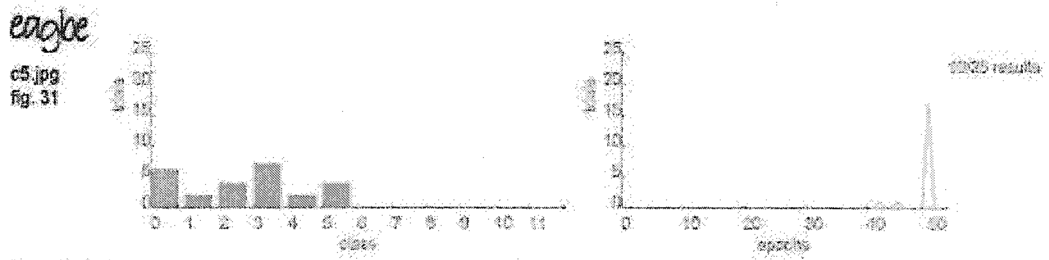


c11.jpg
fig. 29

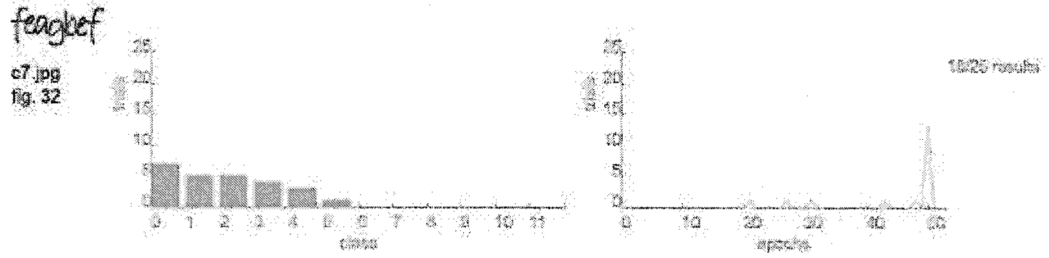
Configuration: Random initialization, full parent search, 50 epoch limit, 10 individuals.



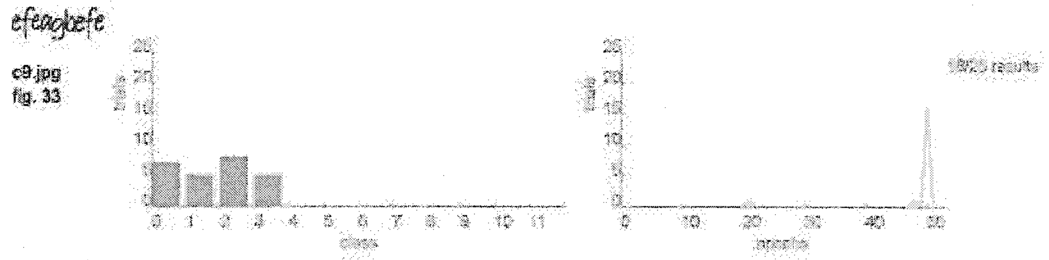
c3.jpg
fig. 30



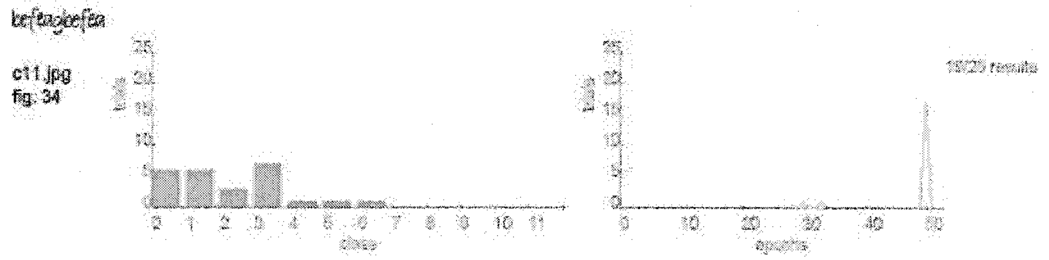
c5.jpg
fig. 31



c7.jpg
fig. 32



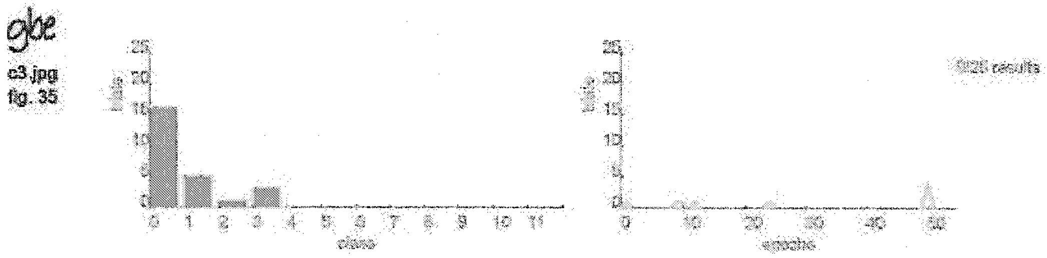
c9.jpg
fig. 33



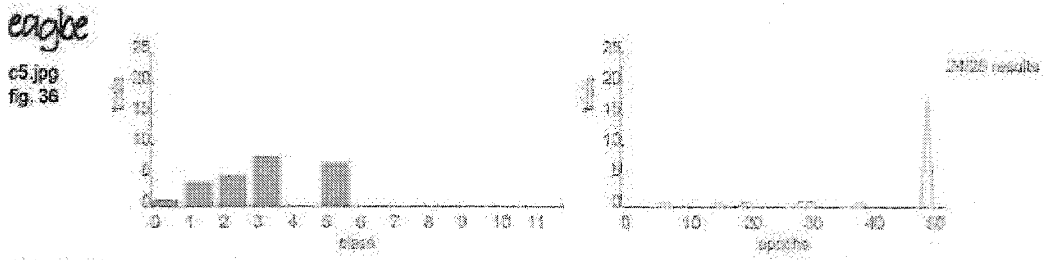
c11.jpg
fig. 34

Configuration: Random initialization, random parent search, 50 epoch limit, 10 individuals.

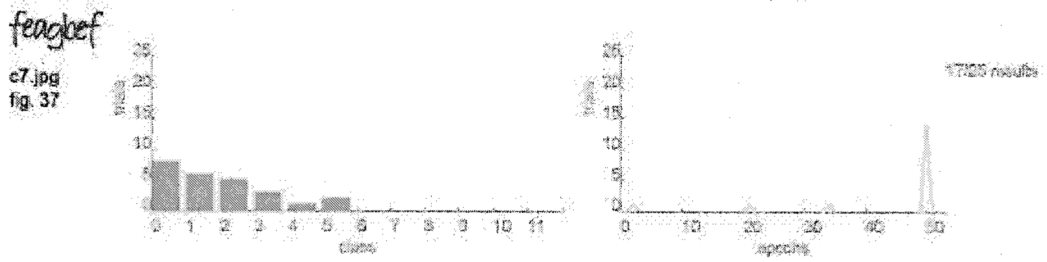
Input Image Accuracy Rating Distribution of Active Evolution



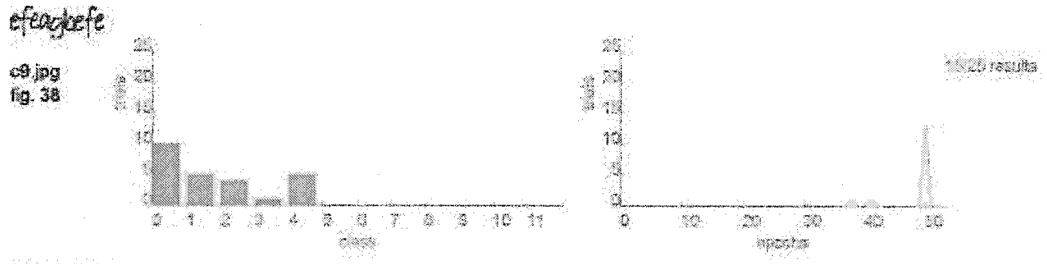
c3.jpg
fig. 35



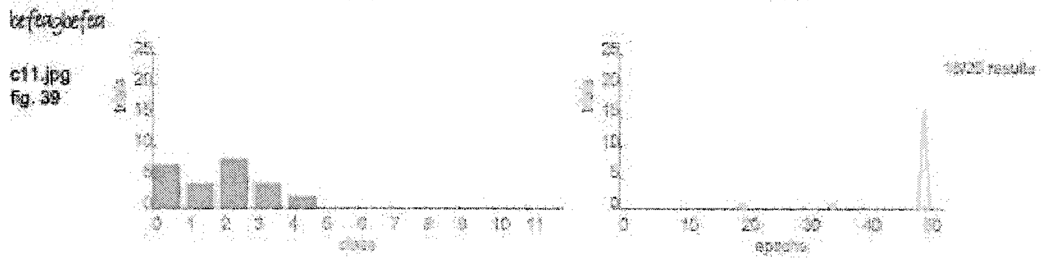
c5.jpg
fig. 36



c7.jpg
fig. 37



c9.jpg
fig. 38



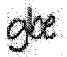
c11.jpg
fig. 39

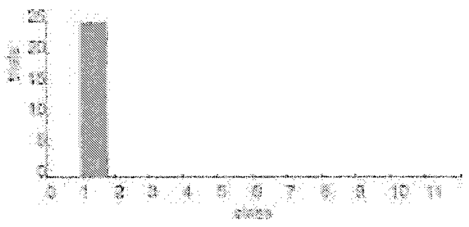
Configuration: Seeded initialization, full parent search, 50 epoch limit, 15 individuals.

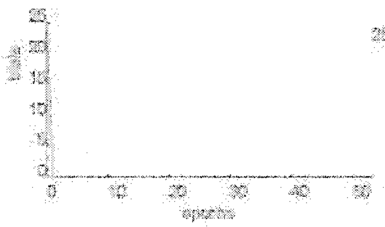
Input Image

Accuracy Rating

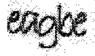
Distribution of Active Evolution

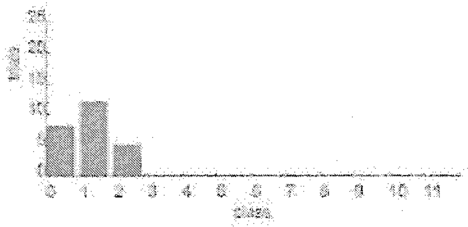

c3.jpg
fig. 40

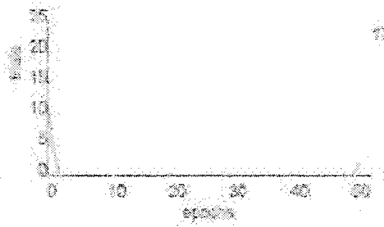




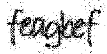
25/25 results

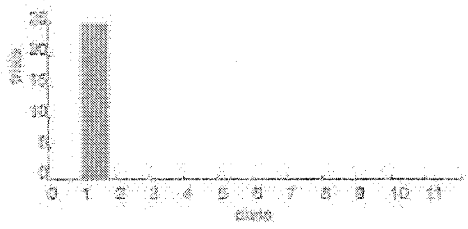

c5.jpg
fig. 41

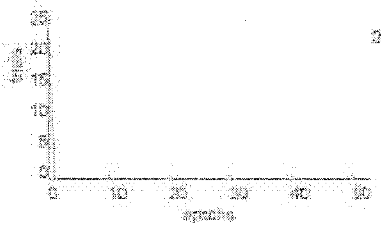




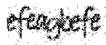
17/25 results

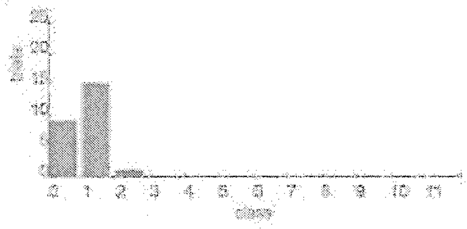

c7.jpg
fig. 42

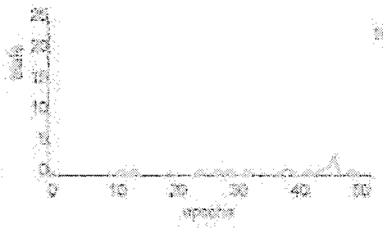




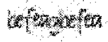
25/25 results

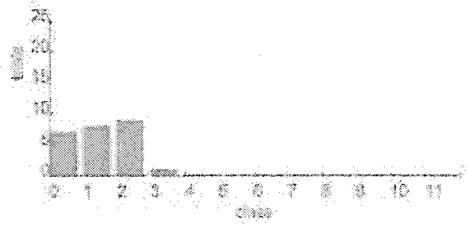

c9.jpg
fig. 43

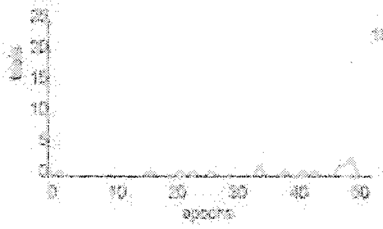




15/25 results


c11.jpg
fig. 44





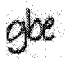
13/25 results

Configuration: Seeded initialization, random parent search, 50 epoch limit, 15 individuals.

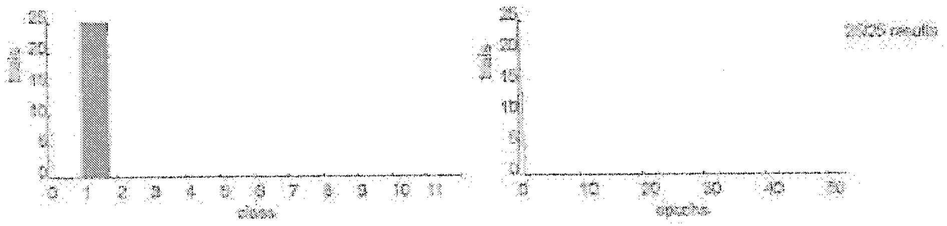
Input Image

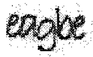
Accuracy Rating

Distribution of Active Evolution

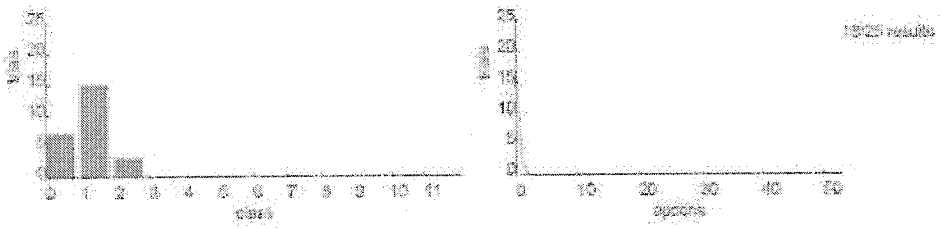


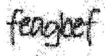
c3.jpg
fig. 45



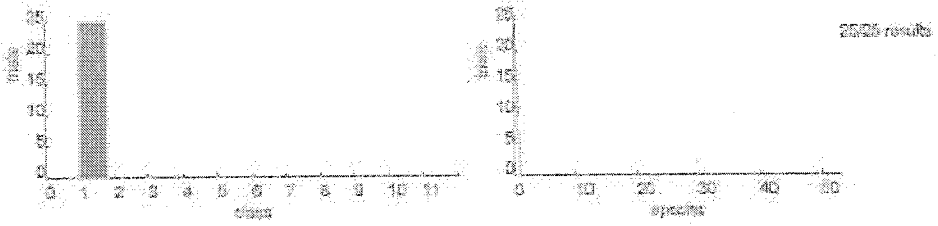


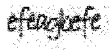
c5.jpg
fig. 46



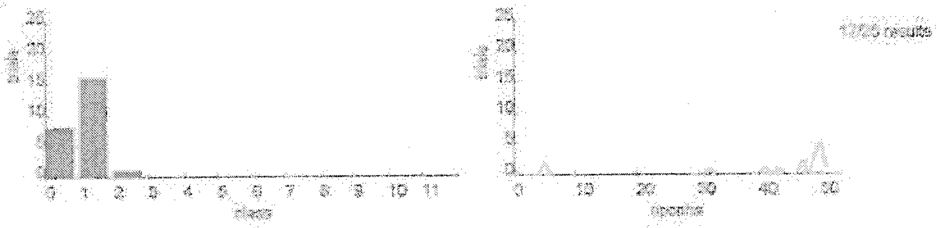


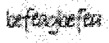
c7.jpg
fig. 47



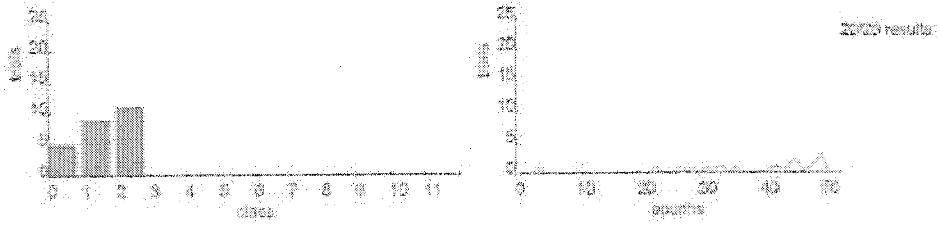


c9.jpg
fig. 48

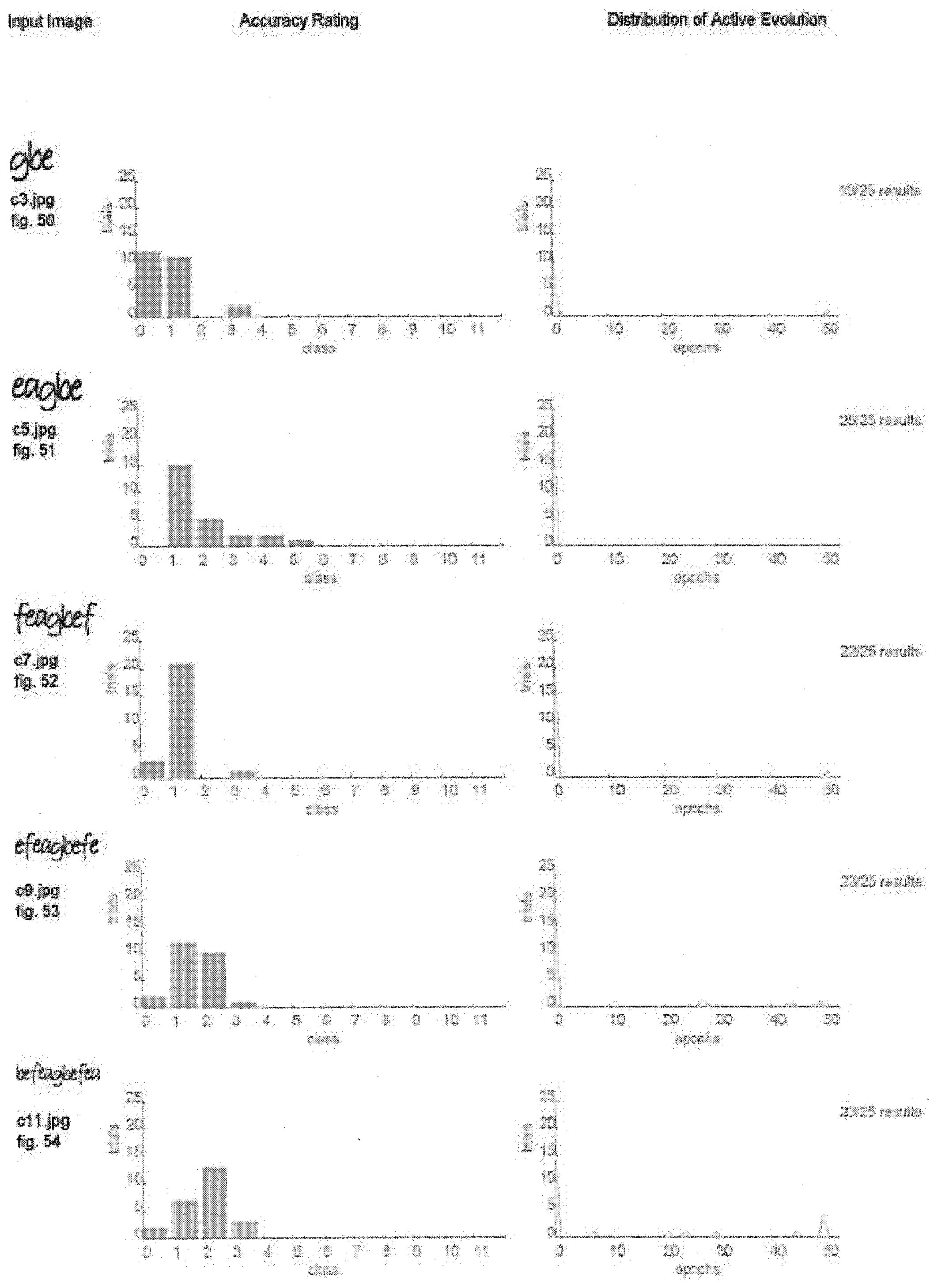




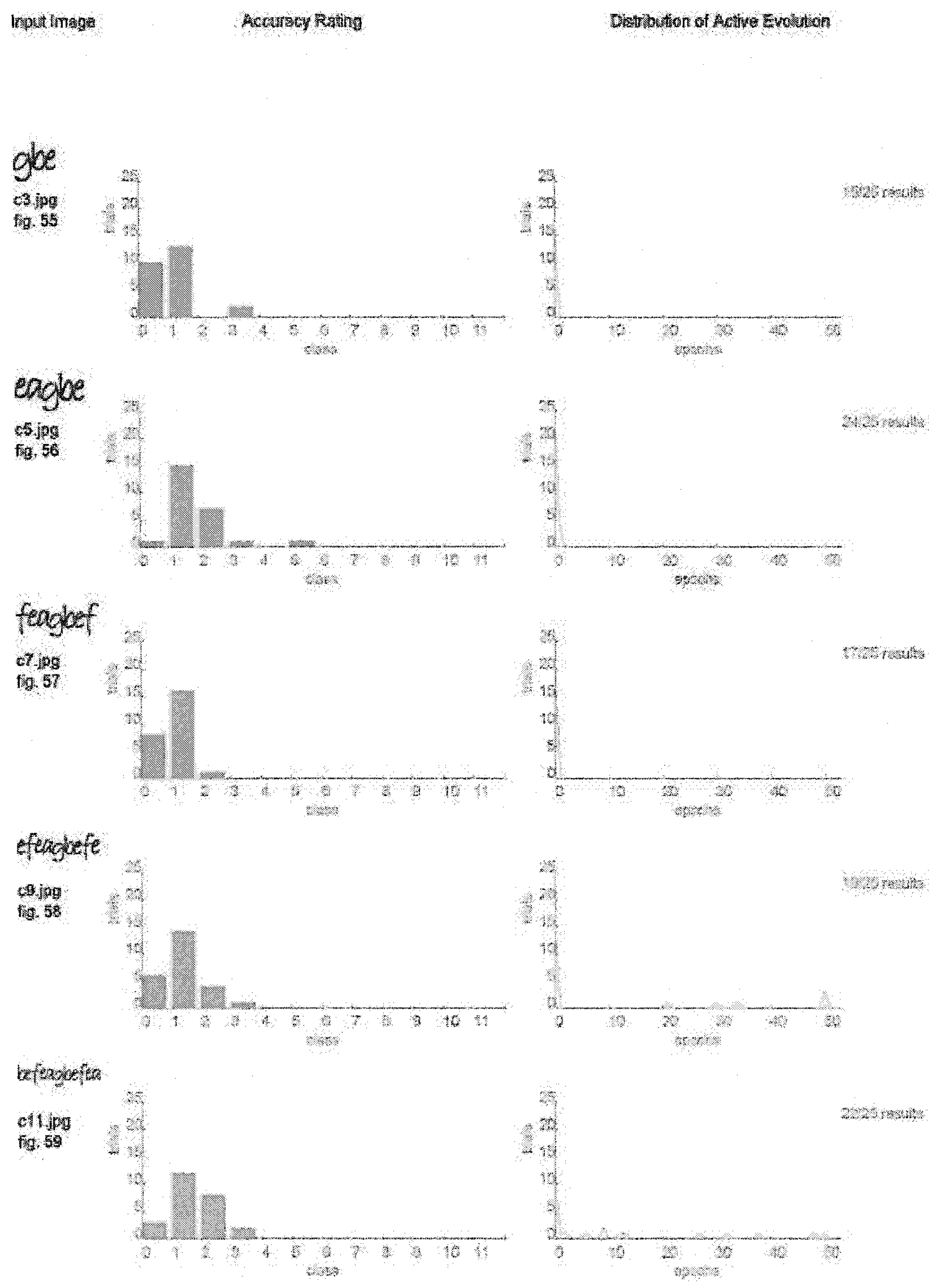
c11.jpg
fig. 49



Configuration: Random initialization, full parent search, 50 epoch limit, 15 individuals.



Configuration: Random initialization, full parent search, 50 epoch limit, 15 individuals.



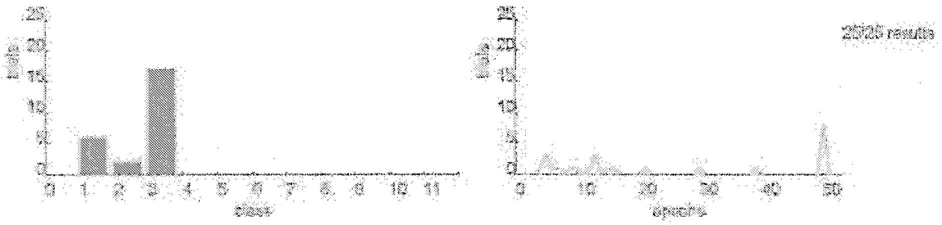
Configuration: Seeded initialization, full parent search, 50 epoch limit, 20 individuals.

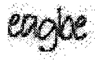
Input Image

Accuracy Rating

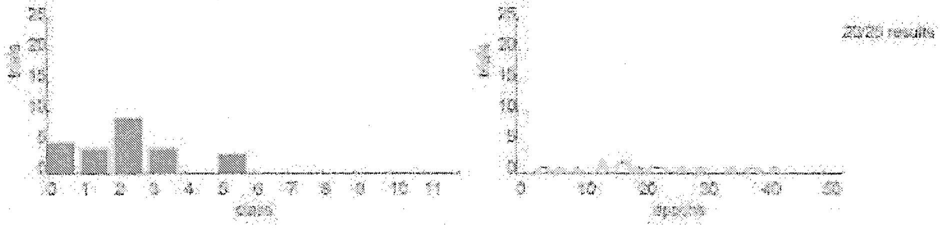
Distribution of Active Evolution

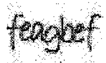
c3.jpg
fig. 60



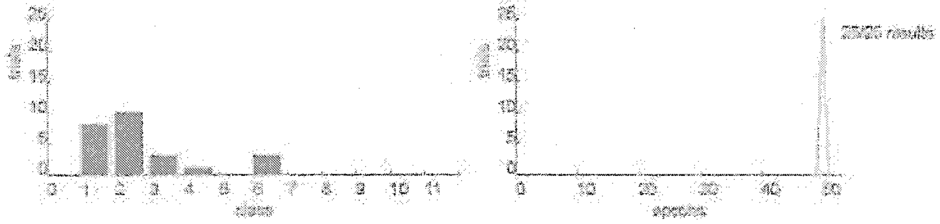


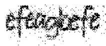
c5.jpg
fig. 61



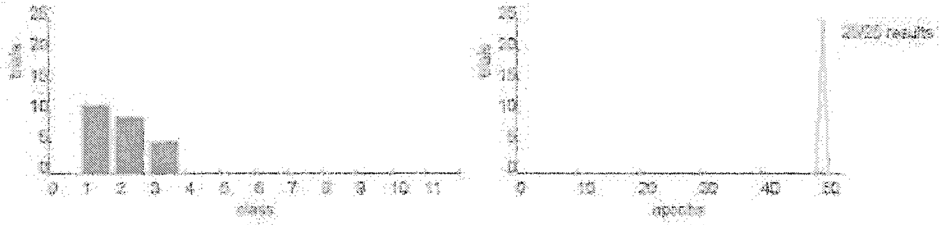


c7.jpg
fig. 62

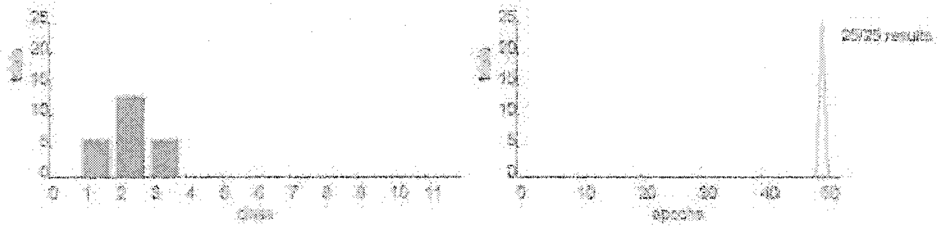




c9.jpg
fig. 63



c11.jpg
fig. 64



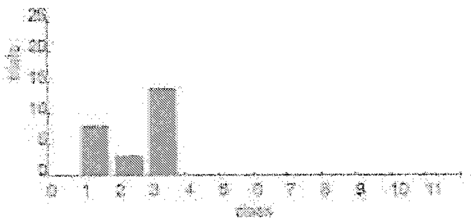
Configuration: Seeded initialization, random parent search, 50 epoch limit, 20 individuals.

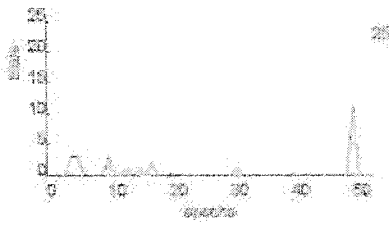
Input Image

Accuracy Rating

Distribution of Active Evolution

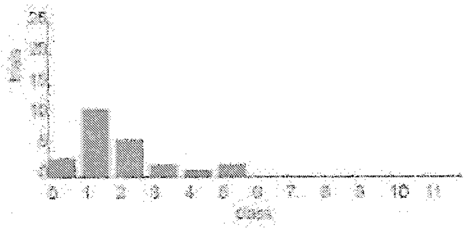
c3.jpg
fig. 65

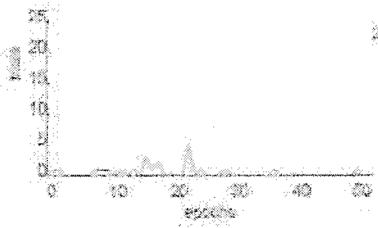




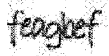
25/25 results

c5.jpg
fig. 66

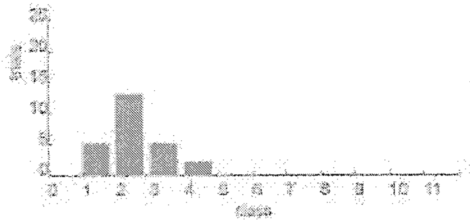


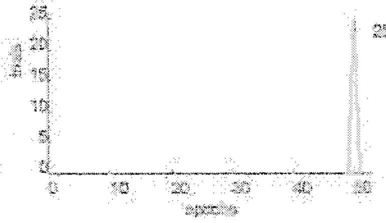


22/25 results

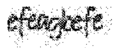


c7.jpg
fig. 67

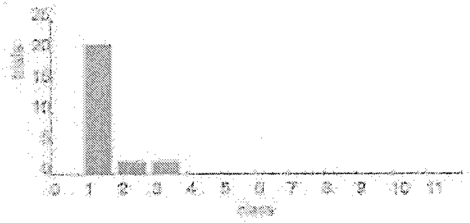


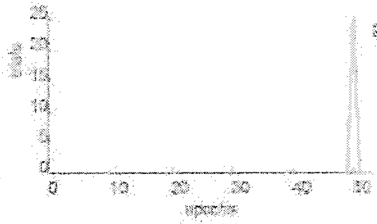


25/25 results

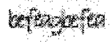


c9.jpg
fig. 68

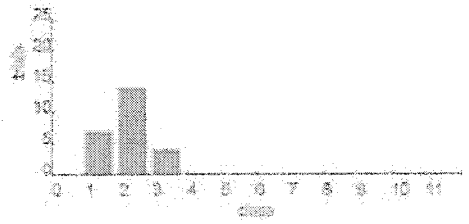


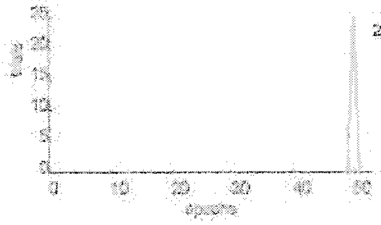


25/25 results



c11.jpg
fig. 69





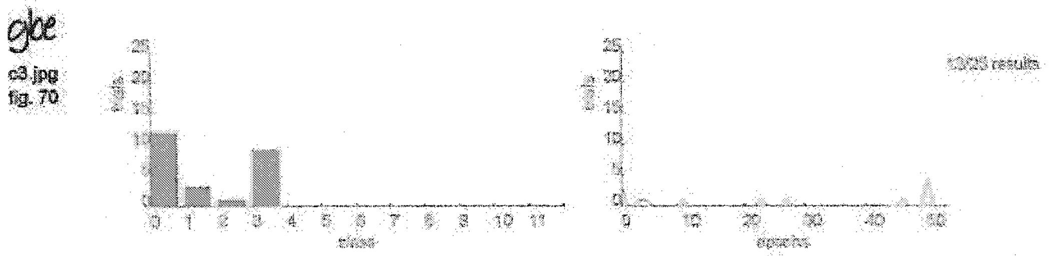
25/25 results

Configuration: Random initialization, full parent search, 50 epoch limit, 20 individuals.

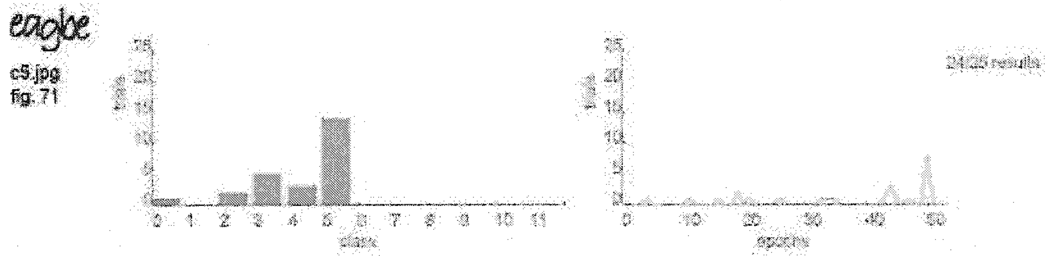
Input Image

Accuracy Rating

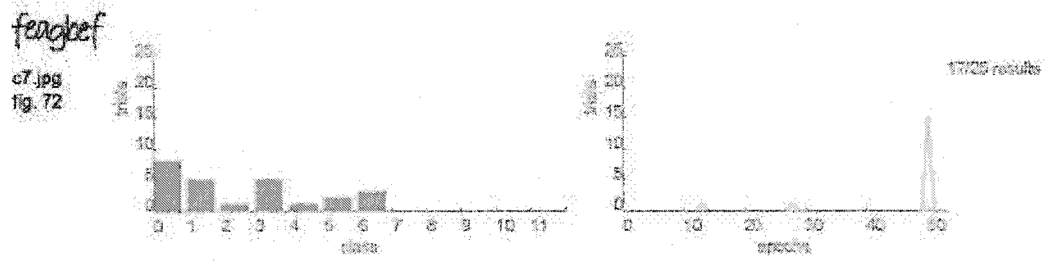
Distribution of Active Evolution



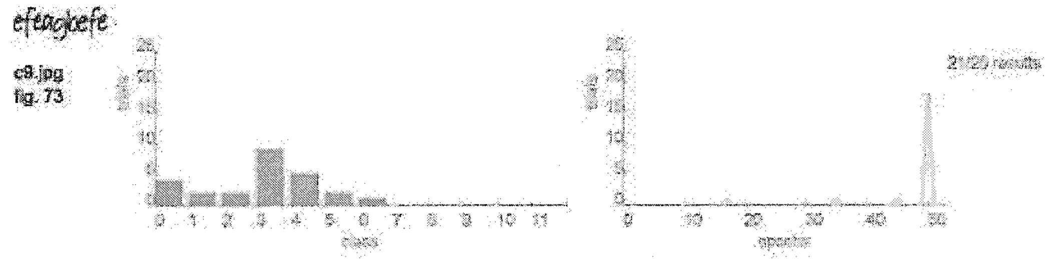
c3.jpg
fig. 70



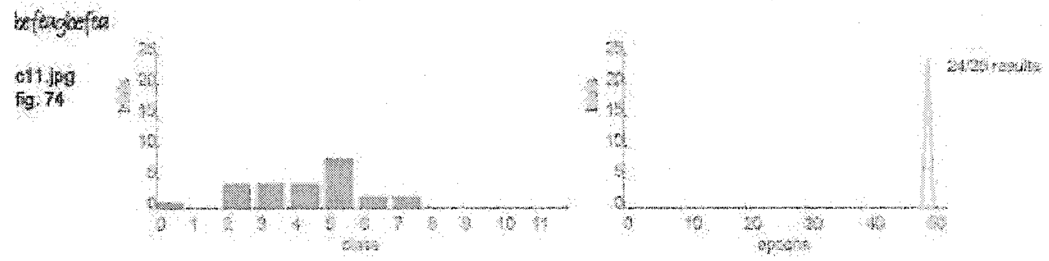
c5.jpg
fig. 71



c7.jpg
fig. 72

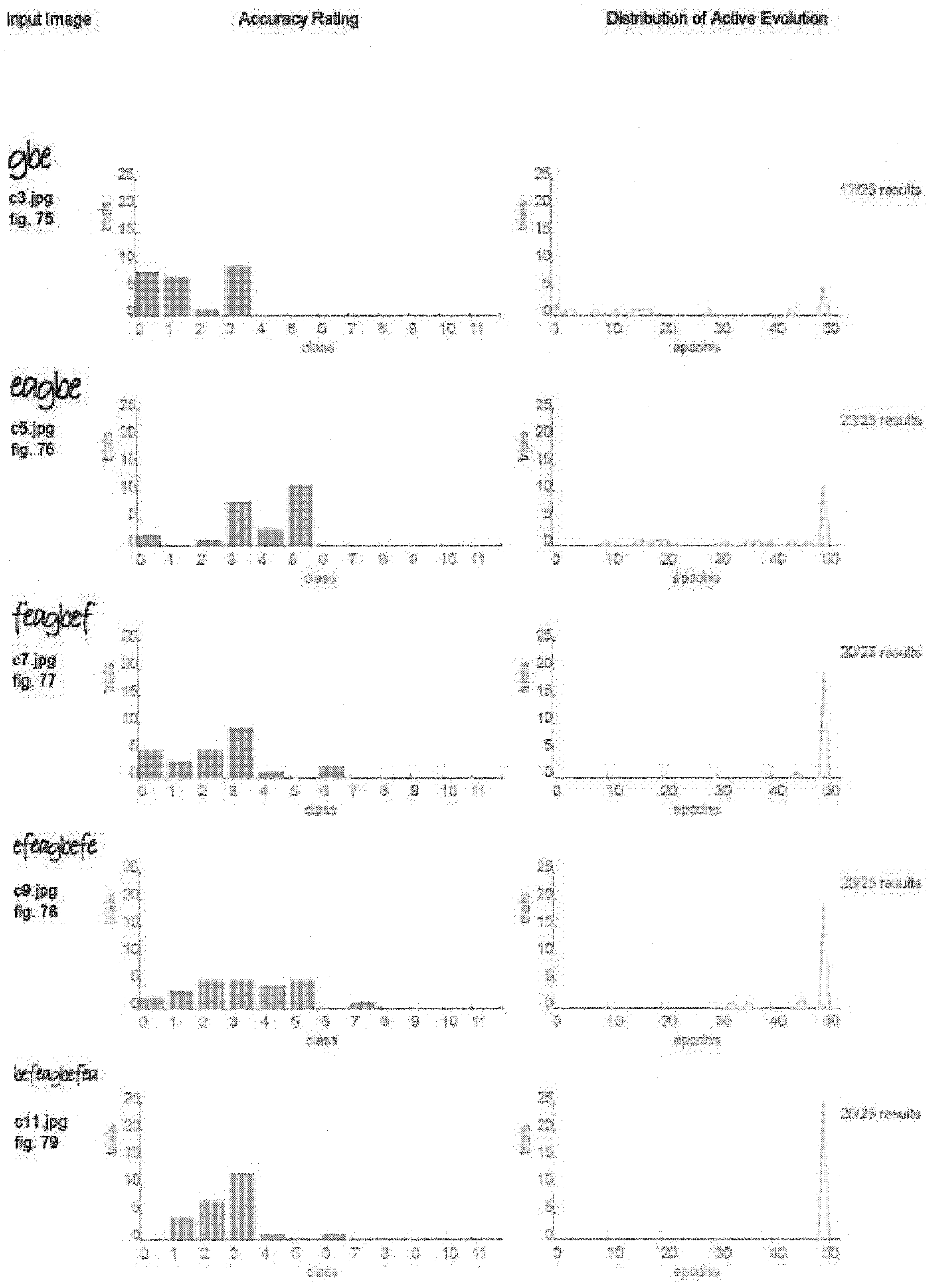


c9.jpg
fig. 73

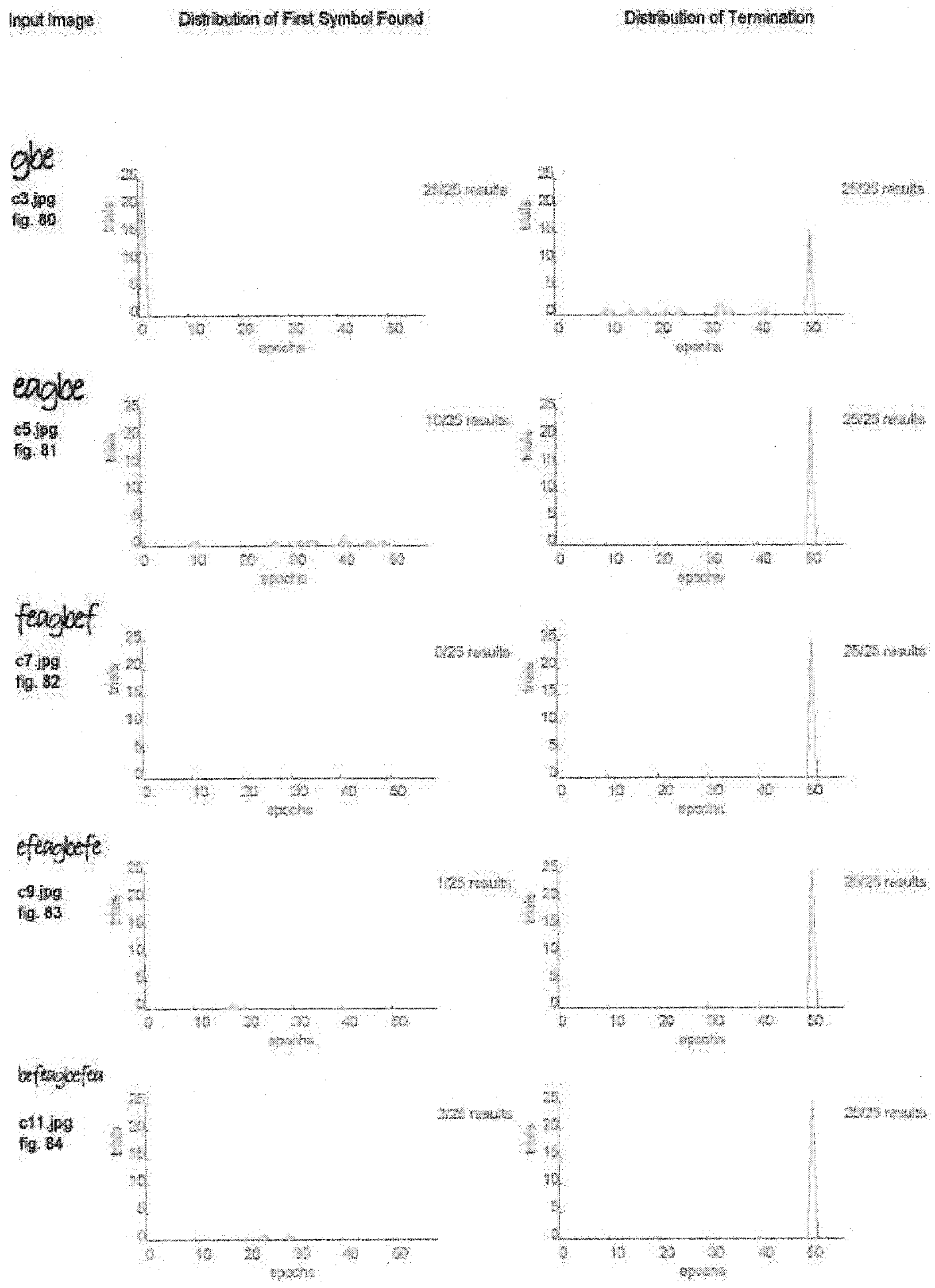


c11.jpg
fig. 74

Configuration: Random initialization, random parent search, 50 epoch limit, 20 individuals.



Configuration: Seeded initialization, full parent search, 50 epoch limit, 5 individuals.

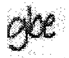


Configuration: Seeded initialization, random parent search, 50 epoch limit, 5 individuals.

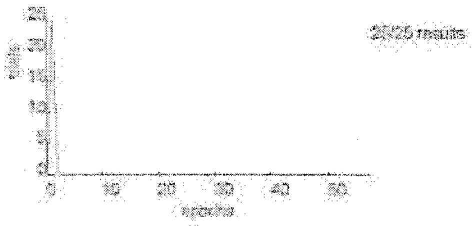
Input Image

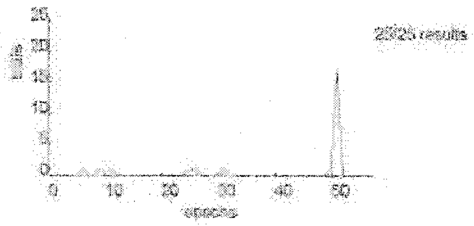
Distribution of First Symbol Found

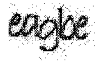
Distribution of Termination



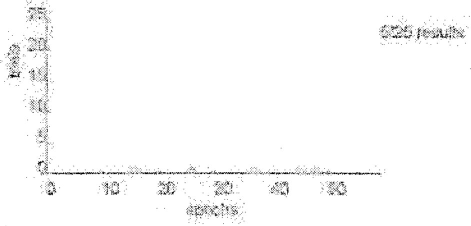
c3.jpg
fig. 85

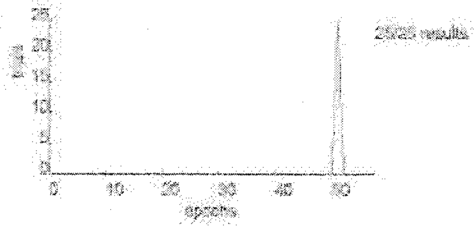


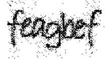




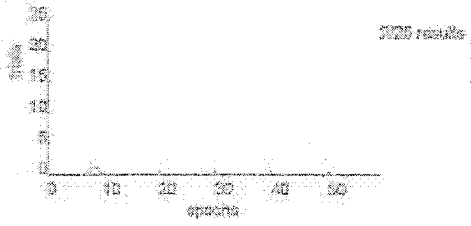
c5.jpg
fig. 86

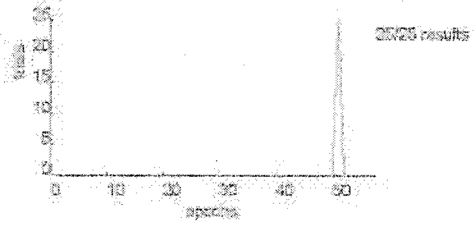


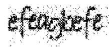




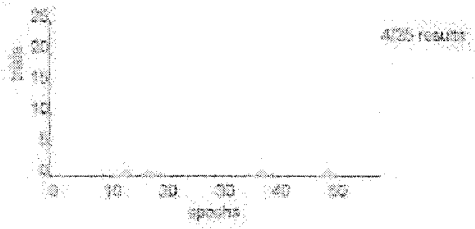
c7.jpg
fig. 87

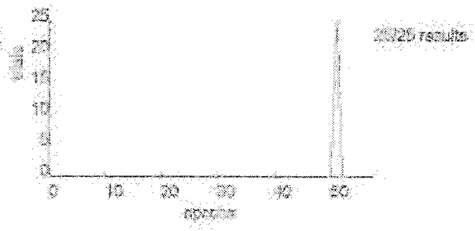


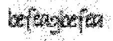




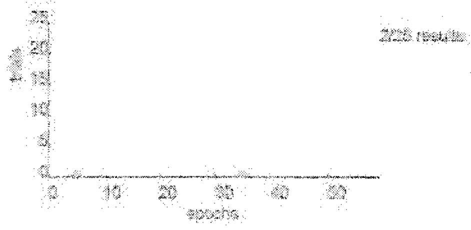
c9.jpg
fig. 88

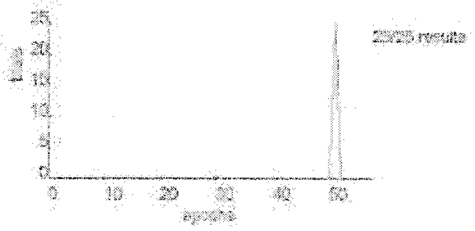






c11.jpg
fig. 89



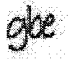


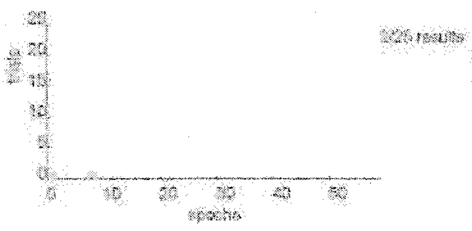
Configuration: Random initialization, full parent search, 50 epoch limit, 5 individuals.

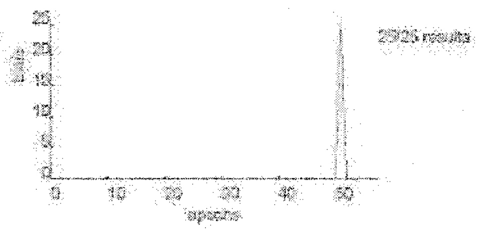
Input image

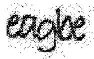
Distribution of First Symbol Found

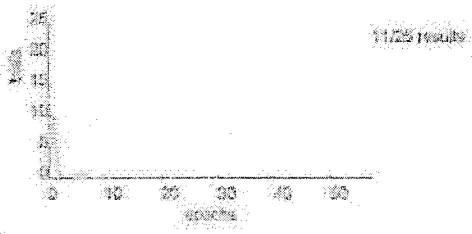
Distribution of Termination

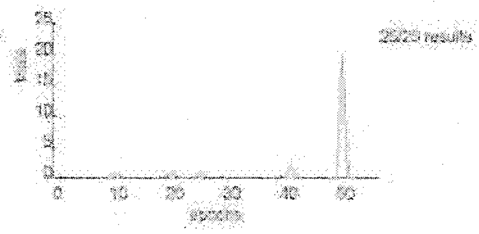

c3.jpg
fig. 90

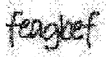


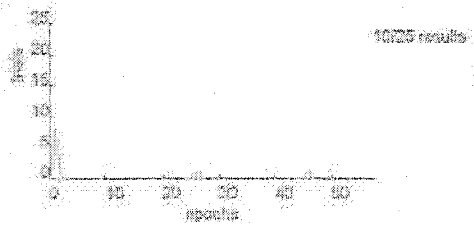


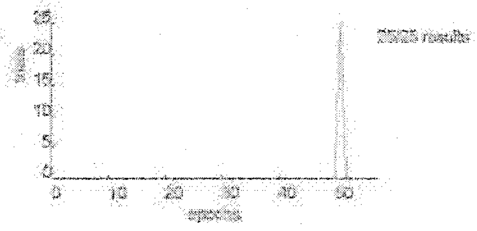

c5.jpg
fig. 91

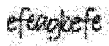


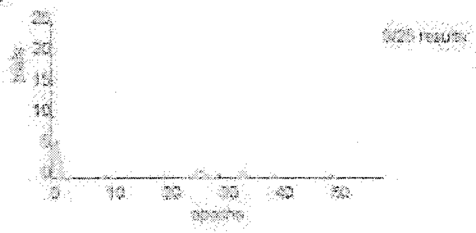


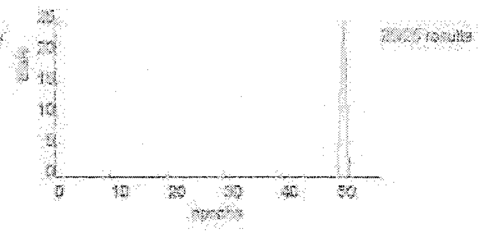

c7.jpg
fig. 92

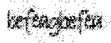


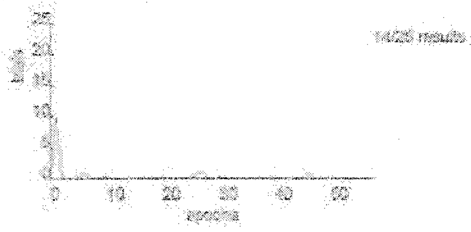


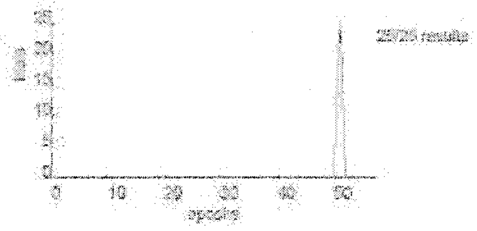

c9.jpg
fig. 93



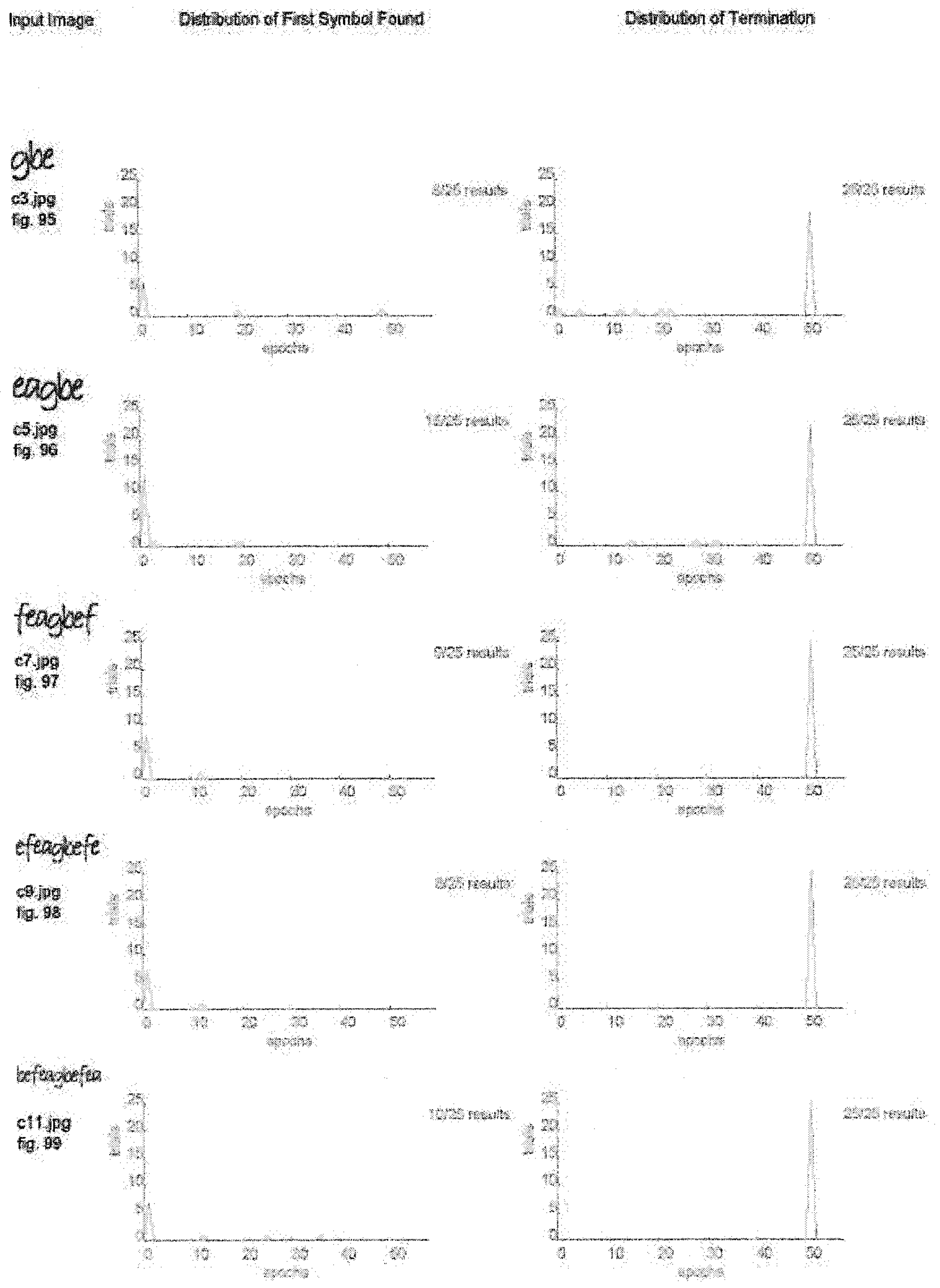



c11.jpg
fig. 94

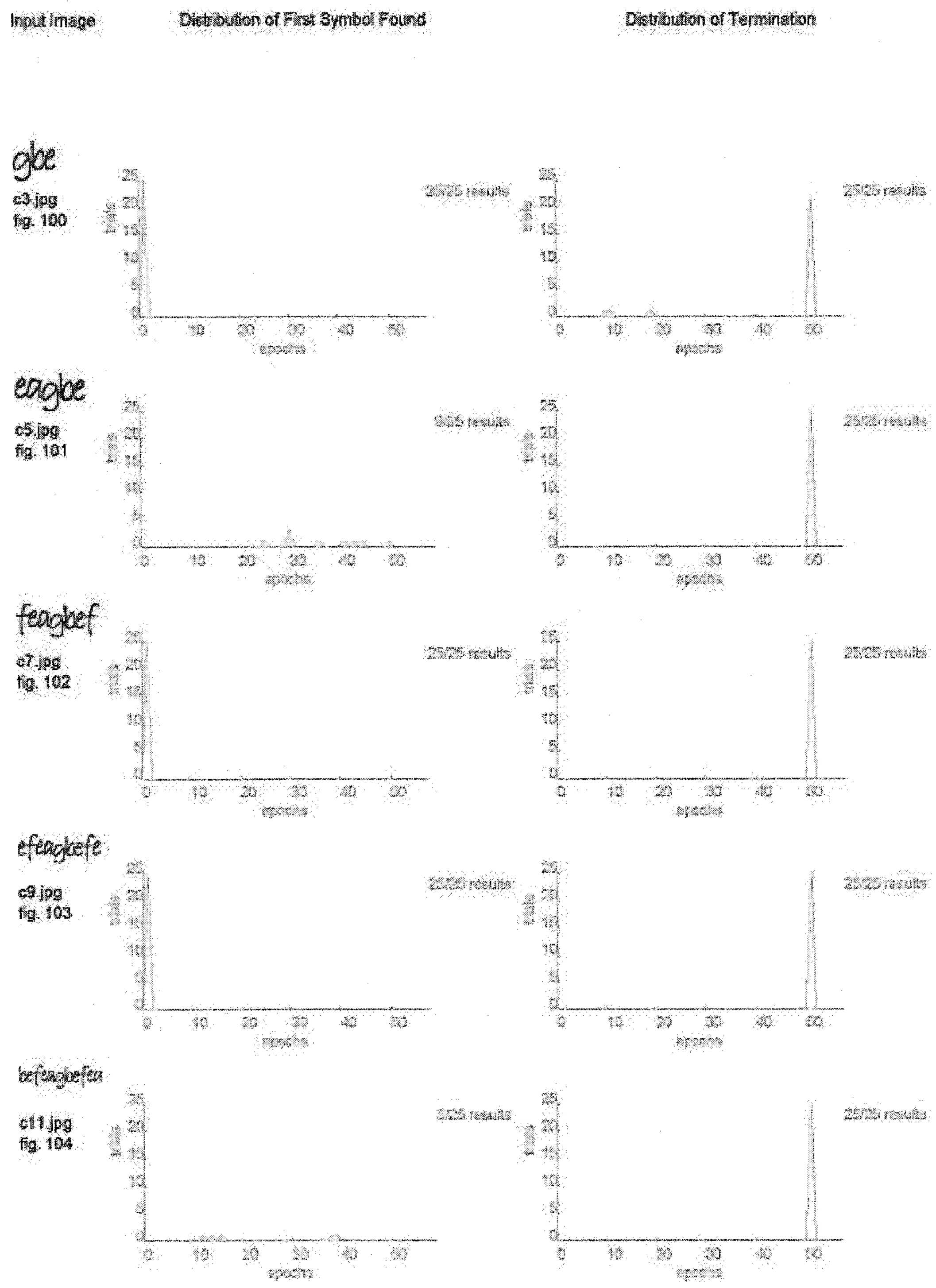




Configuration: Random initialization, random parent search, 50 epoch limit, 5 individuals.



Configuration: Seeded initialization, full parent search, 50 epoch limit, 10 individuals.

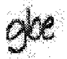


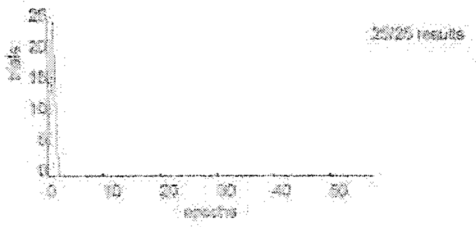
Configuration: Seeded initialization, random parent search, 50 epoch limit, 10 individuals.

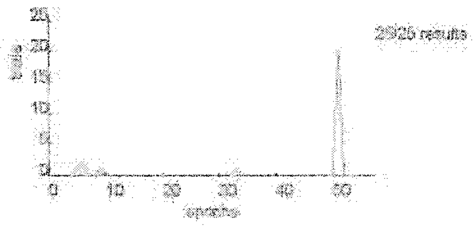
Input Image

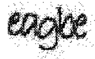
Distribution of First Symbol Found

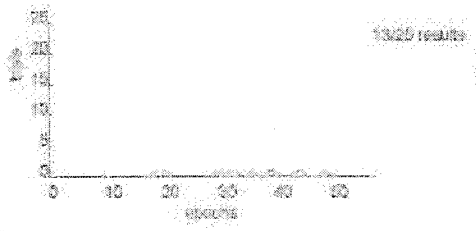
Distribution of Termination

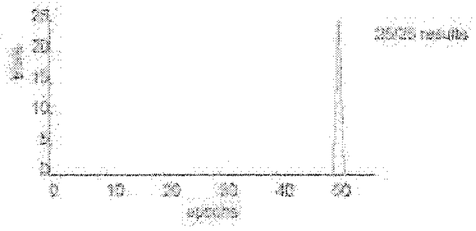

c3.jpg
fig. 105

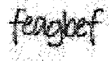


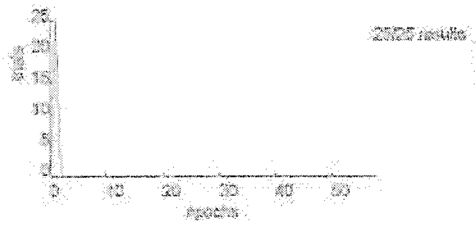


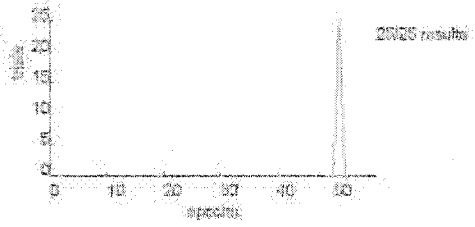

c5.jpg
fig. 106

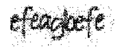


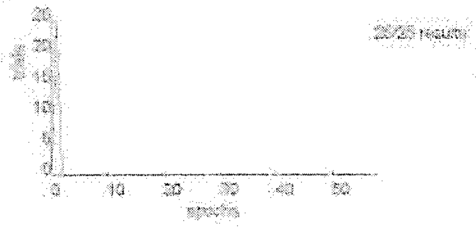


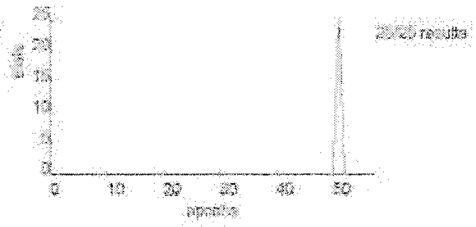

c7.jpg
fig. 107

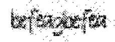


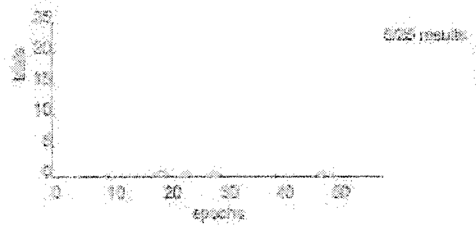


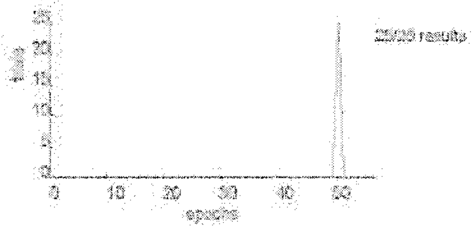

c9.jpg
fig. 108



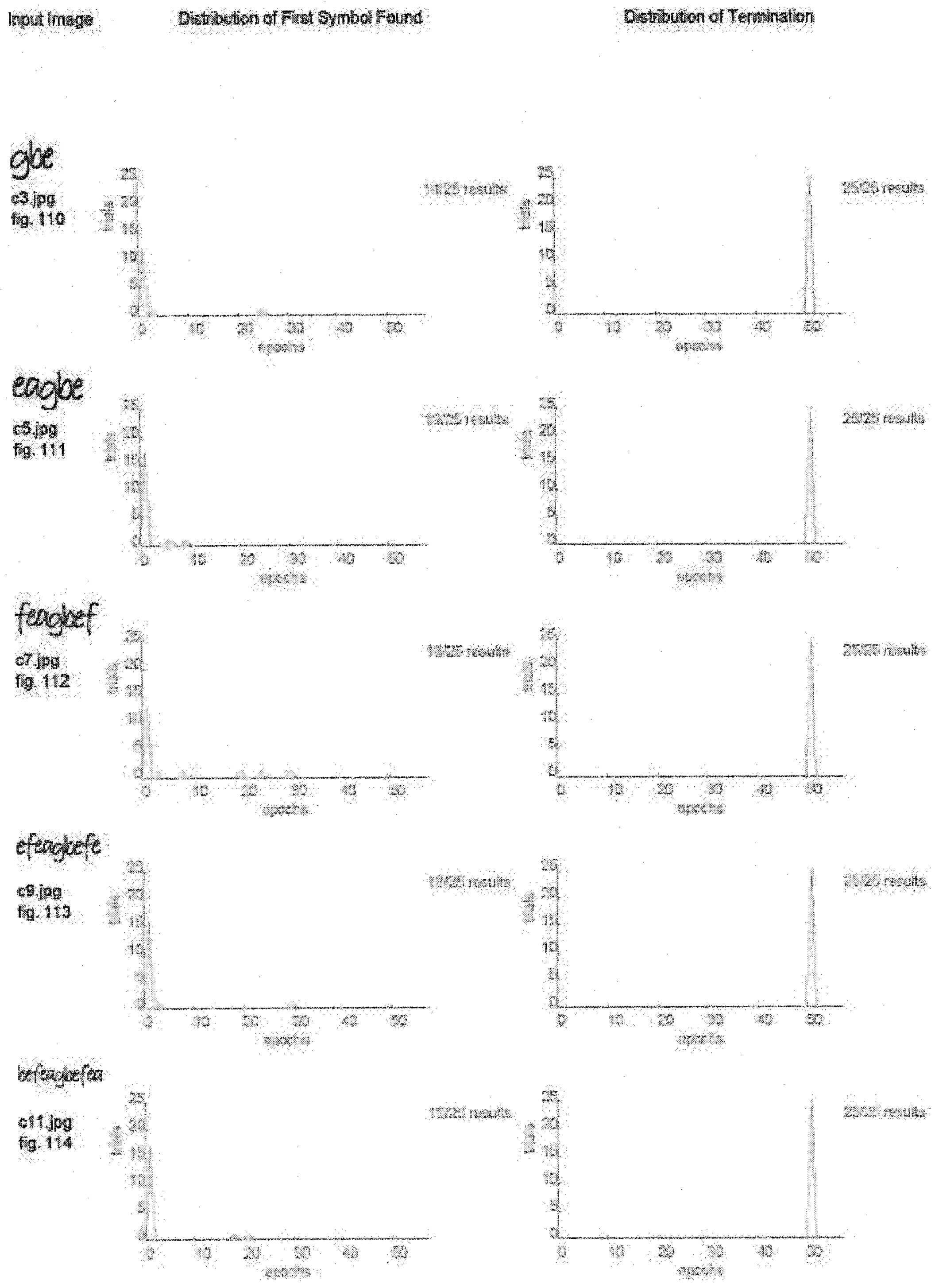



c11.jpg
fig. 109





Configuration: Random initialization, full parent search, 50 epoch limit, 10 individuals.

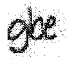


Configuration: Random initialization, random parent search, 50 epoch limit, 10 individuals.

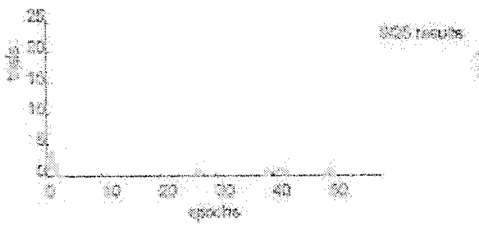
Input Image

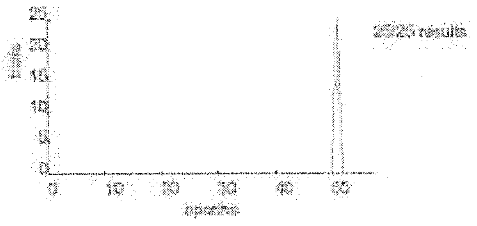
Distribution of First Symbol Found

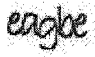
Distribution of Termination



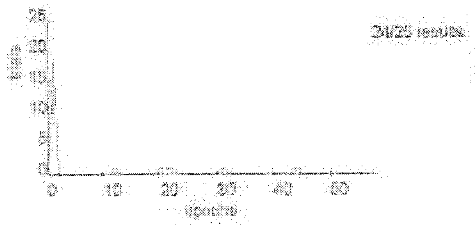
c3.jpg
fig. 115

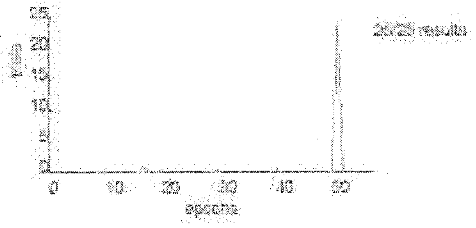


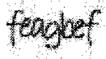




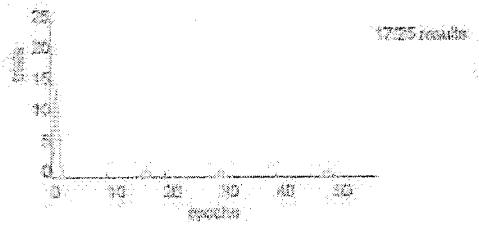
c5.jpg
fig. 116

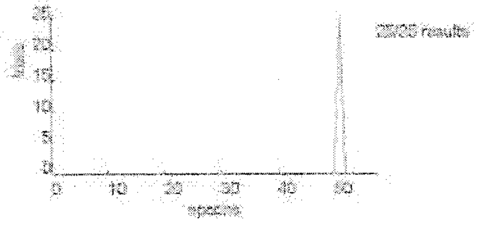


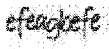




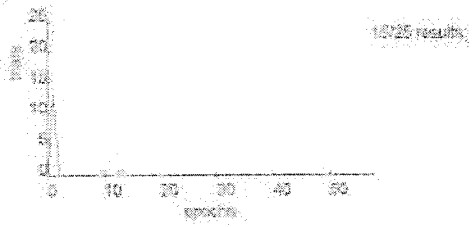
c7.jpg
fig. 117

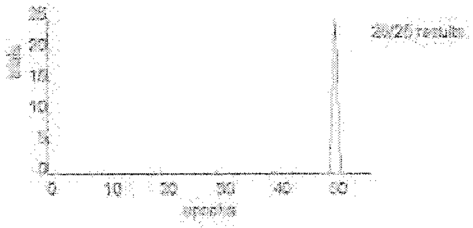


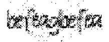




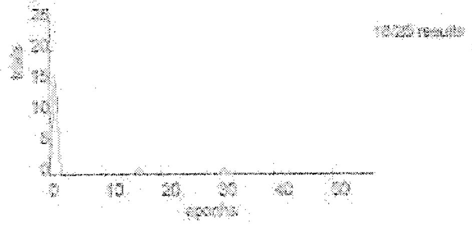
c9.jpg
fig. 118

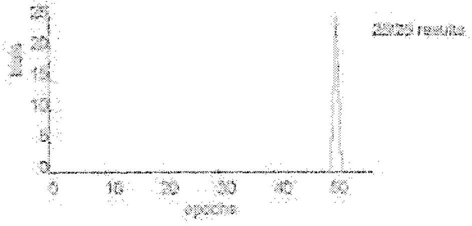






c11.jpg
fig. 119



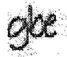


Configuration: Seeded initialization, full parent search, 50 epoch limit, 15 individuals.

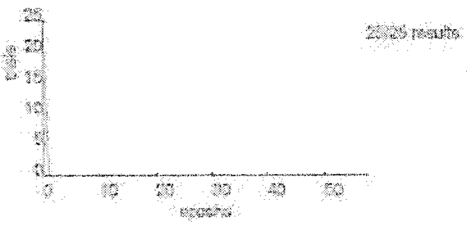
Input Image

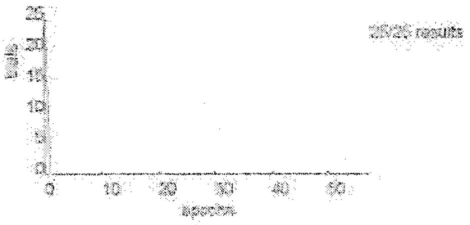
Distribution of First Symbol Found

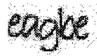
Distribution of Termination



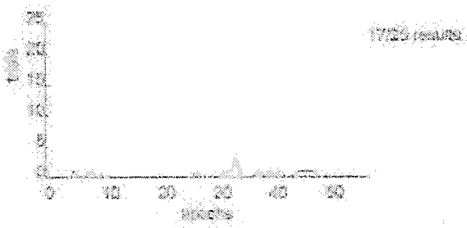
c3.jpg
fig. 120

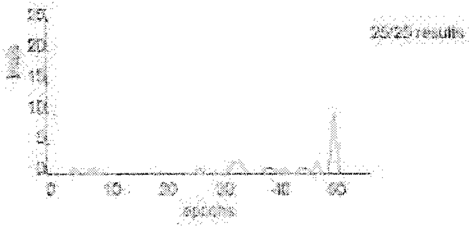


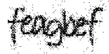




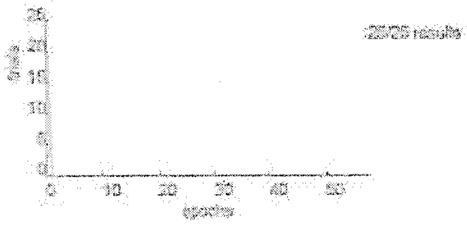
c5.jpg
fig. 121

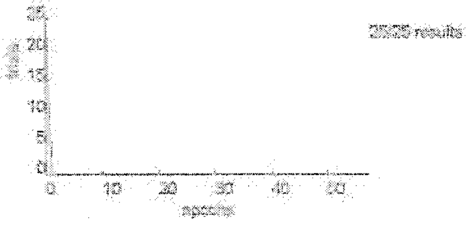


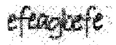




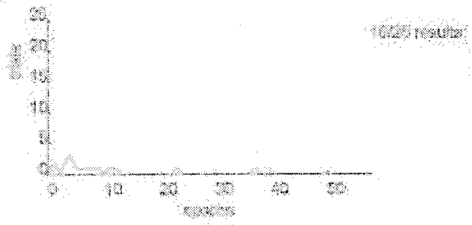
c7.jpg
fig. 122

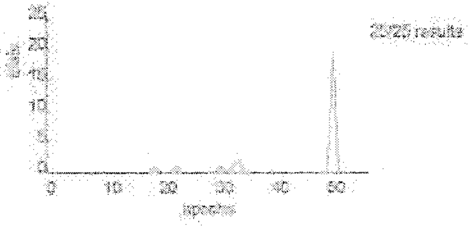


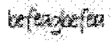




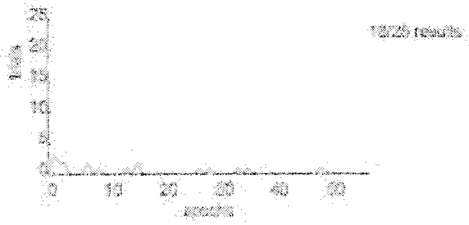
c9.jpg
fig. 123

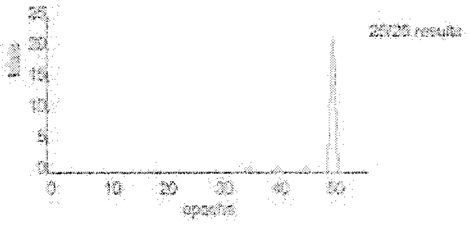




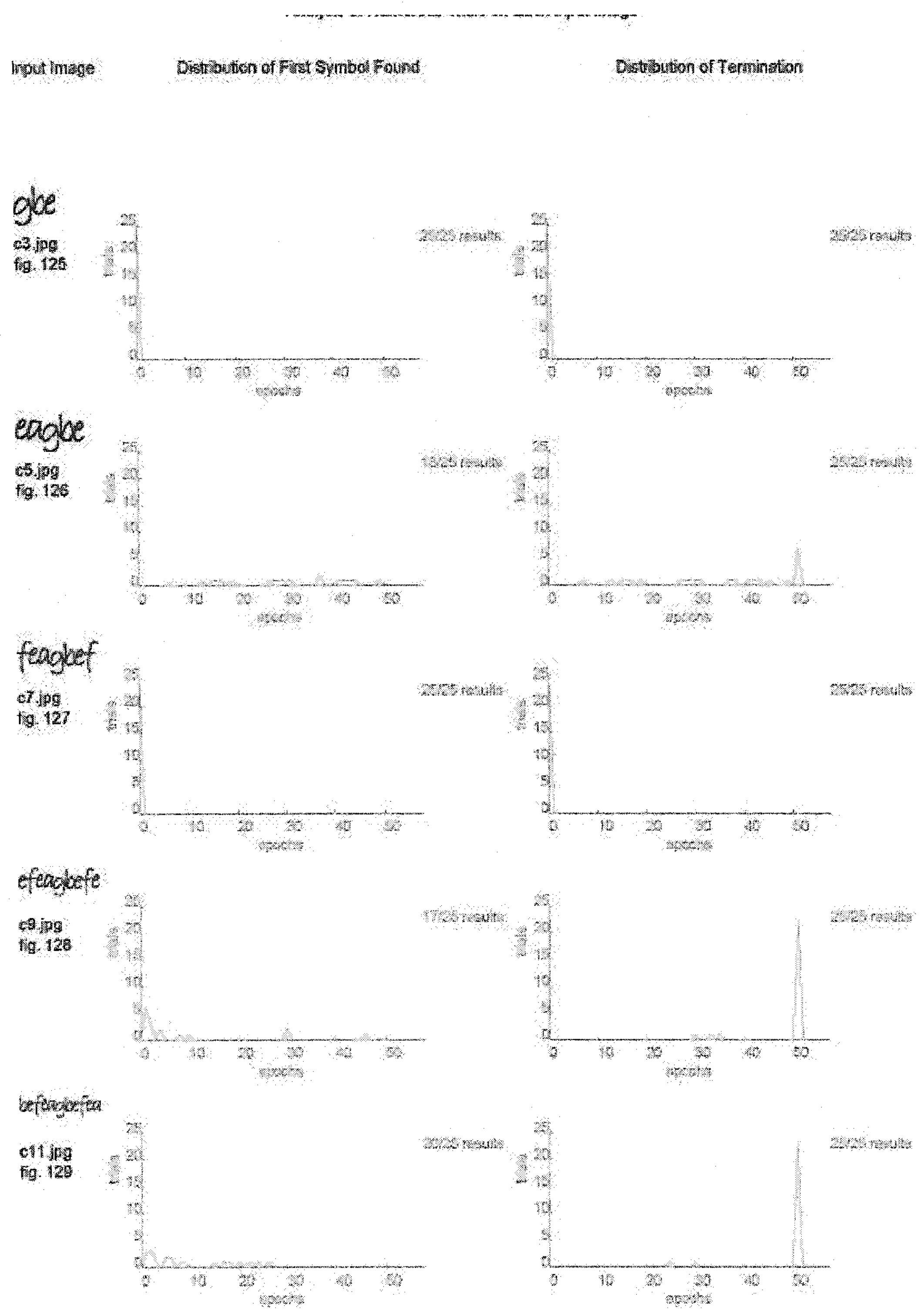


c11.jpg
fig. 124





Configuration: Seeded initialization, random parent search, 50 epoch limit, 15 individuals.

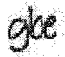


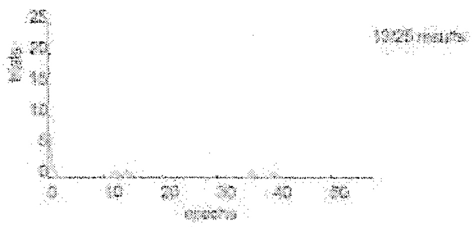
Configuration: Random initialization, full parent search, 50 epoch limit, 15 individuals.

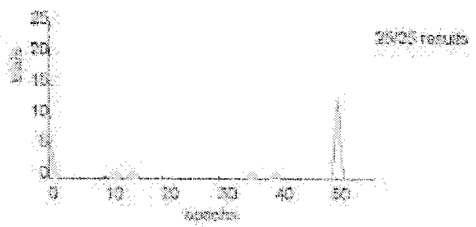
Input Image

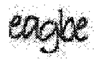
Distribution of First Symbol Found

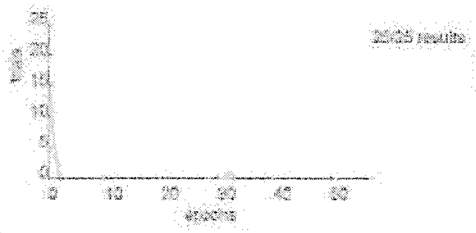
Distribution of Termination

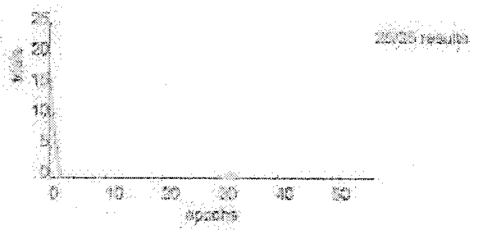

c3.jpg
fig. 130

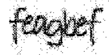


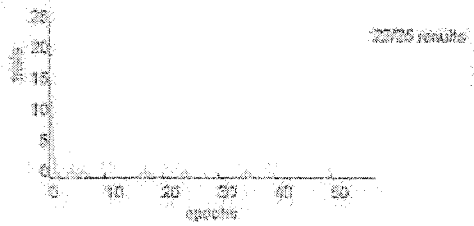


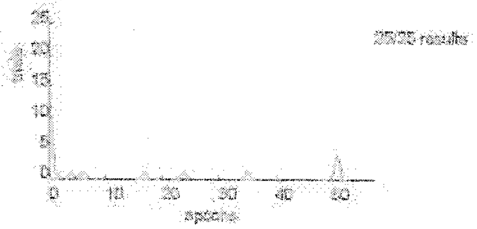

c5.jpg
fig. 131

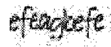


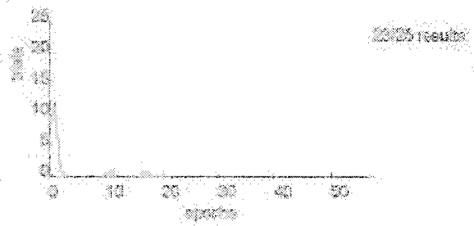


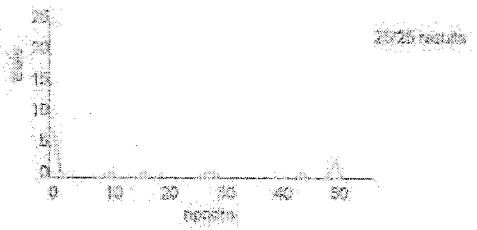

c7.jpg
fig. 132

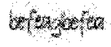


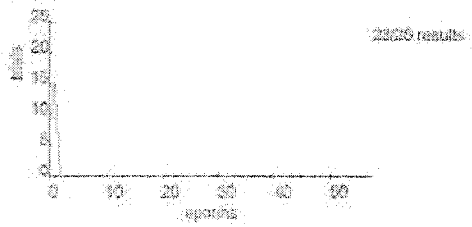


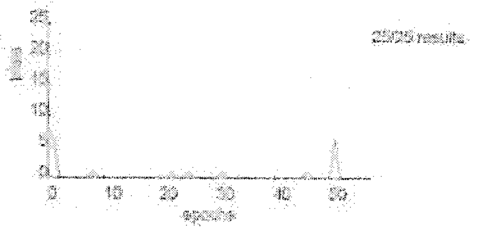

c9.jpg
fig. 133



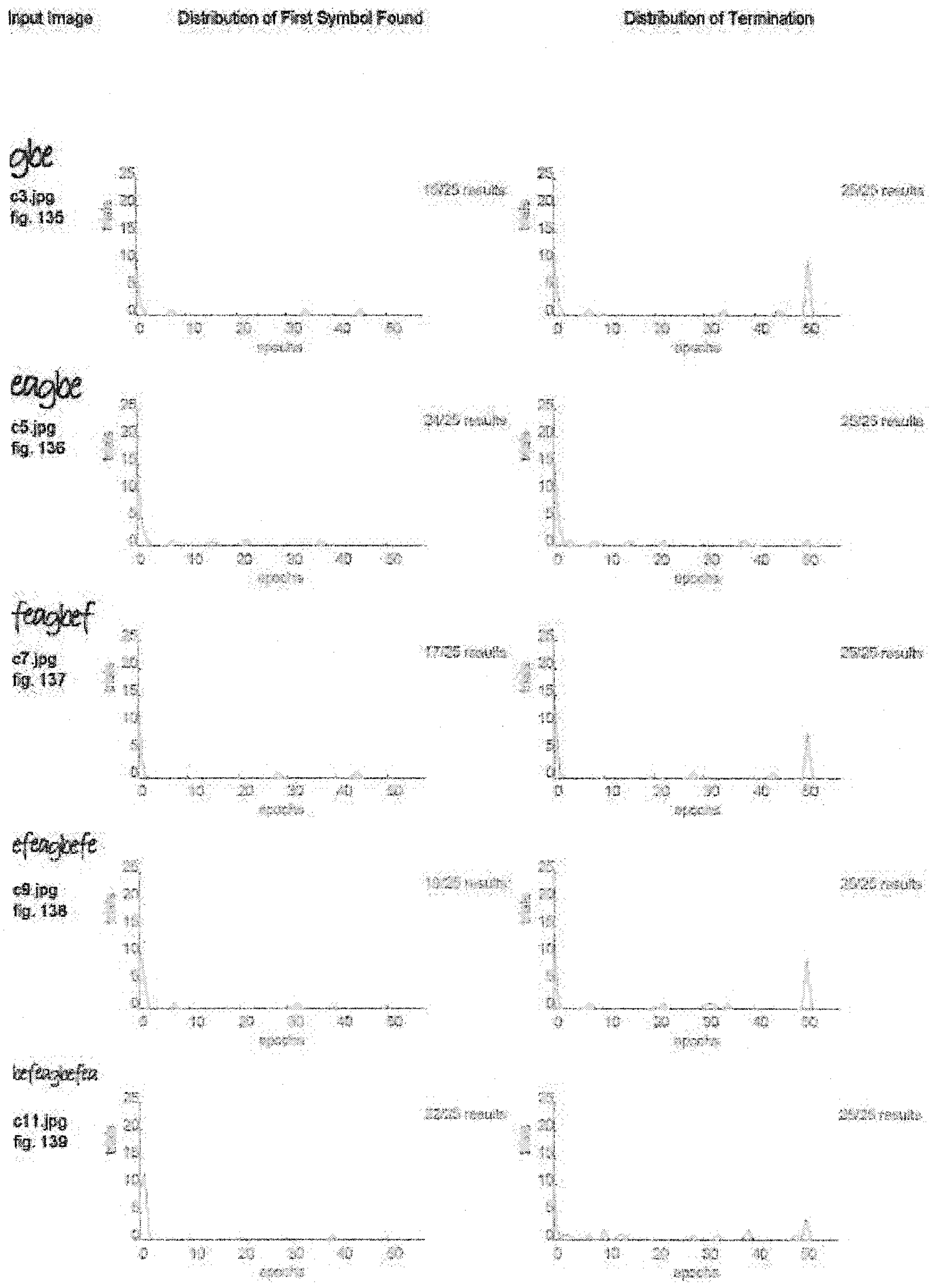



c11.jpg
fig. 134

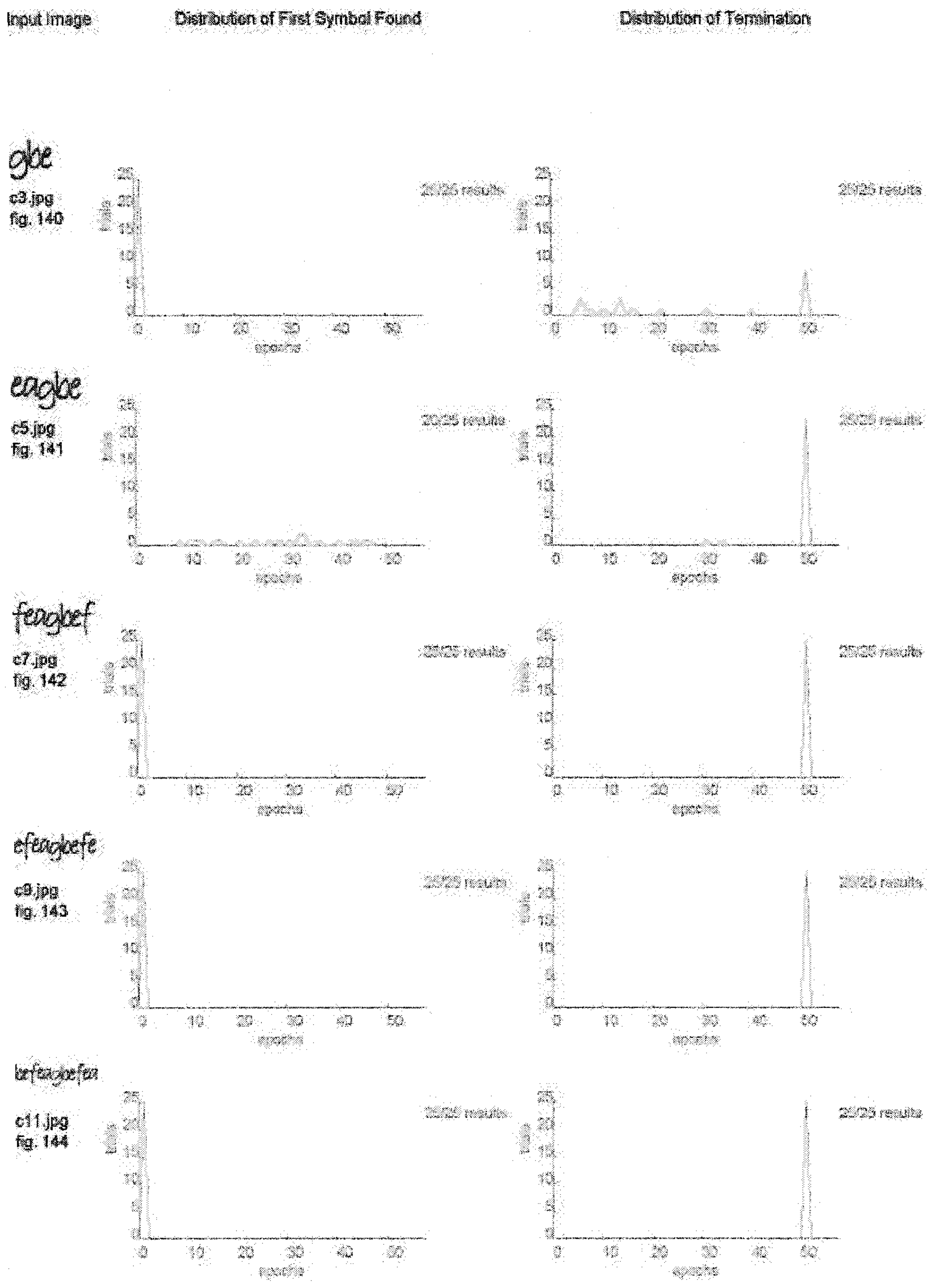




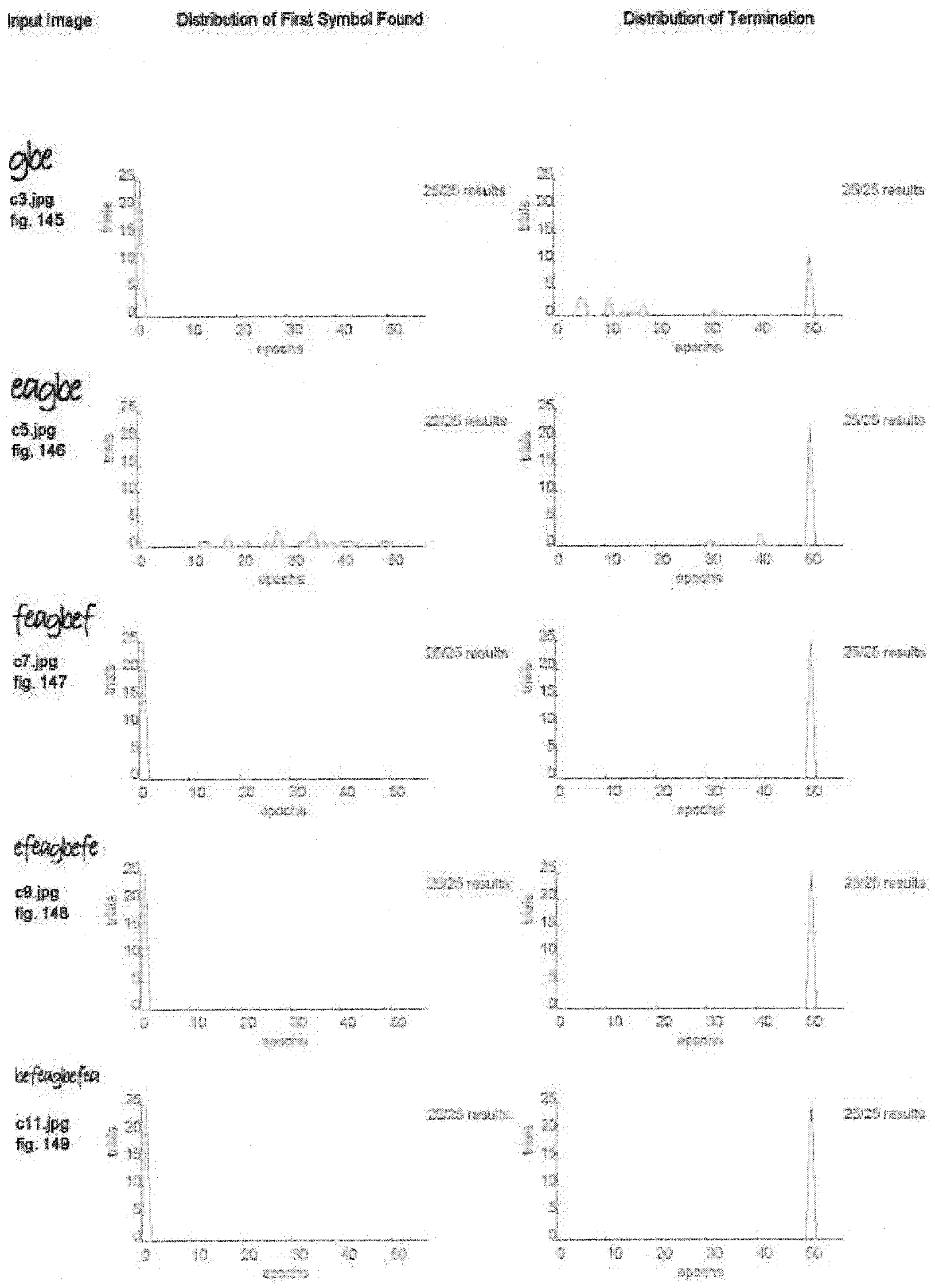
Configuration: Random initialization, random parent search, 50 epoch limit, 15 individuals.



Configuration: Seeded initialization, full parent search, 50 epoch limit, 20 individuals.



Configuration: Seeded initialization, random parent search, 50 epoch limit, 20 individuals.



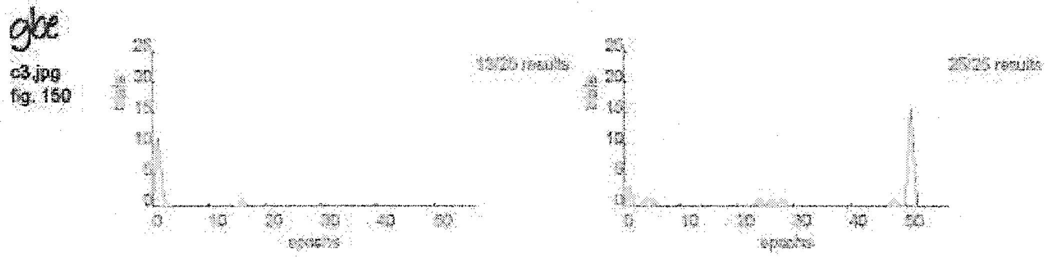
Configuration: Random initialization, full parent search, 50 epoch limit, 20 individuals.

[illegible]

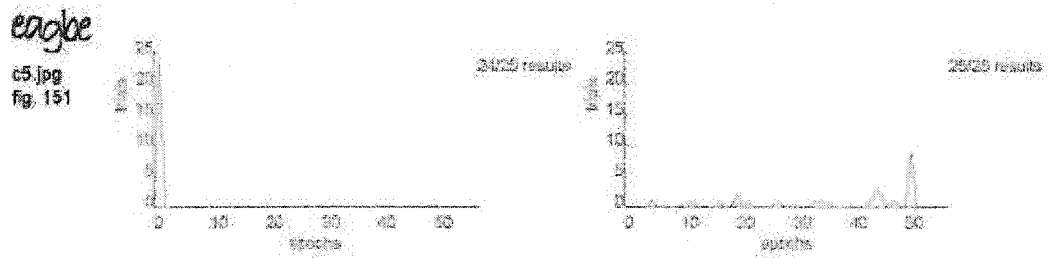
Input Image

Distribution of First Symbol Found

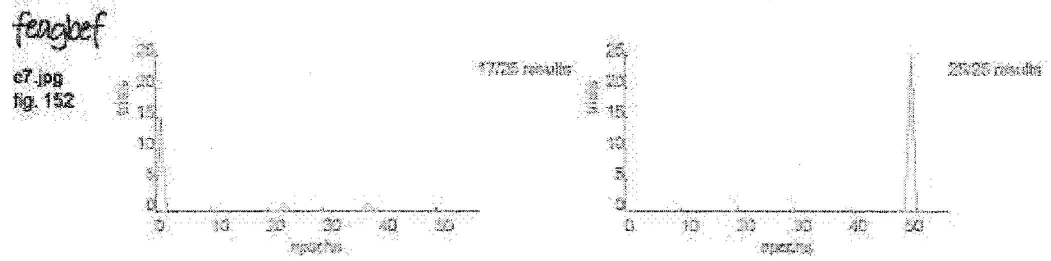
Distribution of Termination



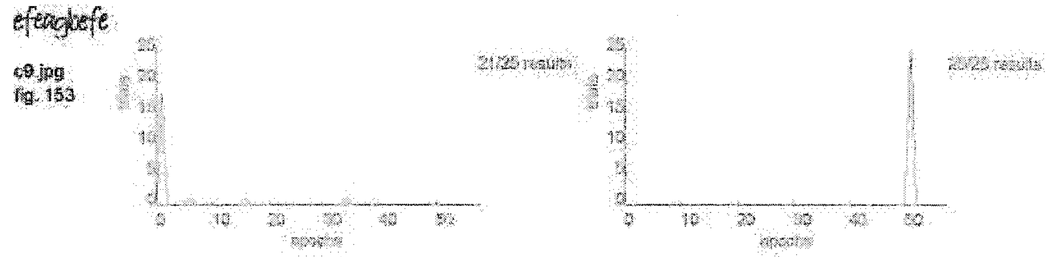
c3.jpg
fig. 150



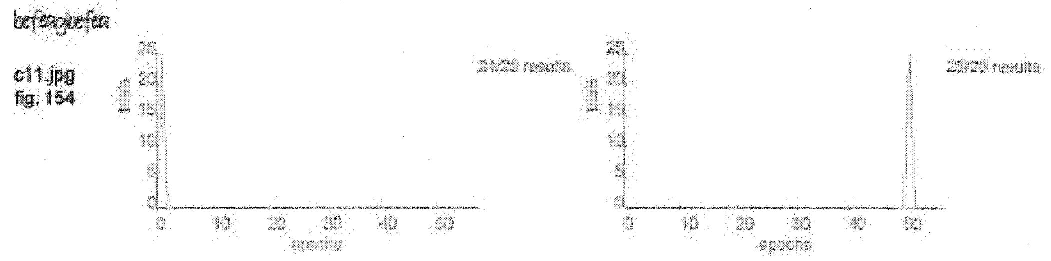
c5.jpg
fig. 151



c7.jpg
fig. 152



c9.jpg
fig. 153



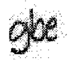
c11.jpg
fig. 154

Configuration: Random initialization, random parent search, 50 epoch limit, 12 individuals.

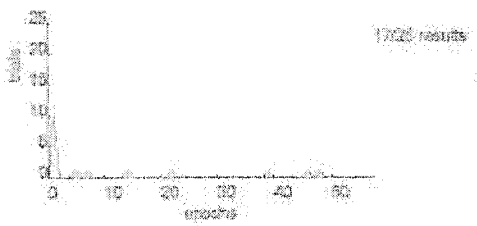
Input Image

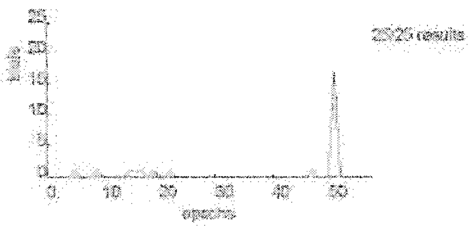
Distribution of First Symbol Found

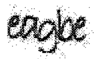
Distribution of Termination



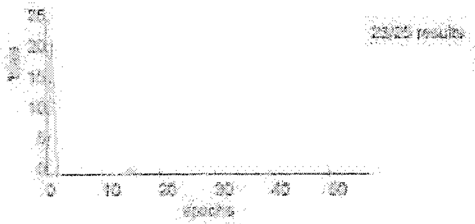
c3.jpg
fig. 155

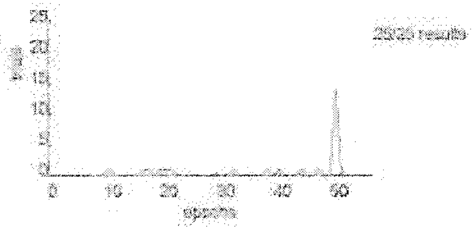


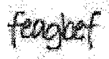




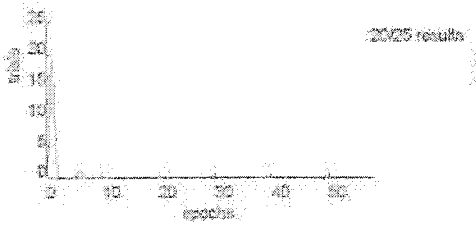
c5.jpg
fig. 156

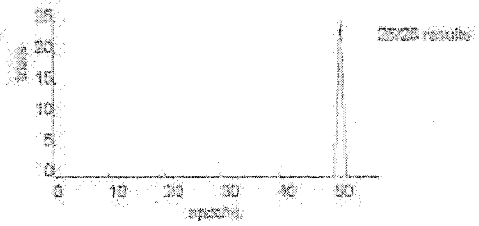


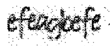




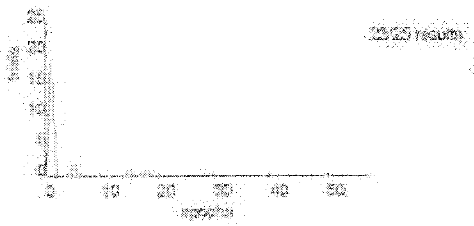
c7.jpg
fig. 157

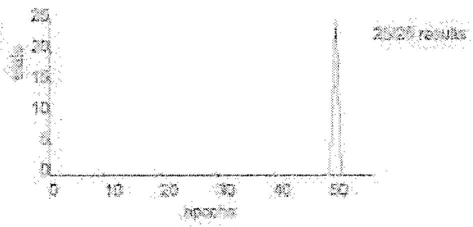


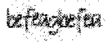




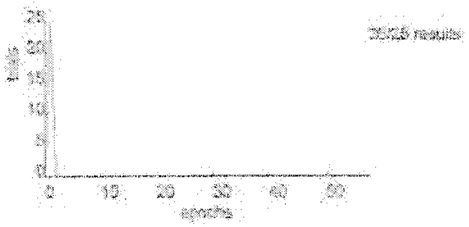
c9.jpg
fig. 158

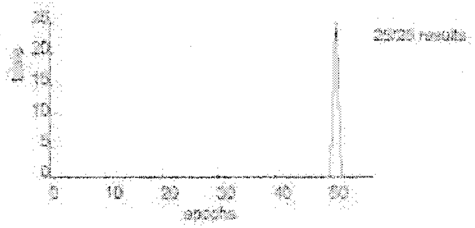






c11.jpg
fig. 159



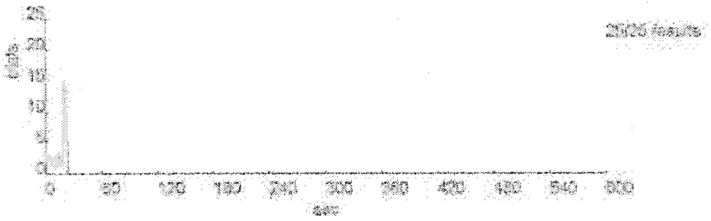


Configuration: Seeded initialization, full parent search, 50 epoch limit, 5 individuals.

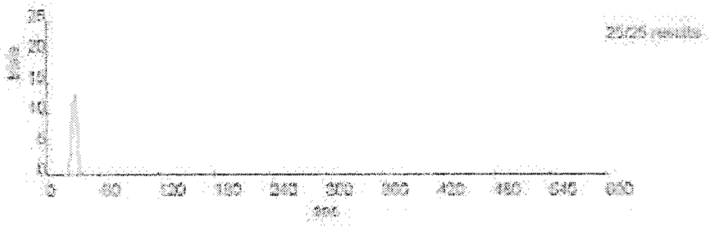
Input Image

Real Time Distribution of Active Evolution

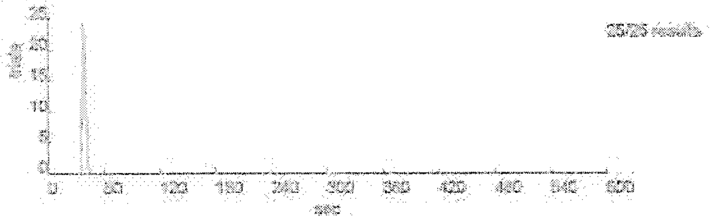
c3.jpg
fig. 160

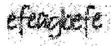


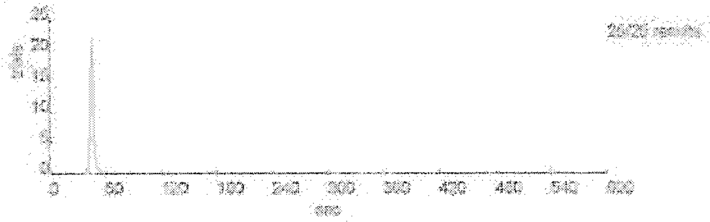
c5.jpg
fig. 161



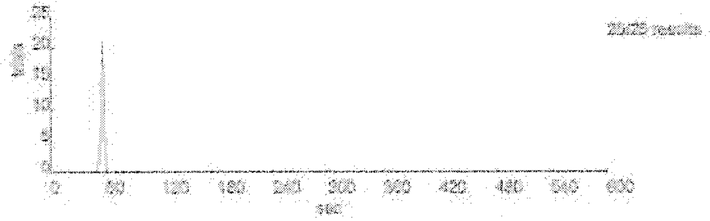
c7.jpg
fig. 162




c9.jpg
fig. 163



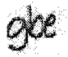
c11.jpg
fig. 164



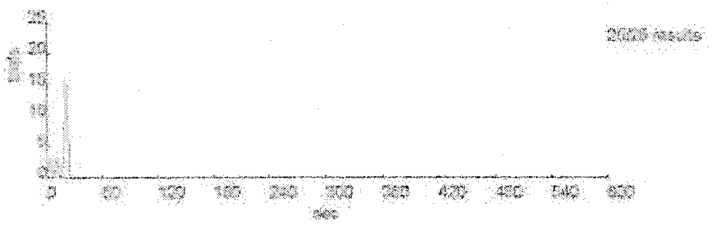
Configuration: Seeded initialization, random parent search, 50 epoch limit, 5 individuals.

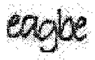
Input Image

Real Time Distribution of Active Evolution

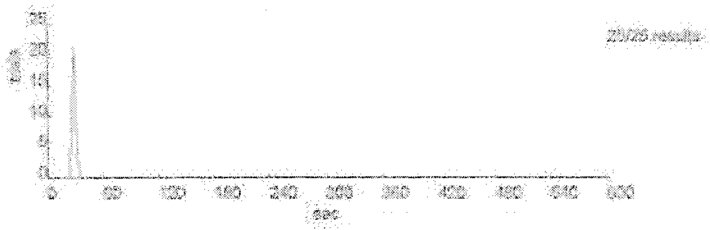


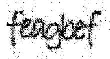
c3.jpg
fig. 165



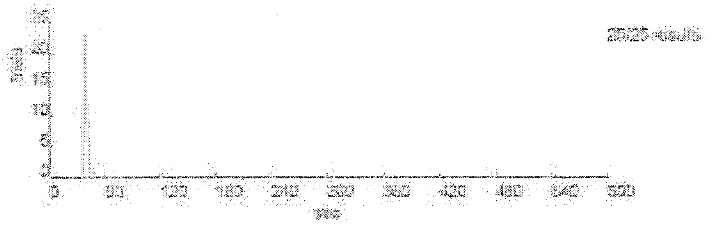


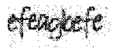
c5.jpg
fig. 166



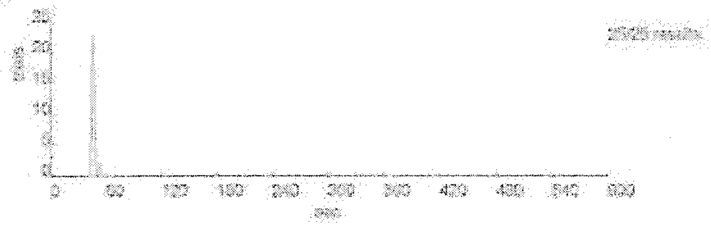


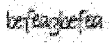
c7.jpg
fig. 167



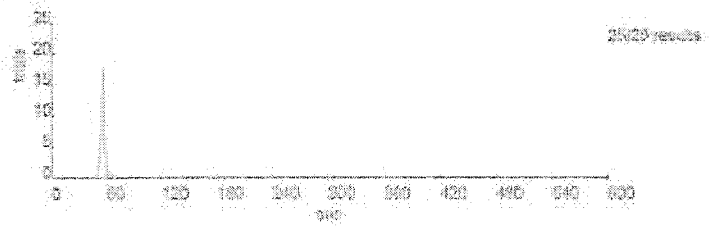


c9.jpg
fig. 168





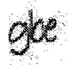
c11.jpg
fig. 169



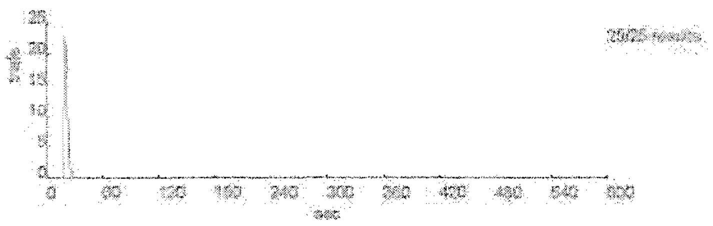
Configuration: Random initialization, full parent search, 50 epoch limit, 5 individuals.

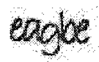
Input Image

Real Time Distribution of Active Evolution

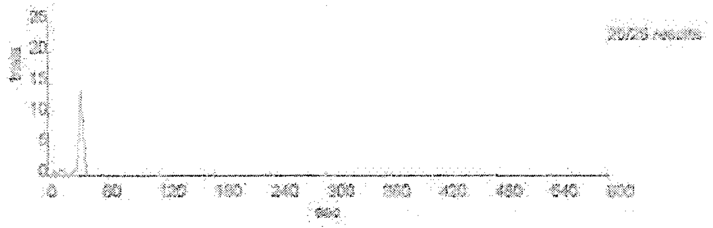


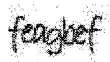
c3.jpg
fig. 170



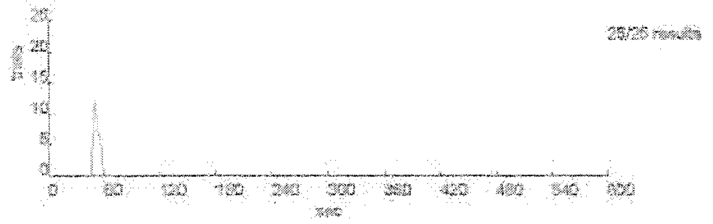


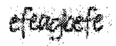
c5.jpg
fig. 171



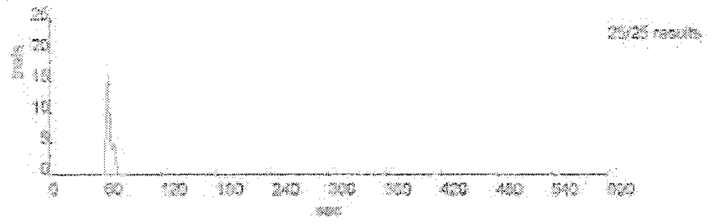


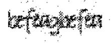
c7.jpg
fig. 172



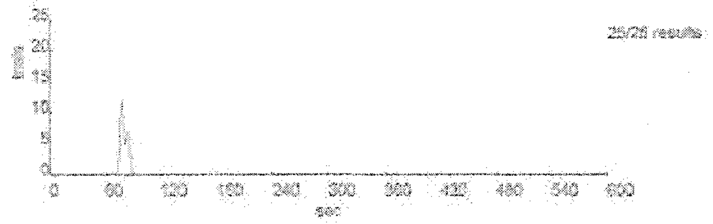


c9.jpg
fig. 173





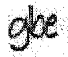
c11.jpg
fig. 174



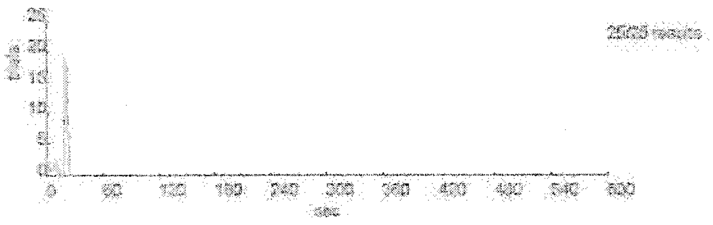
Configuration: Random initialization, random parent search, 50 epoch limit, 5 individuals.

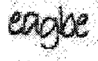
Input Image

Real Time Distribution of Active Evolution

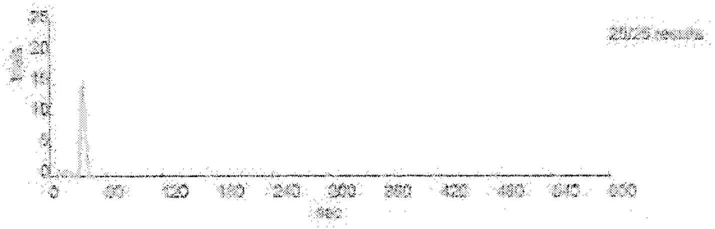


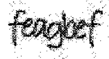
c3.jpg
fig. 175



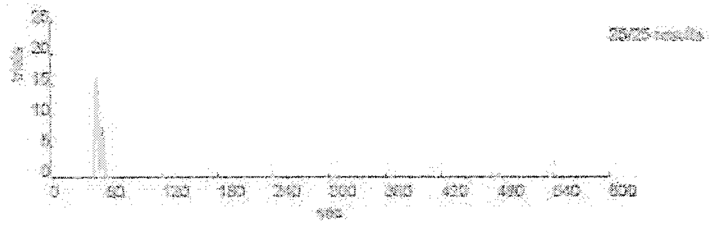


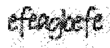
c5.jpg
fig. 176



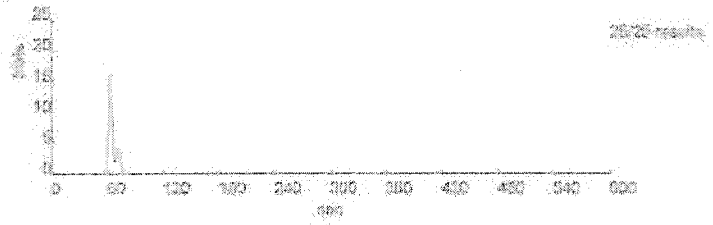


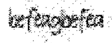
c7.jpg
fig. 177



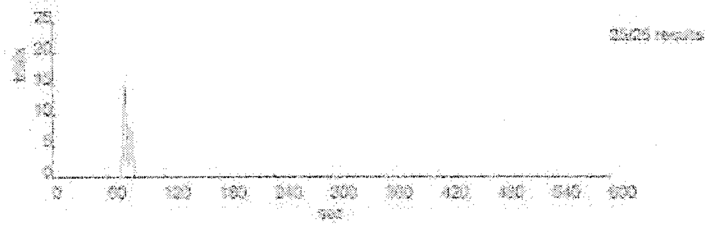


c9.jpg
fig. 178



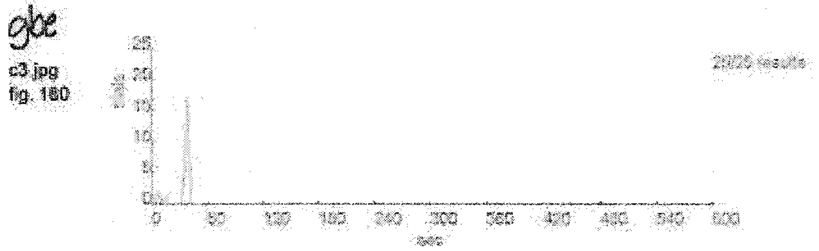


c11.jpg
fig. 179

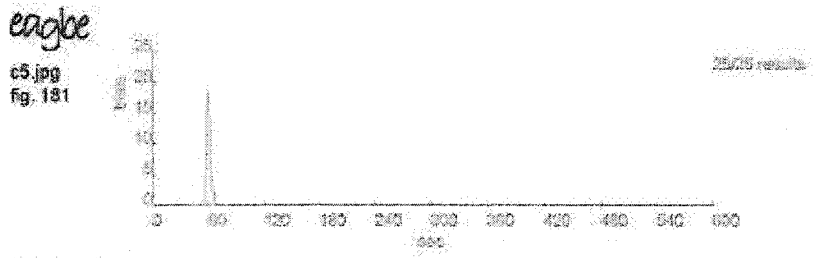


Configuration: Seeded initialization, full parent search, 50 epoch limit, 10 individuals.

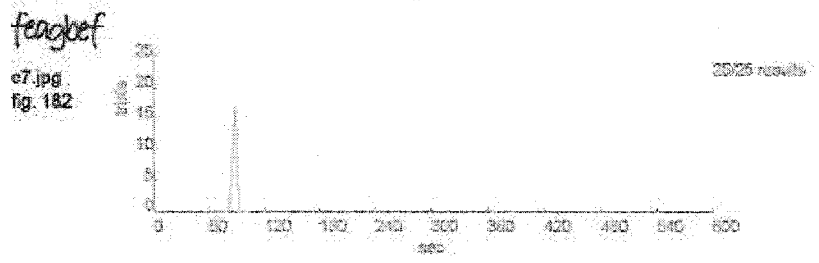
Input Image Real Time Distribution of Active Evolution



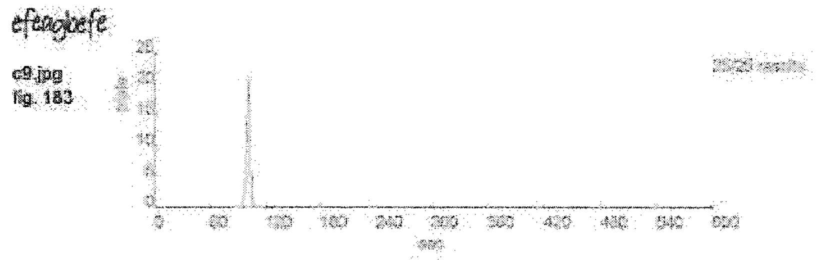
c3.jpg
fig. 180



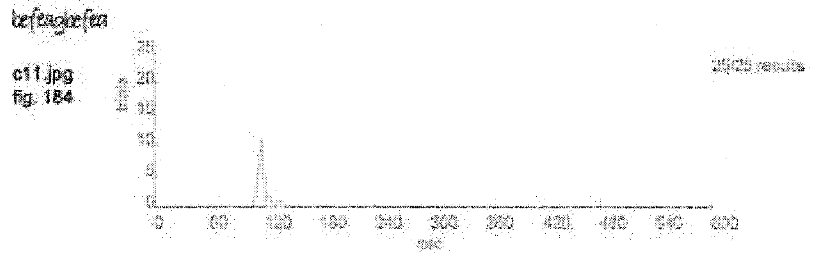
c5.jpg
fig. 181



c7.jpg
fig. 182



c9.jpg
fig. 183

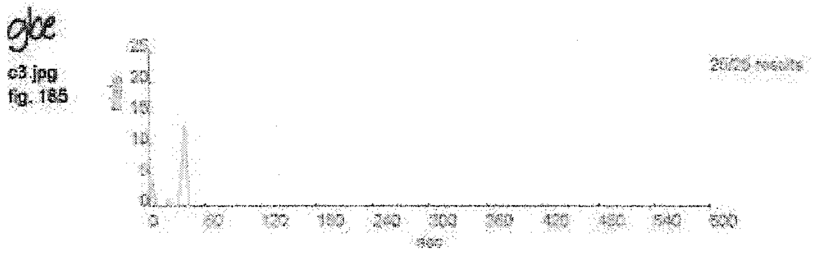


c11.jpg
fig. 184

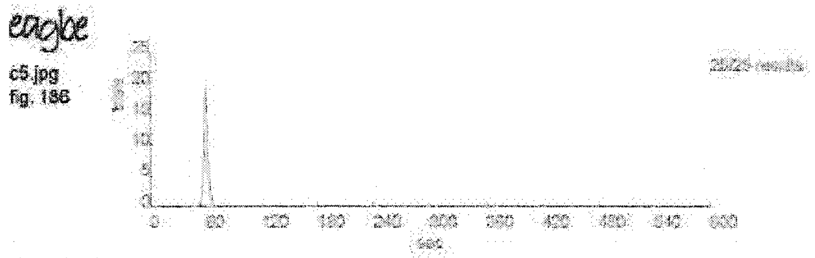
Configuration: Seeded initialization, random parent search, 50 epoch limit, 10 individuals.

Input Image

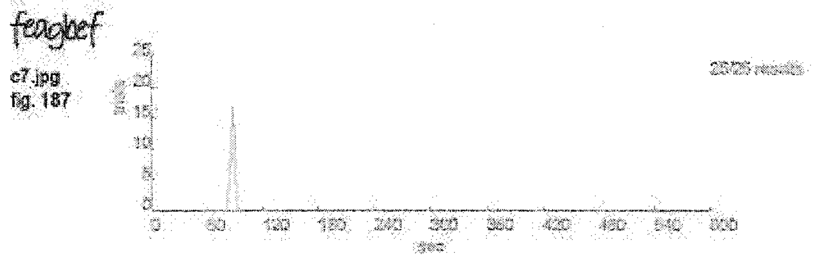
Real Time Distribution of Active Evolution



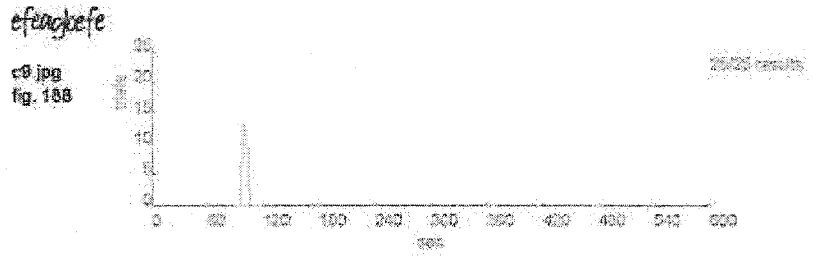
c3.jpg
fig. 185



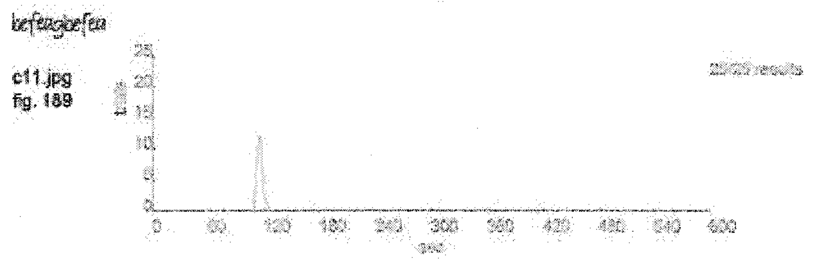
c5.jpg
fig. 186



c7.jpg
fig. 187



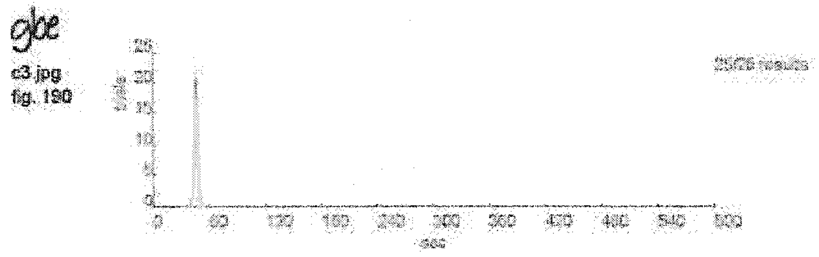
c9.jpg
fig. 188



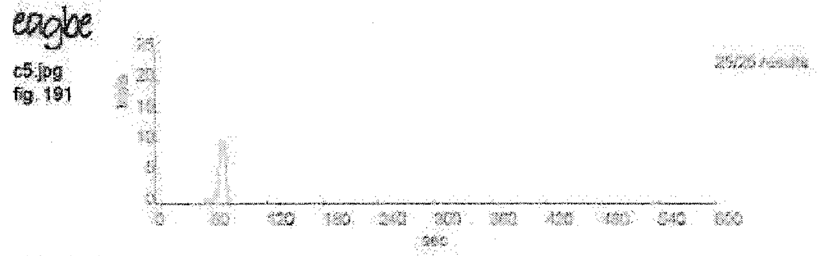
c11.jpg
fig. 189

Configuration: Random initialization, full parent search, 50 epoch limit, 10 individuals.

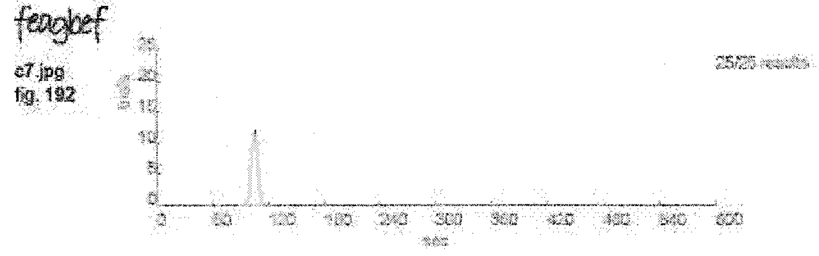
Input Image Real Time Distribution of Active Evolution



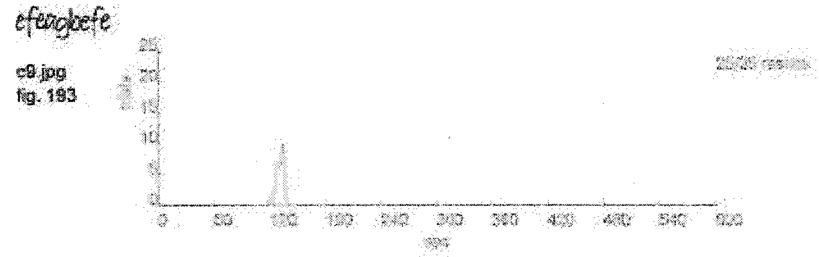
c3.jpg
fig. 190



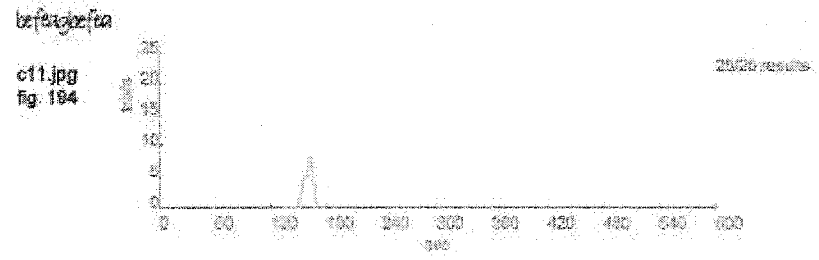
c5.jpg
fig. 191



c7.jpg
fig. 192

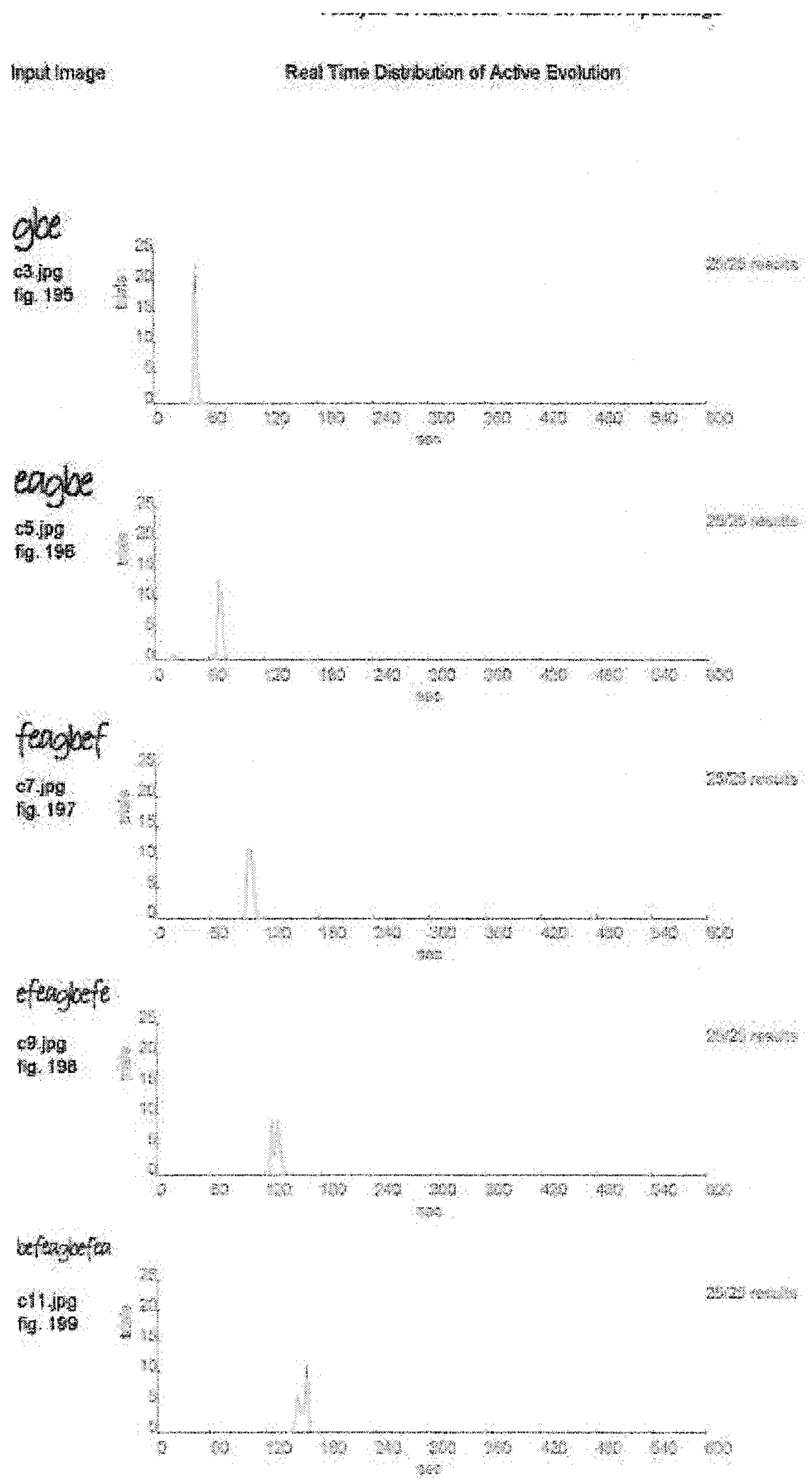


c9.jpg
fig. 193



c11.jpg
fig. 194

Configuration: Random initialization, random parent search, 50 epoch limit, 10 individuals.

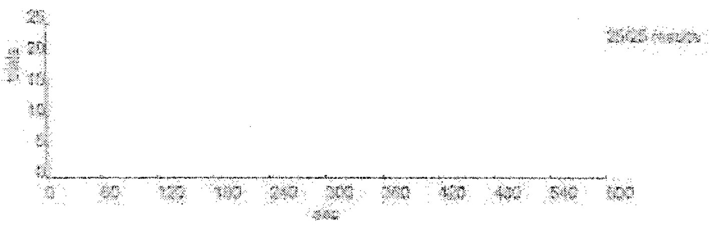


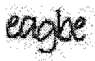
Configuration: Seeded initialization, full parent search, 50 epoch limit, 15 individuals.

Input image

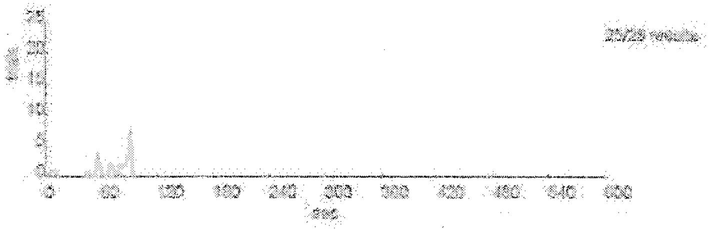
Real Time Distribution of Active Evolution

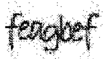
c3.jpg
fig. 200



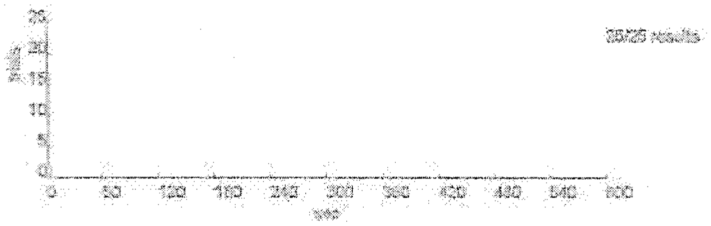


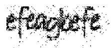
c5.jpg
fig. 201



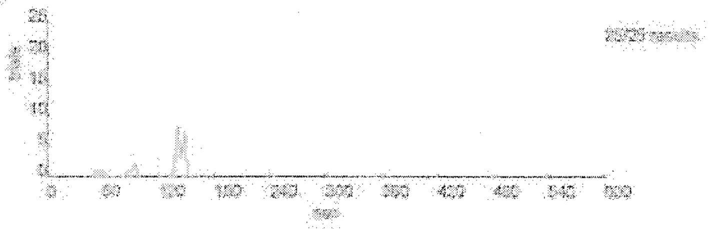


c7.jpg
fig. 202

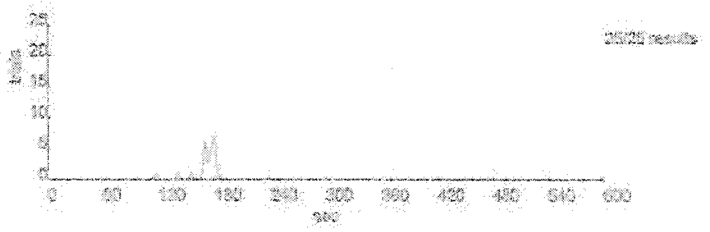




c9.jpg
fig. 203

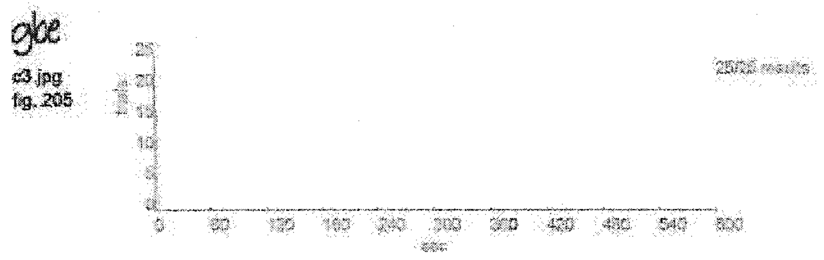


c11.jpg
fig. 204

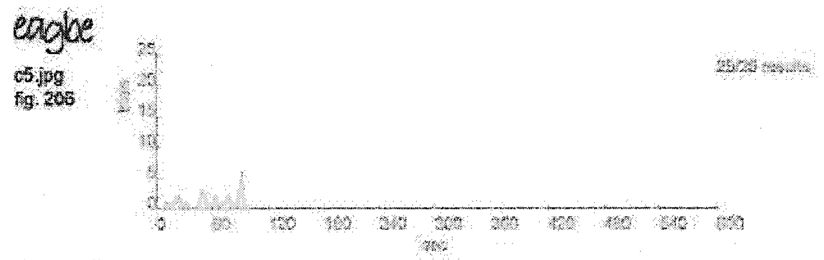


Configuration: Seeded initialization, random parent search, 50 epoch limit, 15 individuals.

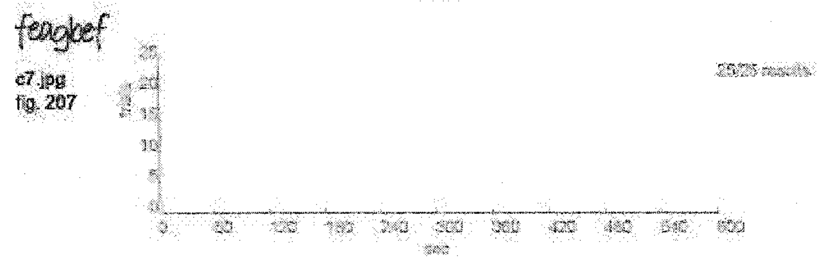
Input Image Real Time Distribution of Active Evolution



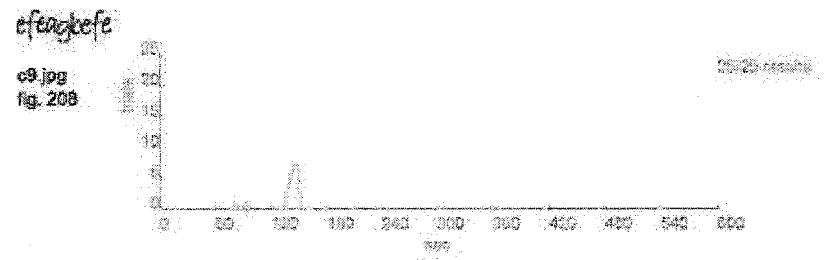
c3.jpg
fig. 205



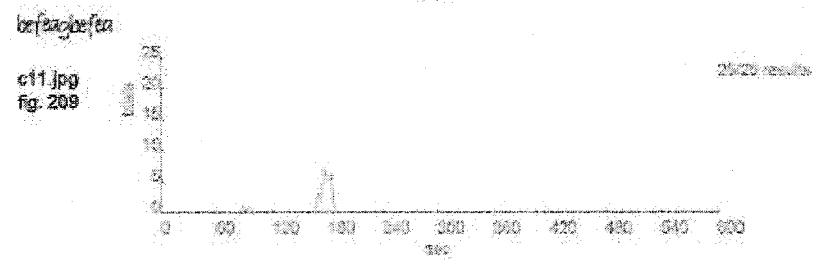
c5.jpg
fig. 206



c7.jpg
fig. 207



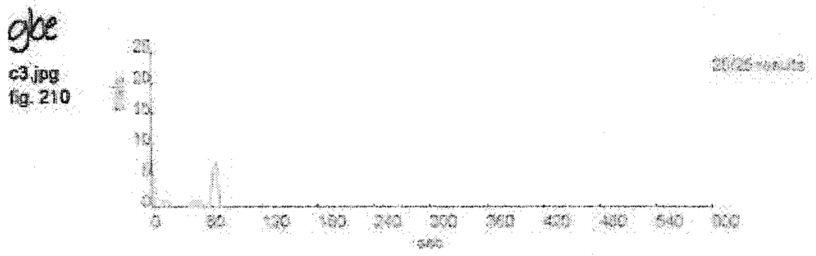
c9.jpg
fig. 208



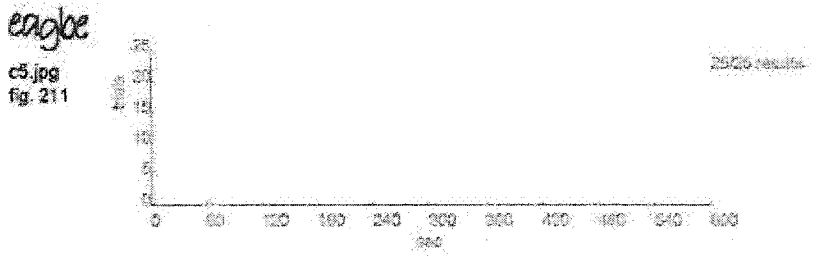
c11.jpg
fig. 209

Configuration: Random initialization, full parent search, 50 epoch limit, 15 individuals.

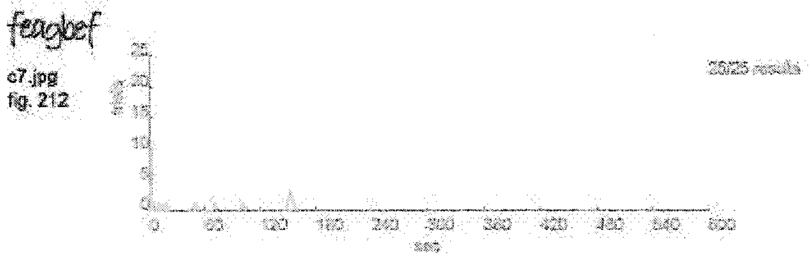
Input Image Real Time Distribution of Active Evolution



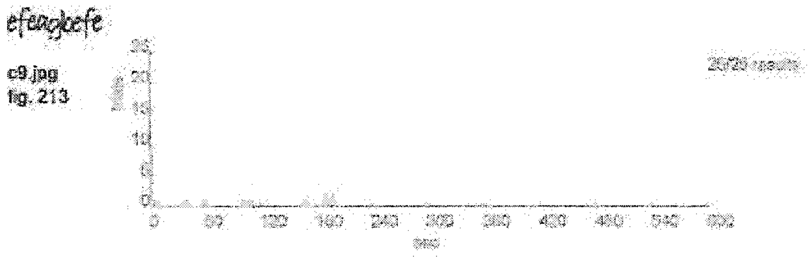
c3.jpg
fig. 210



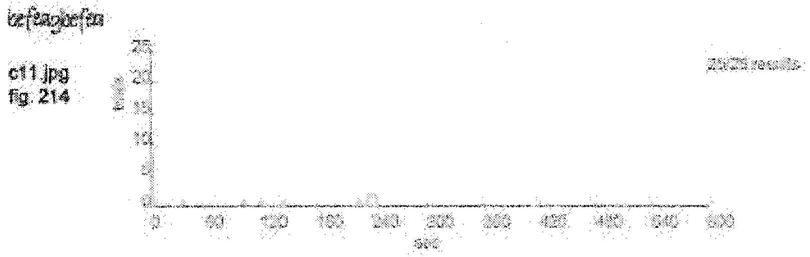
c5.jpg
fig. 211



c7.jpg
fig. 212



c9.jpg
fig. 213

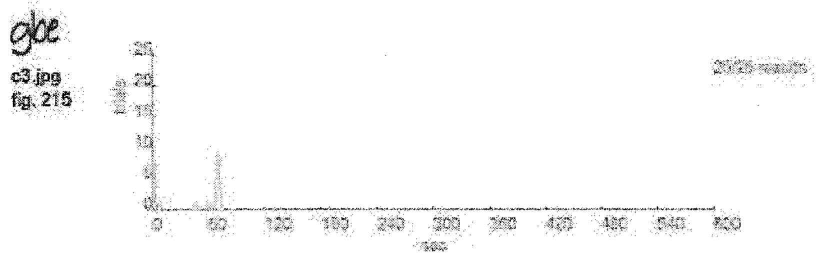


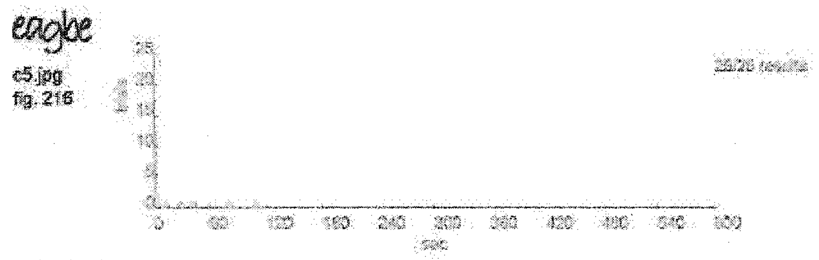
c11.jpg
fig. 214

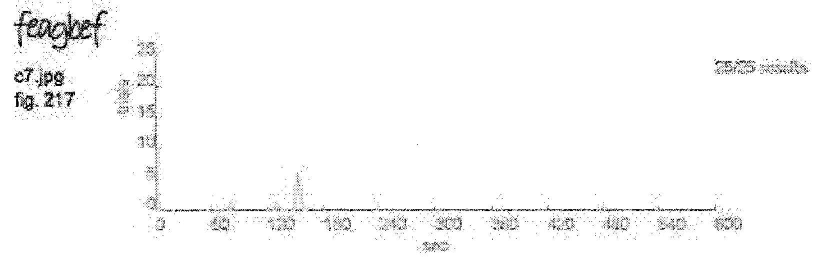
Configuration: Random initialization, random parent search, 50 epoch limit, 15 individuals.

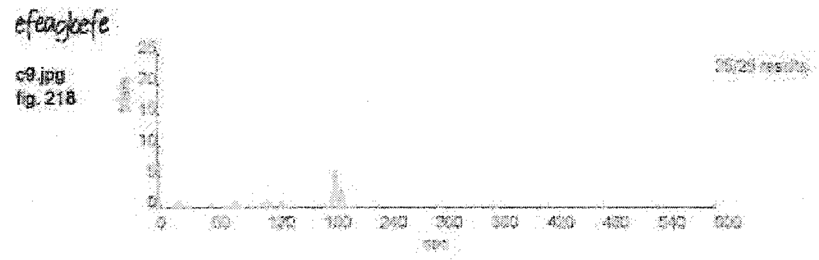
Input Image

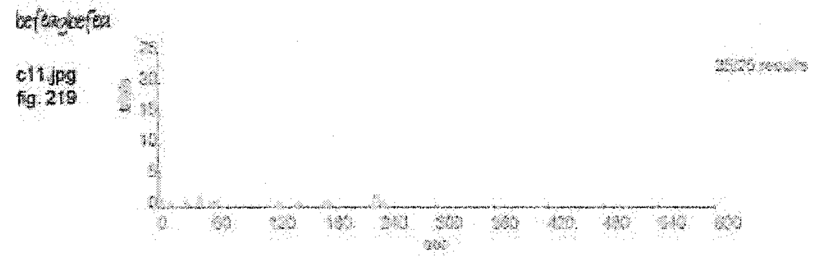
Real Time Distribution of Active Evolution



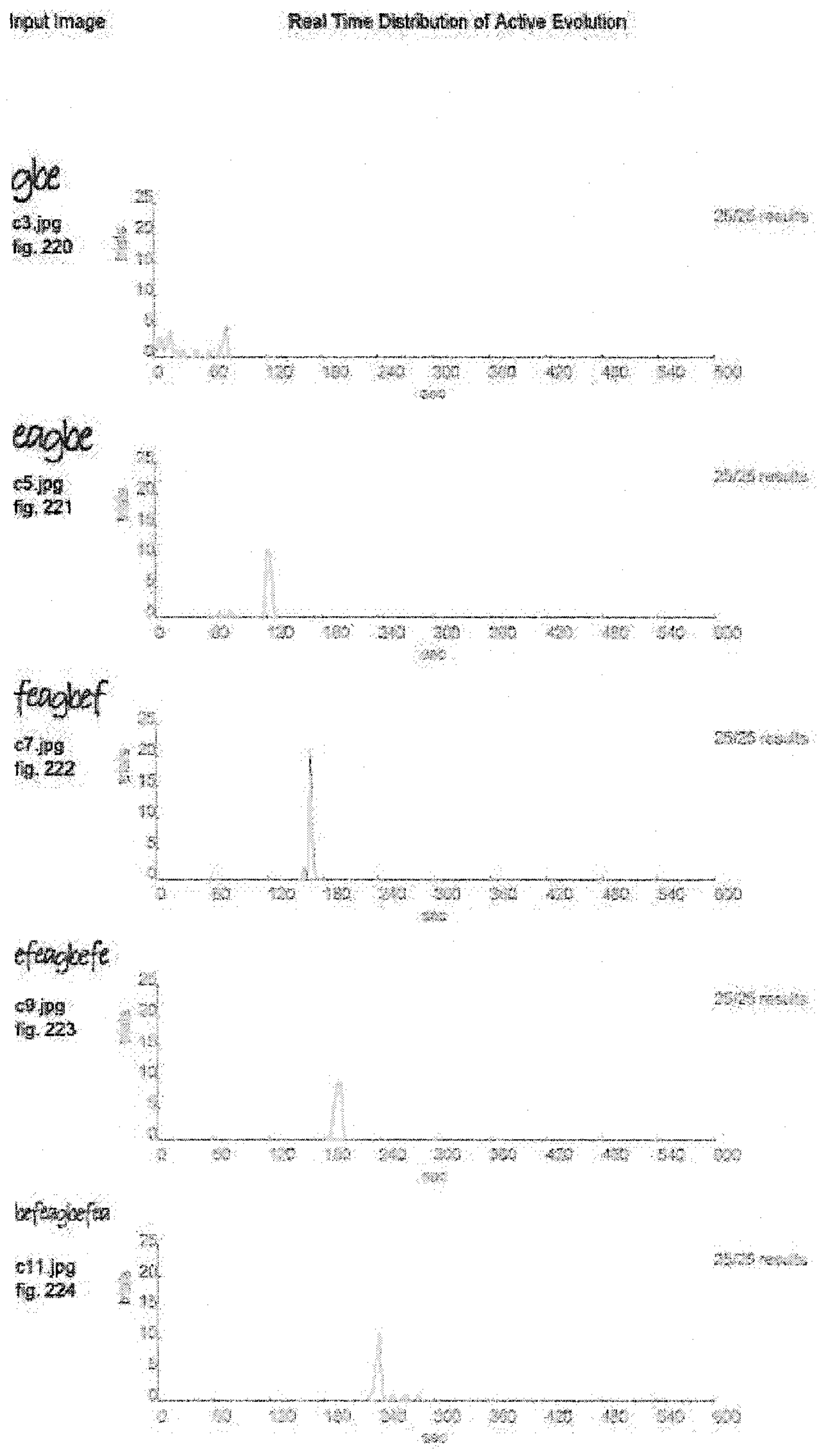








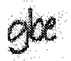
Configuration: Seeded initialization, full parent search, 50 epoch limit, 20 individuals.



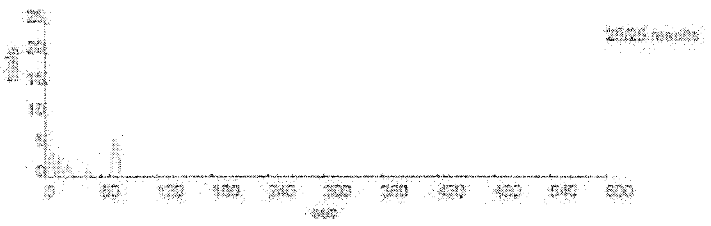
Configuration: Seeded initialization, random parent search, 50 epoch limit, 20 individuals.

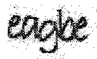
Input Image

Real Time Distribution of Active Evolution

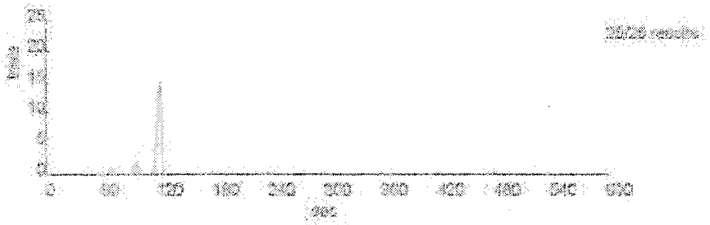

gbe

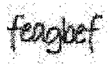
c3.jpg
fig. 225



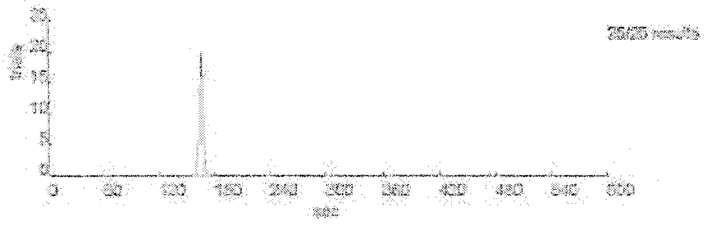

eagbe

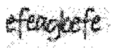
c5.jpg
fig. 226



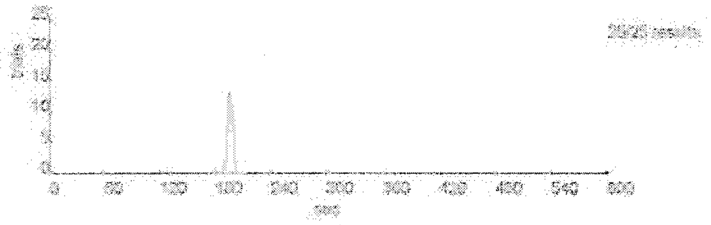

feagbef

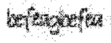
c7.jpg
fig. 227



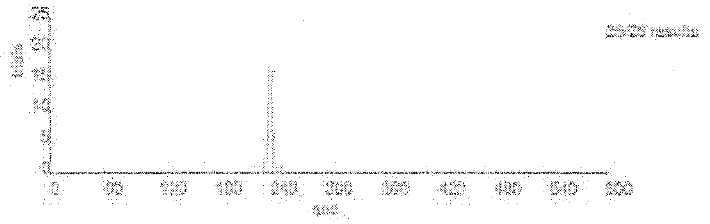

efeagbefe

c9.jpg
fig. 228




befeagbefea

c11.jpg
fig. 229

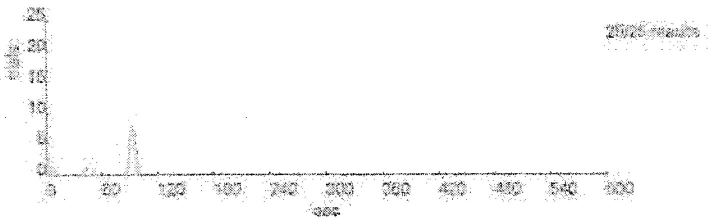


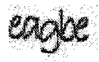
Configuration: Random initialization, full parent search, 50 epoch limit, 20 individuals.

Input Image

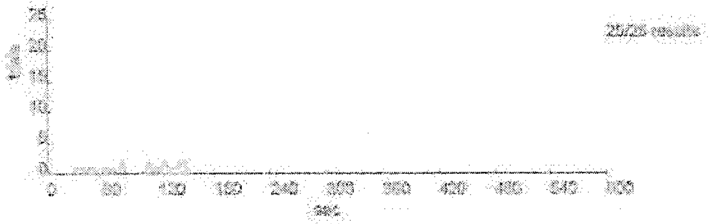
Real Time Distribution of Active Evolution

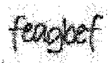
c3.jpg
fig. 230



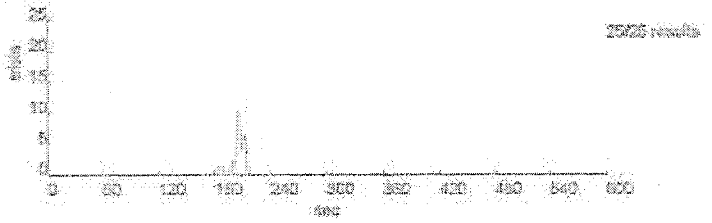


c5.jpg
fig. 231

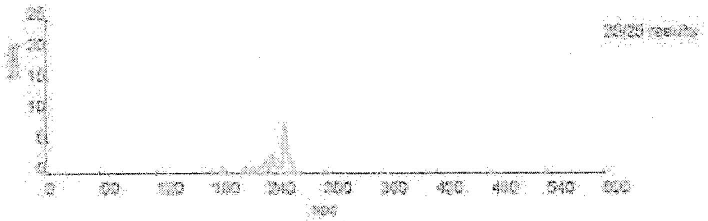


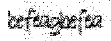


c7.jpg
fig. 232

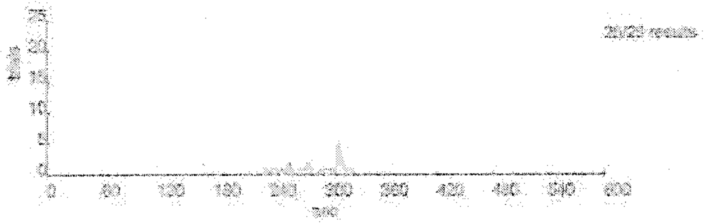


c9.jpg
fig. 233



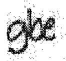


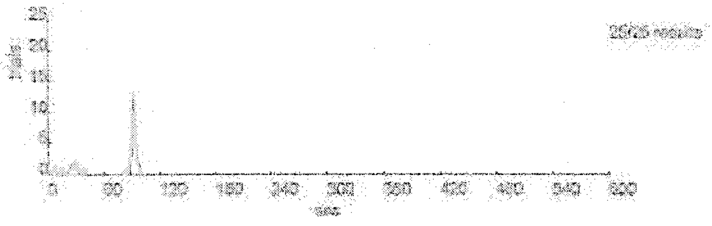
c11.jpg
fig. 234

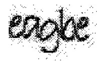


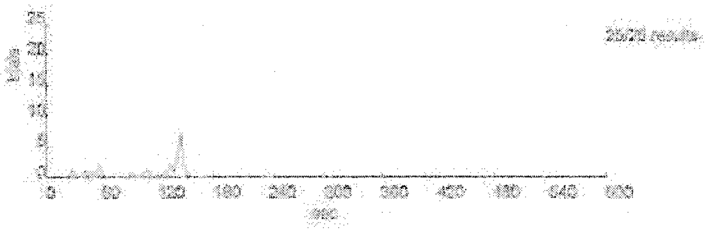
Configuration: Random initialization, random parent search, 50 epoch limit, 20 individuals.

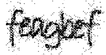
Input Image Real Time Distribution of Active Evolution

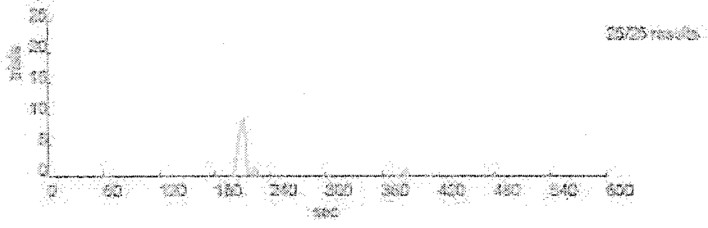

c3.jpg
fig. 235

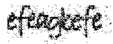


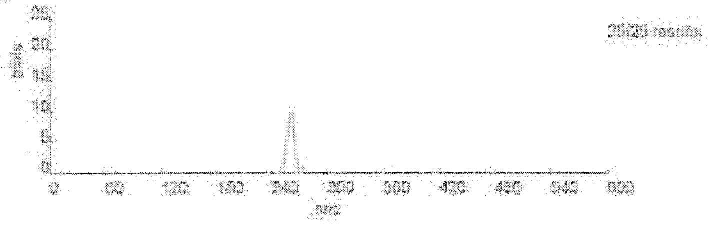

c5.jpg
fig. 236

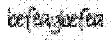


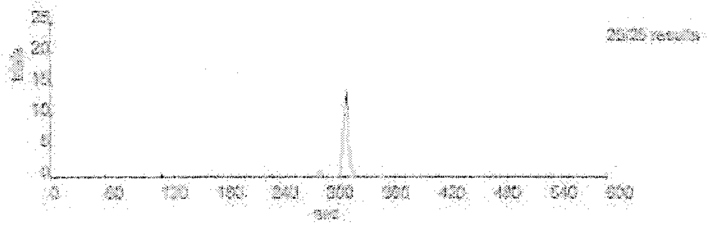

c7.jpg
fig. 237




c9.jpg
fig. 238




c11.jpg
fig. 239



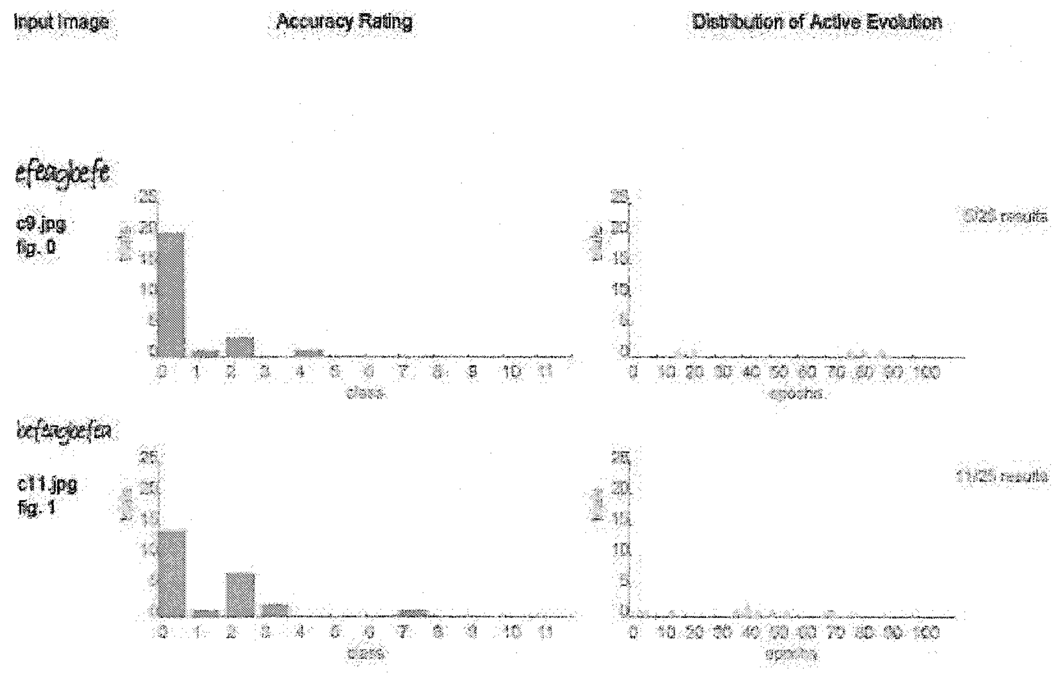
Appendix D:

This appendix contains the results for the experiments with various algoirthm configurations, on a strict set of input images. For all experiments, evolution was terminated after 100 epochs. The Appendix is separated into three sections:

- The first seciton presents the accuracy and duration of active evolution.
- The second section presents the epoch of first symbol found, and the epoch of termination data.
- The third section presents the duration of time in seconds for which the algoirhtm was run.

The separation is necessary to present the charts clearly on standard size paper.

Configuration: Seeded initialization, full parent search, 100 epoch limit, 5 individuals.



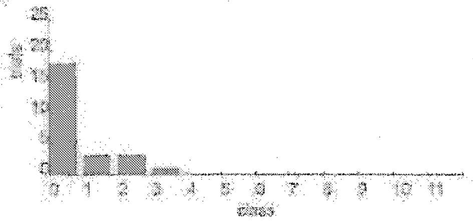
Configuration: Seeded initialization, random parent search, 100 epoch limit, 5 individuals.

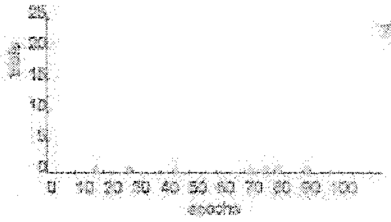
Input Image

Accuracy Rating

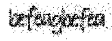
Distribution of Active Evolution

c9.jpg
fig. 2

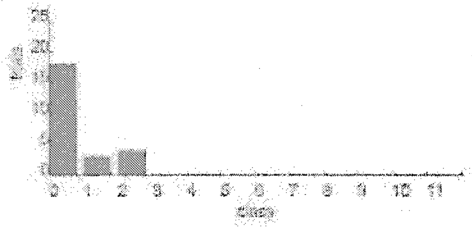


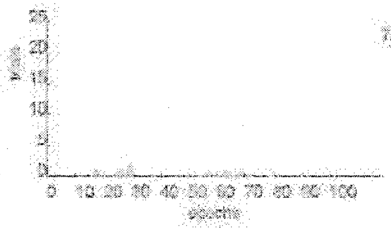


7/25 results



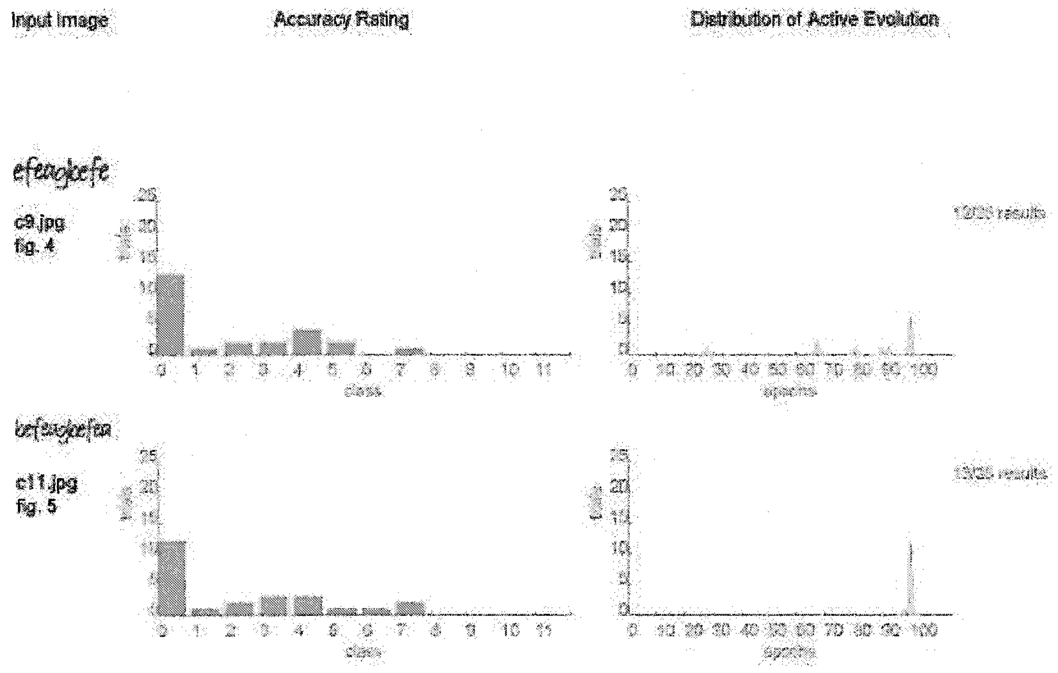
c11.jpg
fig. 3





7/25 results

Configuration: Random initialization, full parent search, 100 epoch limit, 5 individuals.

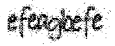


Configuration: Seeded initialization, full parent search, 100 epoch limit, 5 individuals.

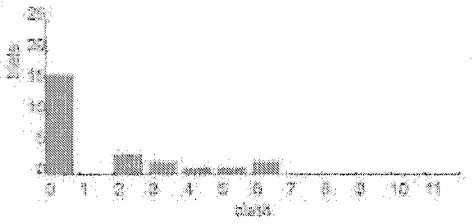
Input Image

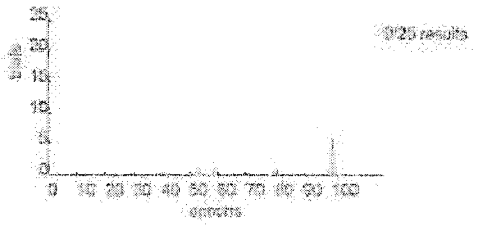
Accuracy Rating

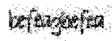
Distribution of Active Evolution



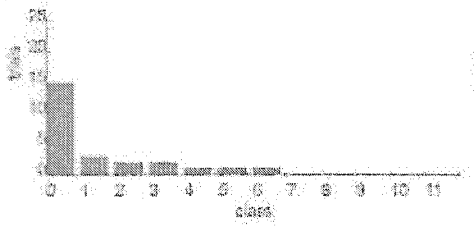
c9.jpg
fig. 6

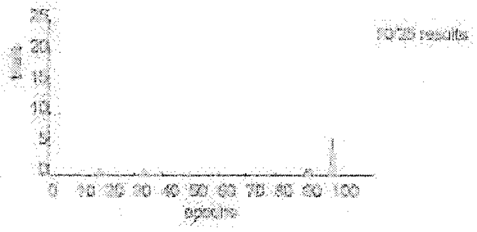




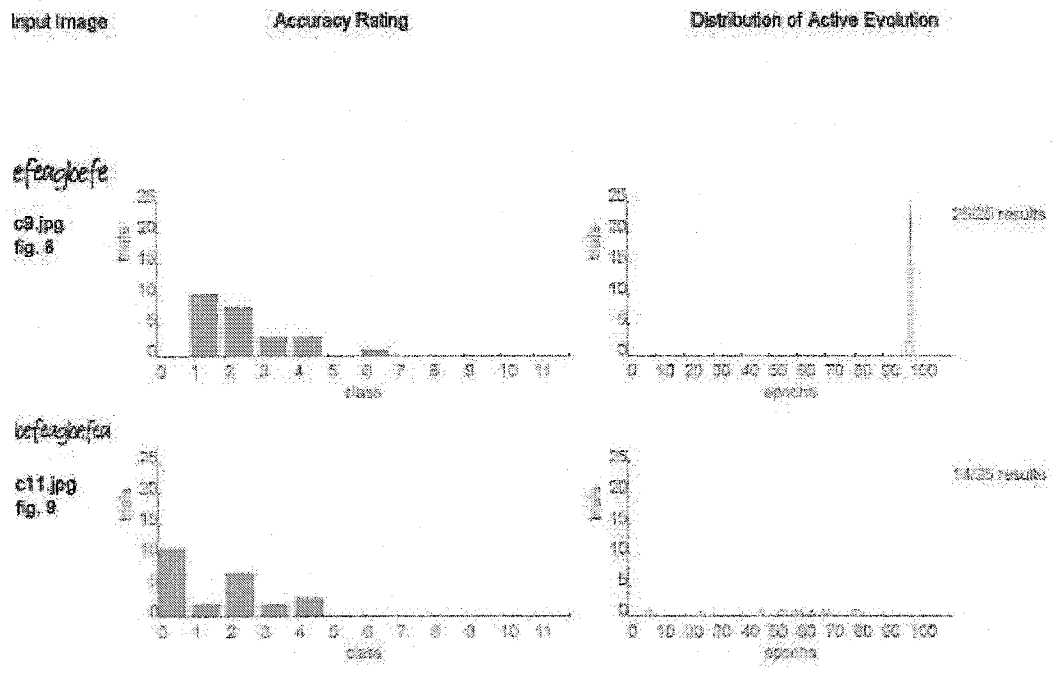


c11.jpg
fig. 7





Configuration: Seeded initialization, full parent search, 100 epoch limit, 10 individuals.

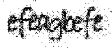


Configuration: Seeded initialization, random parent search, 100 epoch limit, 10 individuals.

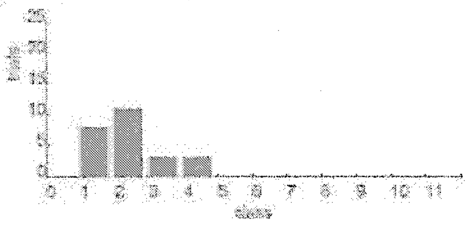
Input Image

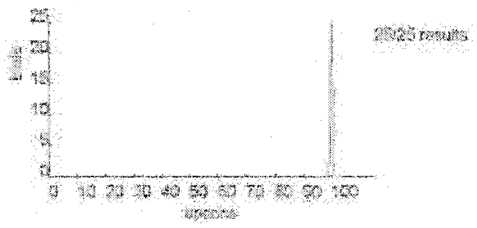
Accuracy Rating

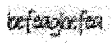
Distribution of Active Evolution



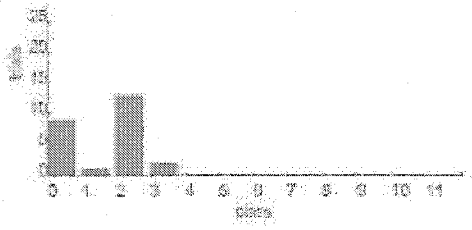
c9.jpg
fig. 10

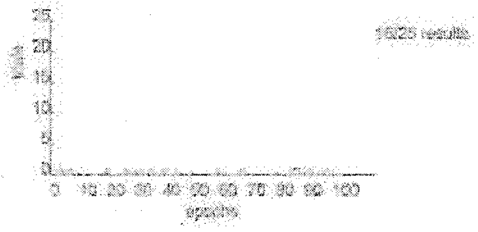




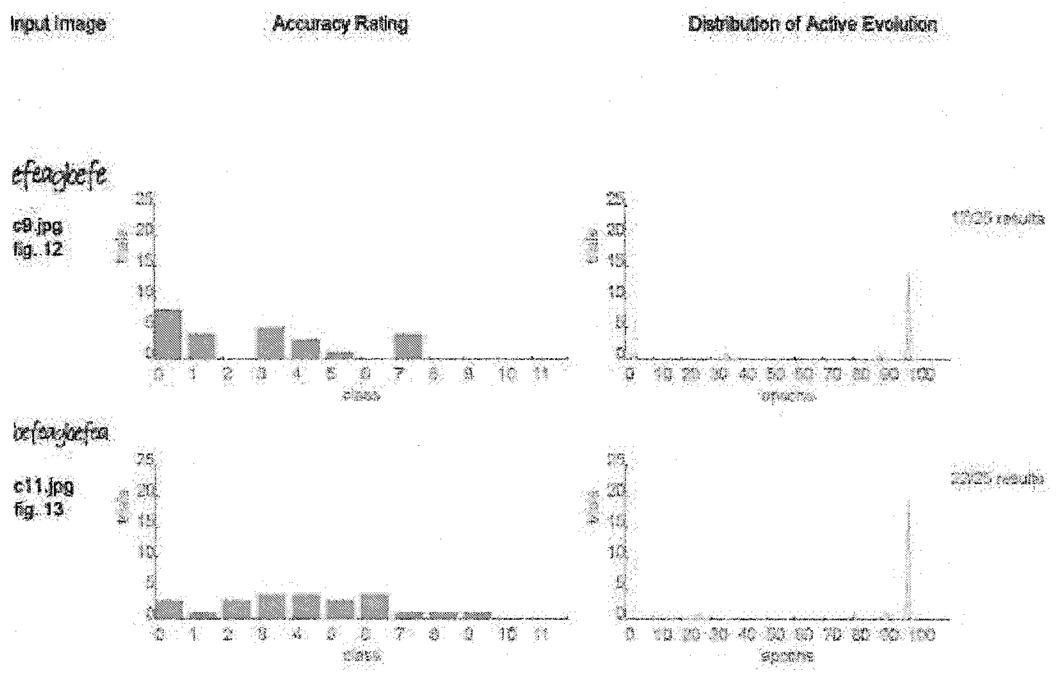


c11.jpg
fig. 11

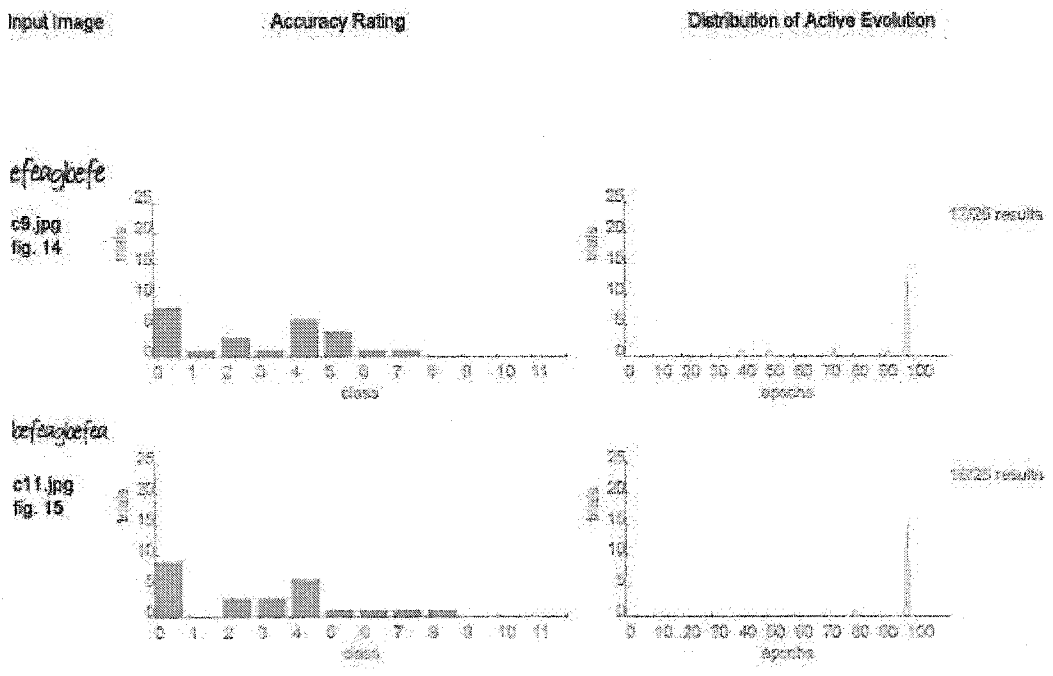




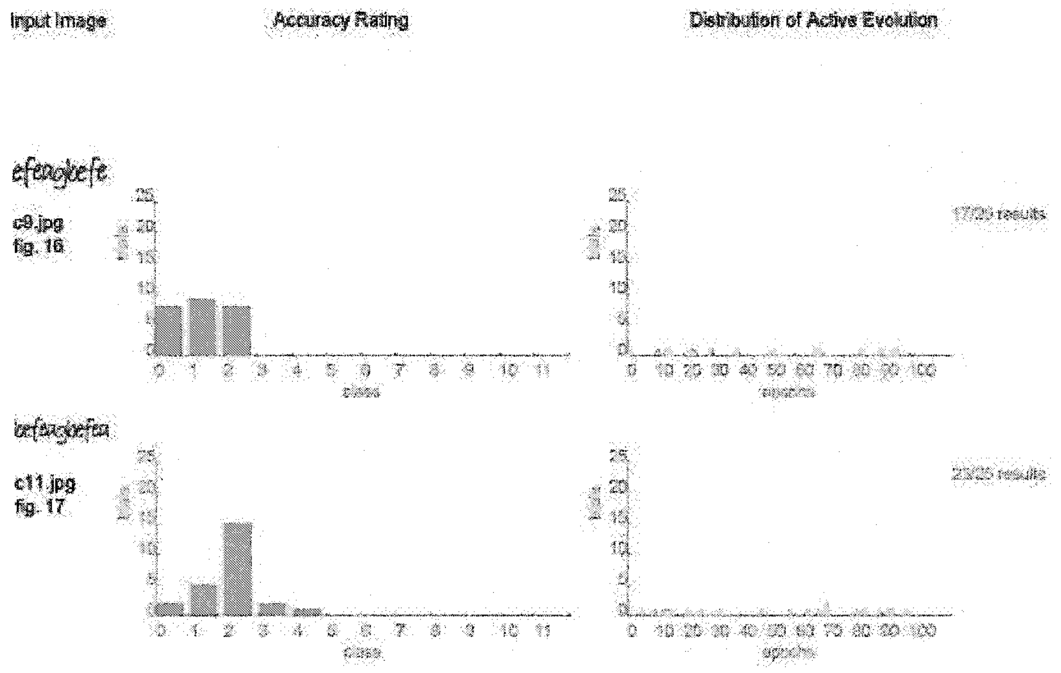
Configuration: Random initialization, full parent search, 100 epoch limit, 10 individuals.



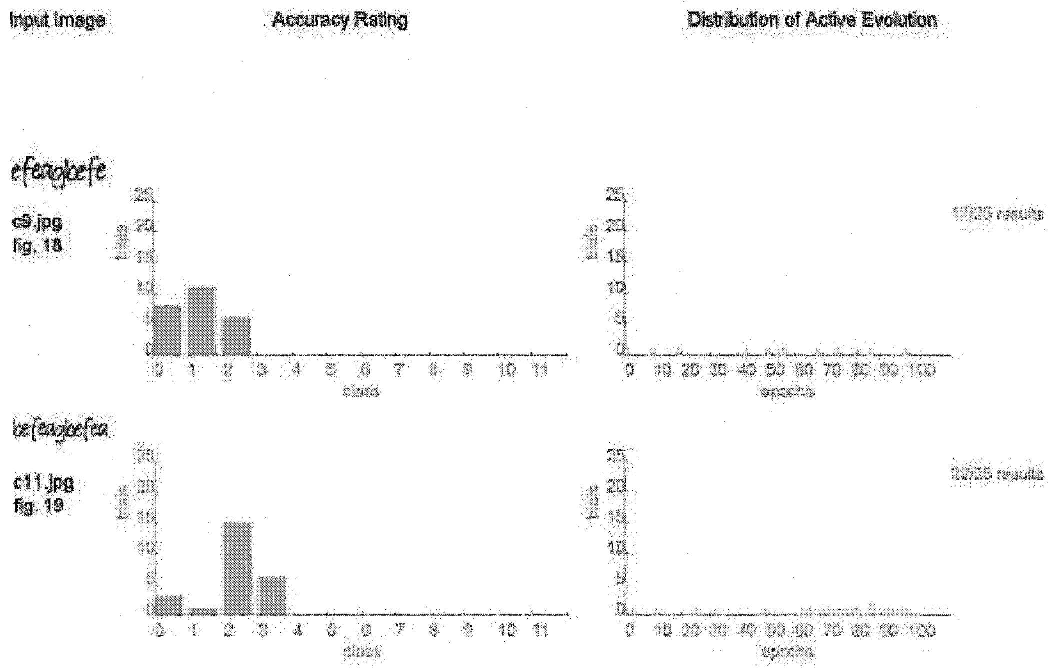
Configuration: Random initialization, random parent search, 100 epoch limit, 10 individuals.



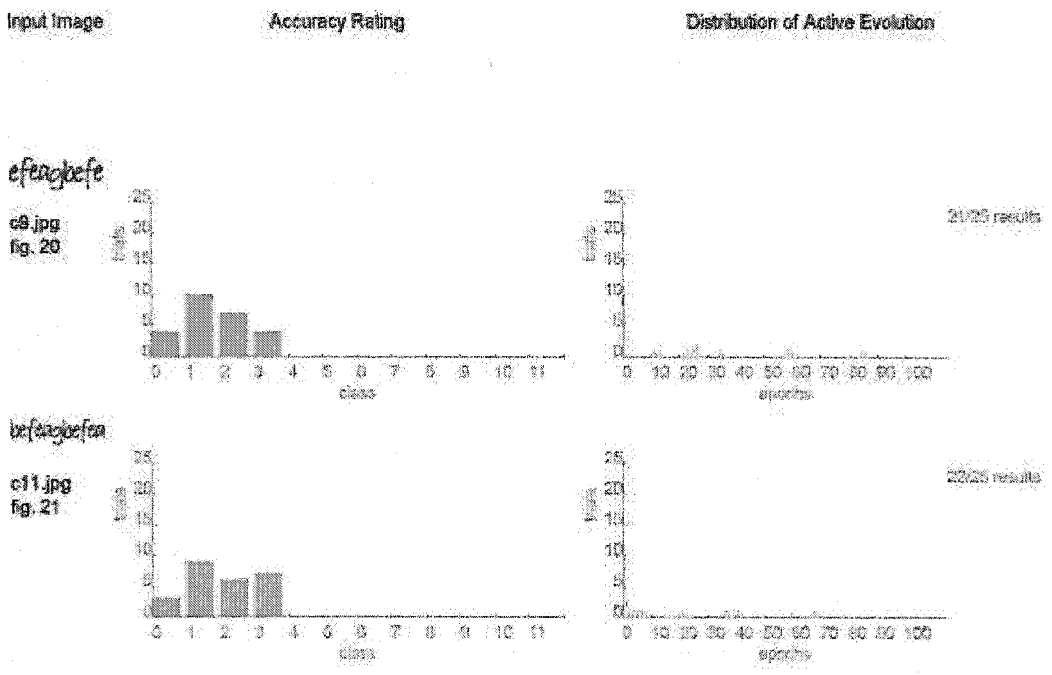
Configuration: Seeded initialization, full parent search, 100 epoch limit, 15 individuals.



Configuration: Seeded initialization, random parent search, 100 epoch limit, 15 individuals.



Configuration: Random initialization, full parent search, 100 epoch limit, 15 individuals.

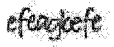


Configuration: Random initialization, random parent search, 100 epoch limit, 15 individuals.

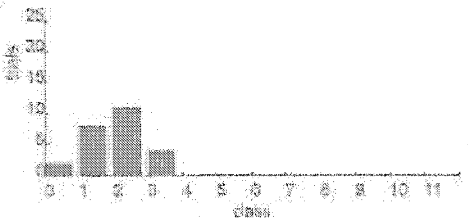
Input Image

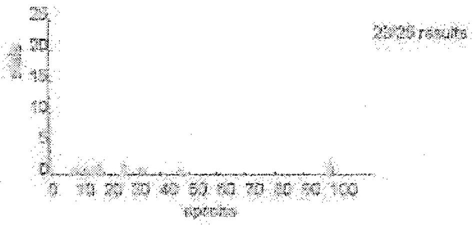
Accuracy Rating

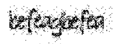
Distribution of Active Evolution



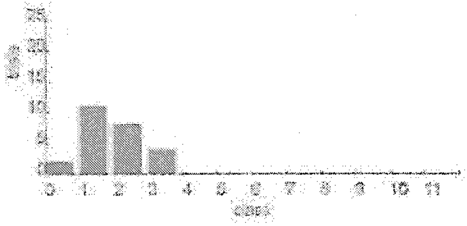
c9.jpg
fig. 22

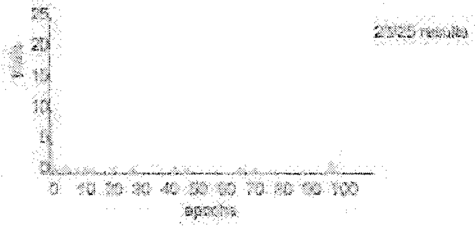






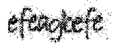
c11.jpg
fig. 23



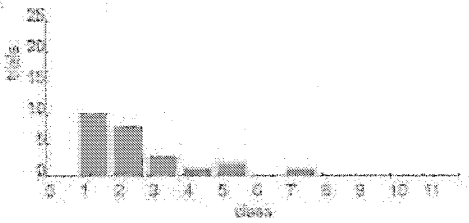


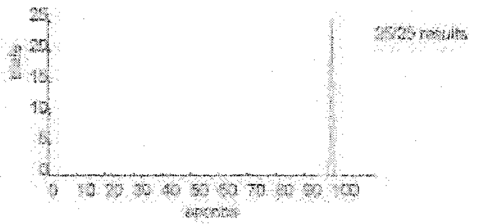
Configuration: Seeded initialization, full parent search, 100 epoch limit, 20 individuals.

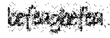
Input Image Accuracy Rating Distribution of Active Evolution



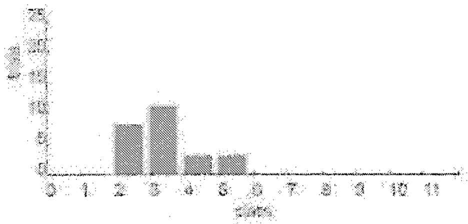
c9.jpg
fig. 24

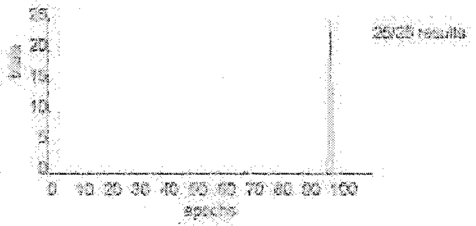






c11.jpg
fig. 25



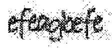


Configuration: Seeded initialization, random parent search, 100 epoch limit, 20 individuals.

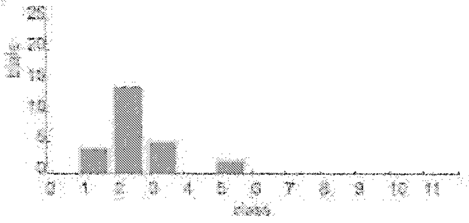
Input Image

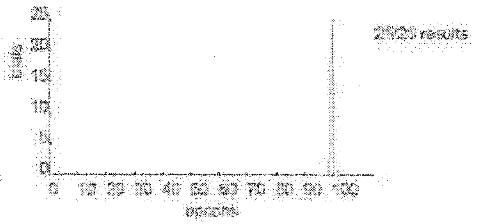
Accuracy Rating

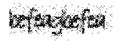
Distribution of Active Evolution



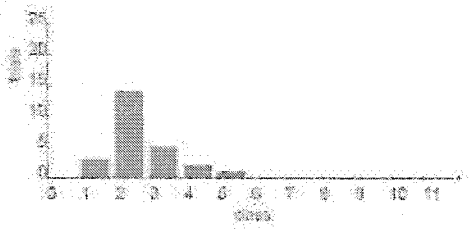
c9.jpg
fig. 26

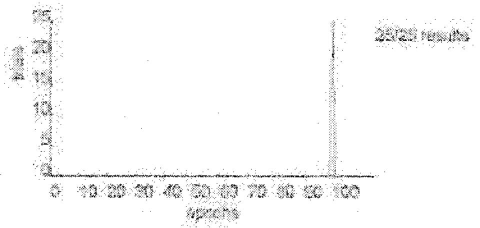




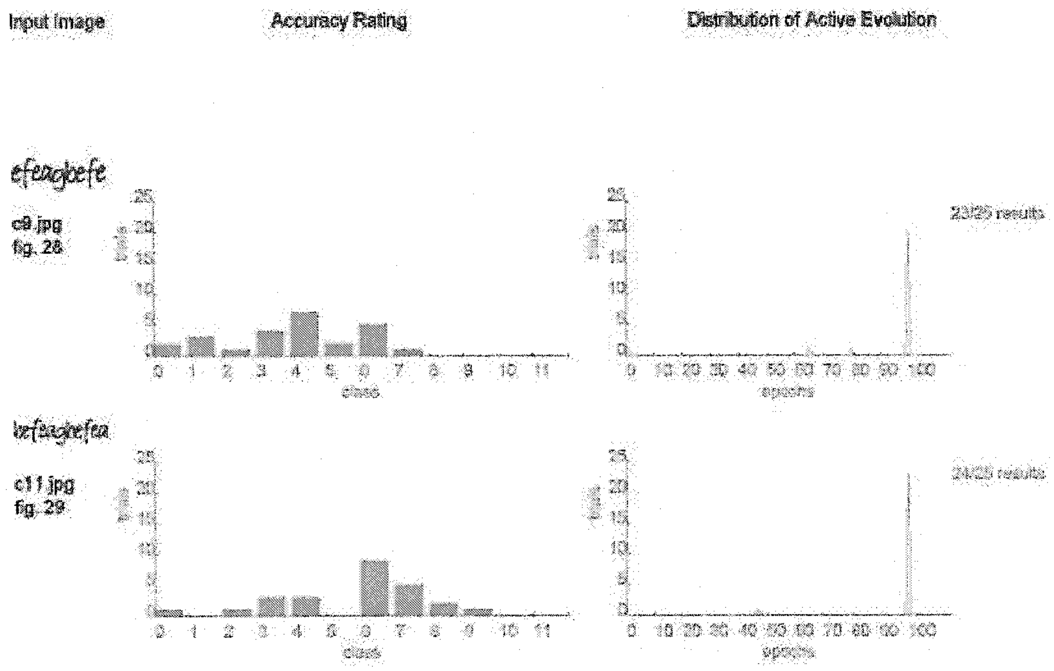


c11.jpg
fig. 27





Configuration: Random initialization, full parent search, 100 epoch limit, 20 individuals.

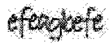


Configuration: Random initialization, random parent search, 100 epoch limit, 20 individuals.

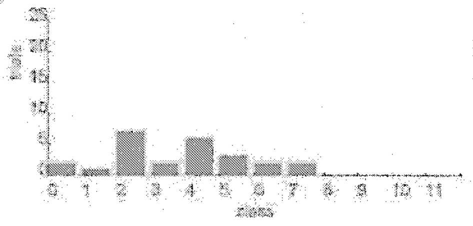
Input Image

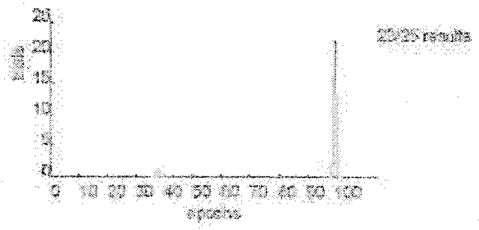
Accuracy Rating

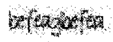
Distribution of Active Evolution



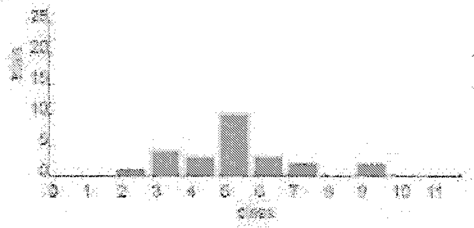
c9.jpg
fig. 30

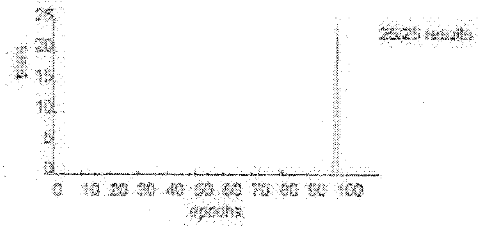




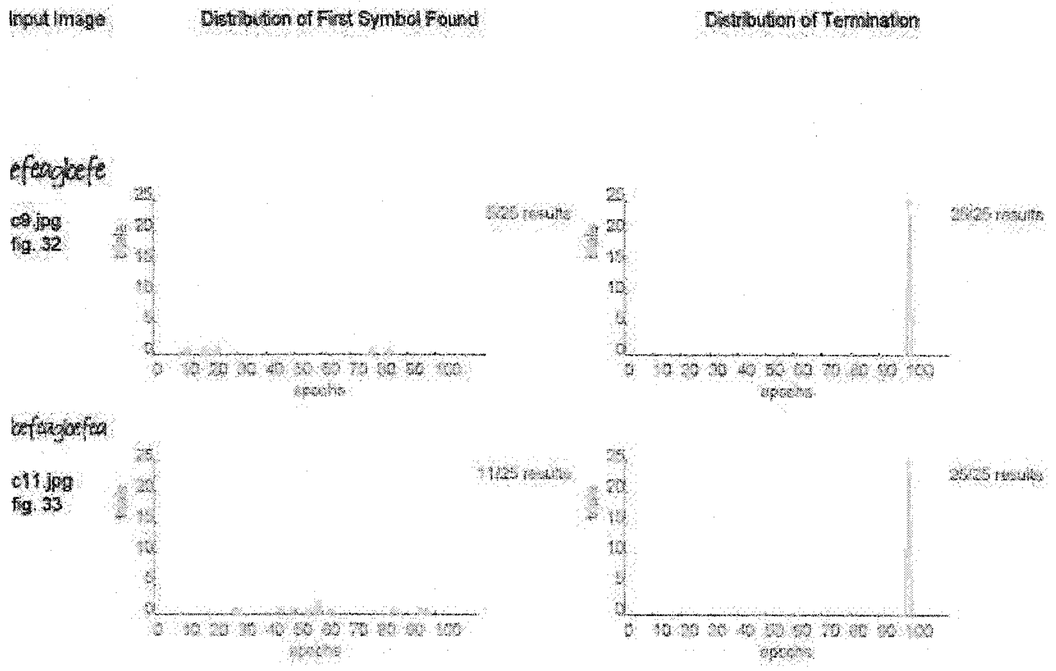


c11.jpg
fig. 31



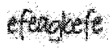


Configuration: Seeded initialization, full parent search, 100 epoch limit, 5 individuals.

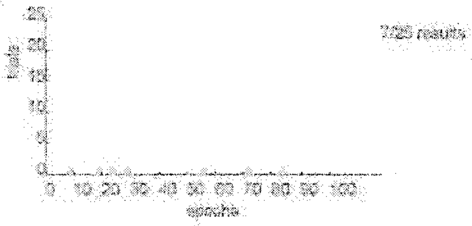


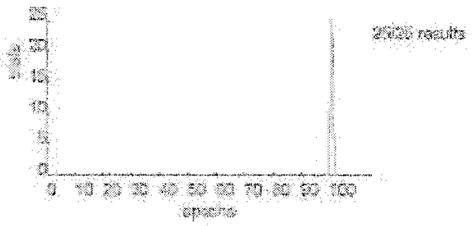
Configuration: Seeded initialization, random parent search, 100 epoch limit, 5 individuals.

Input Image   Distribution of First Symbol Found   Distribution of Termination

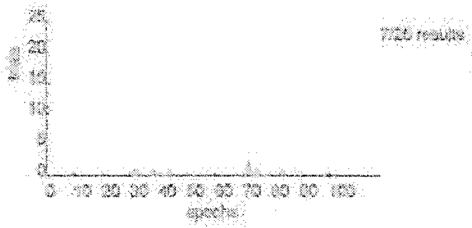


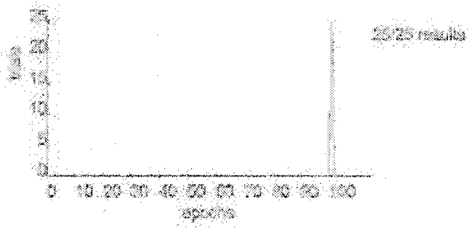
c9.jpg
fig. 34





c11.jpg
fig. 35



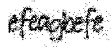


Configuration: Random initialization, full parent search, 100 epoch limit, 5 individuals.

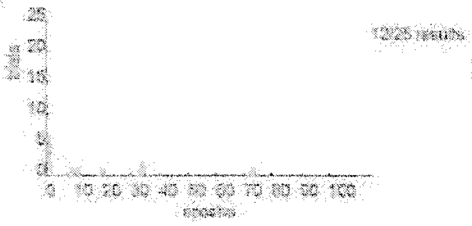
Input Image

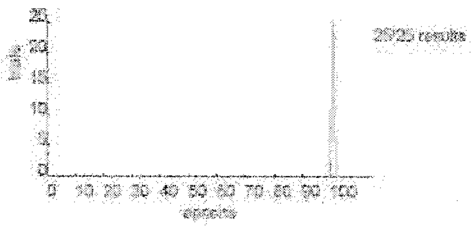
Distribution of First Symbol Found

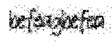
Distribution of Termination



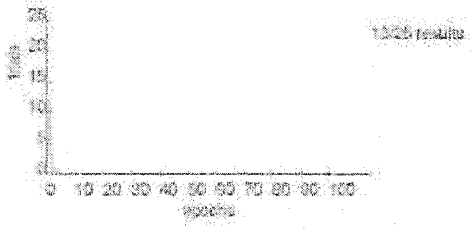
c9.jpg
fig. 36

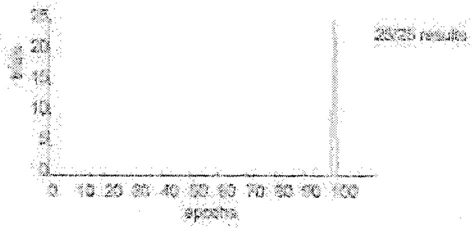






c11.jpg
fig. 37



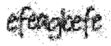


Configuration: Random initialization, random parent search, 100 epoch limit, 5 individuals.

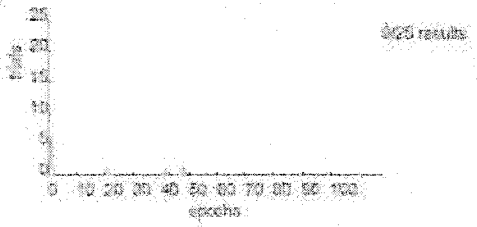
Input Image

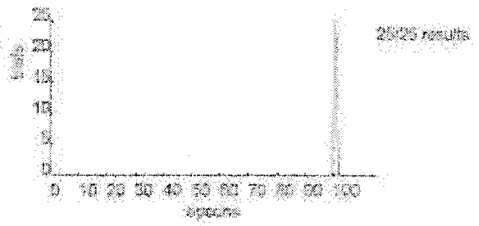
Distribution of First Symbol Found

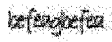
Distribution of Termination



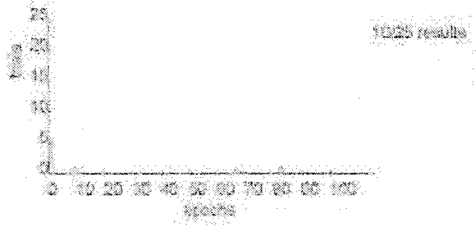
c9.jpg
fig. 38

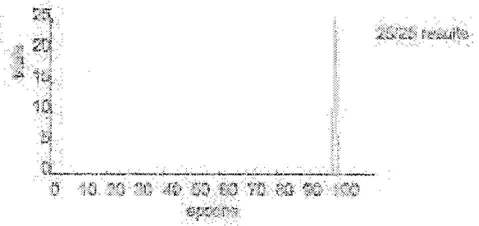






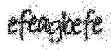
c11.jpg
fig. 39



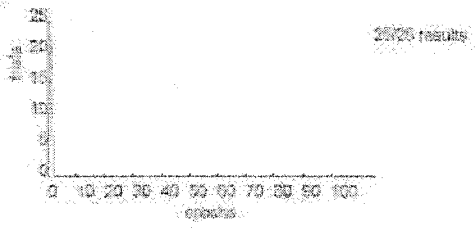


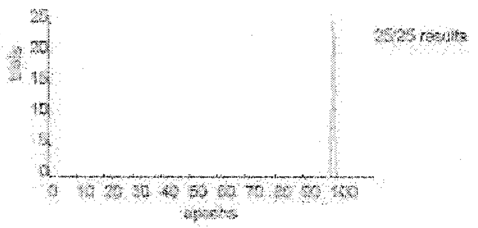
Configuration: Seeded initialization, full parent search, 100 epoch limit, 10 individuals.

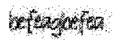
Input Image Distribution of First Symbol Found Distribution of Termination



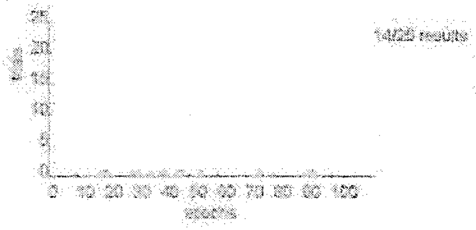
c9.jpg
fig. 40

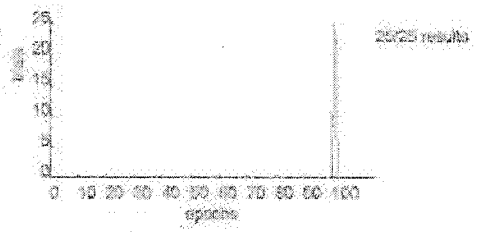




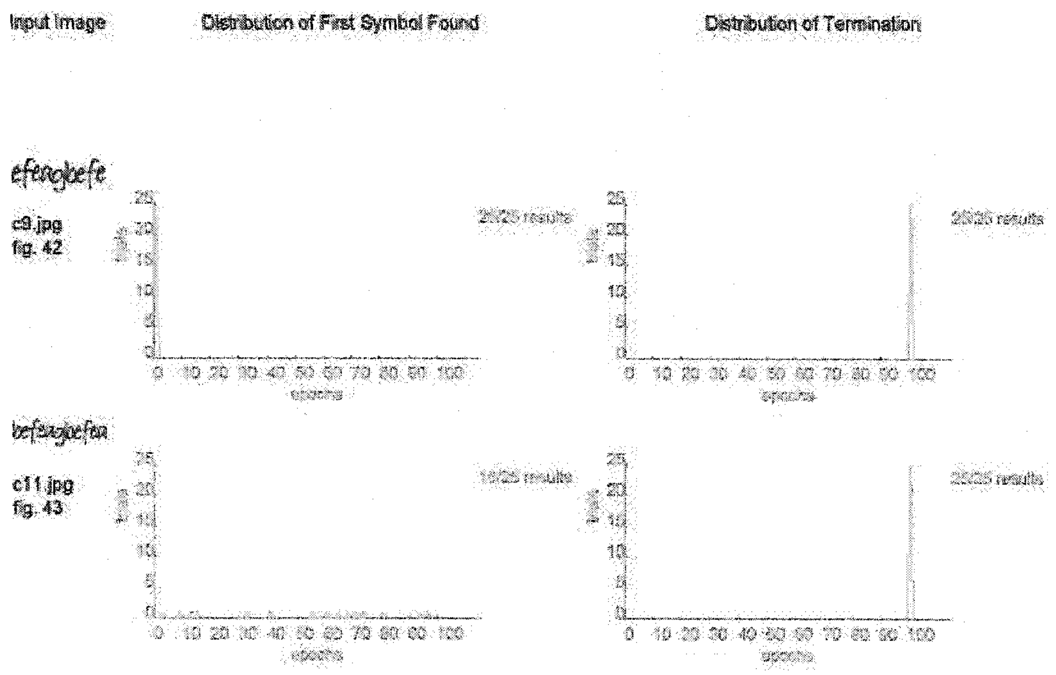


c11.jpg
fig. 41

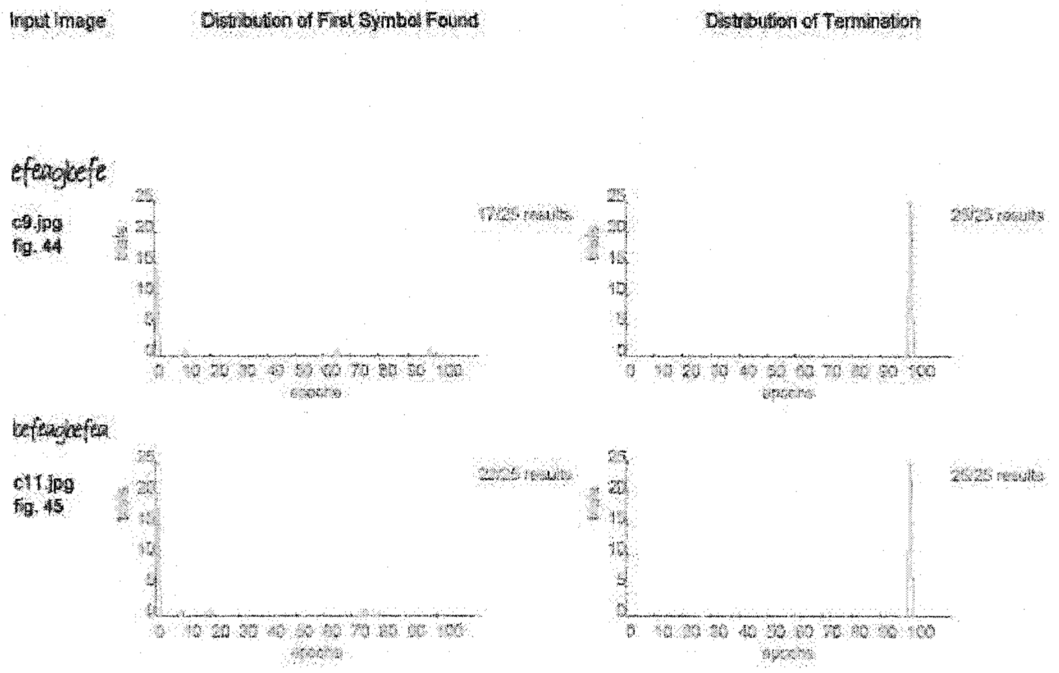




Configuration: Seeded initialization, random parent search, 100 epoch limit, 10 individuals.



Configuration: Random initialization, full parent search, 100 epoch limit, 10 individuals.

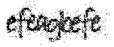


Configuration: Random initialization, random parent search, 100 epoch limit, 10 individuals.

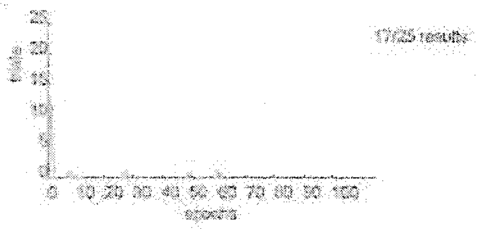
Input Image

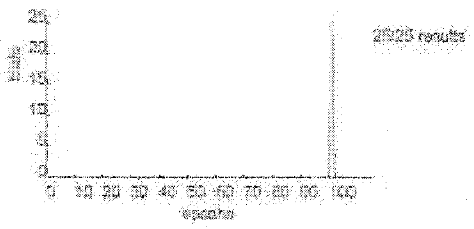
Distribution of First Symbol Found

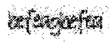
Distribution of Termination



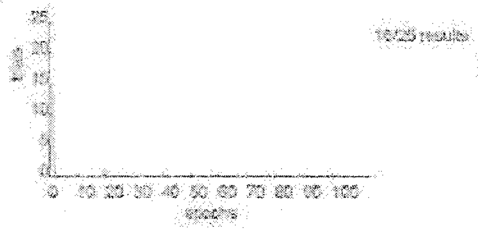
c9.jpg
fig. 46

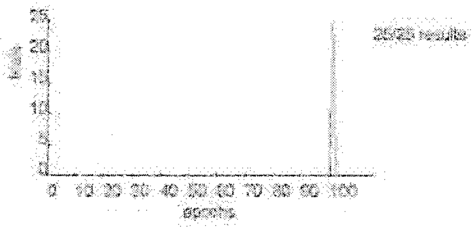




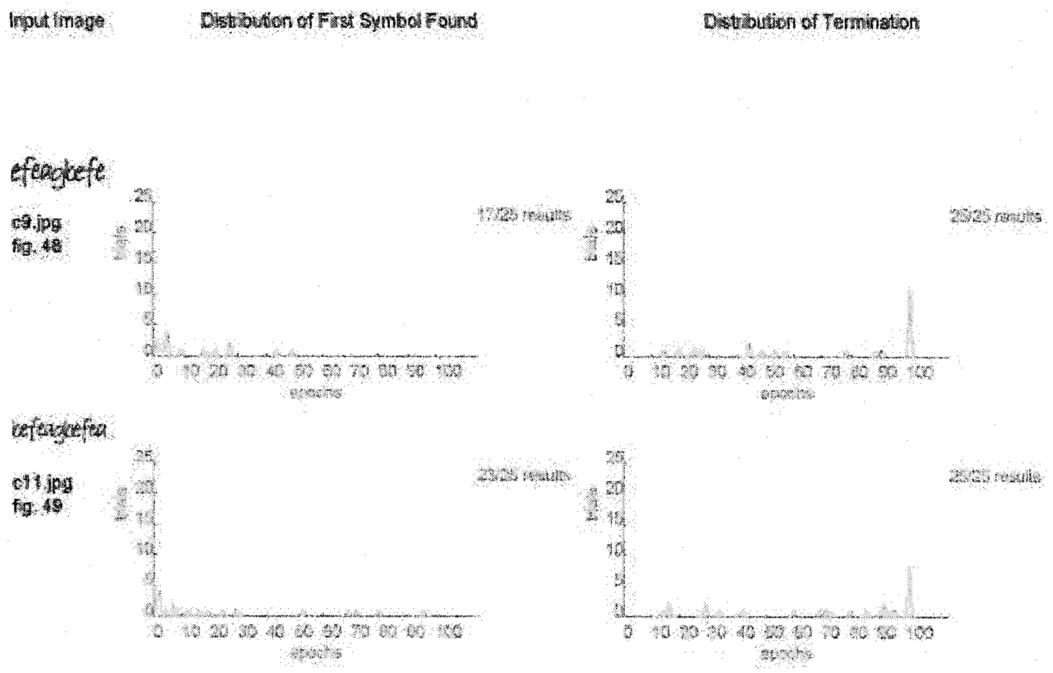


c11.jpg
fig. 47



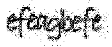


Configuration: Seeded initialization, full parent search, 100 epoch limit, 15 individuals.

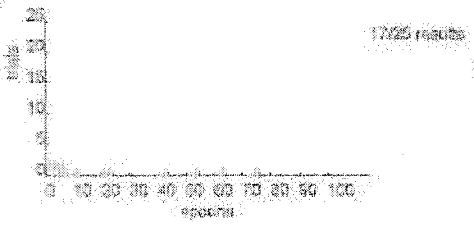


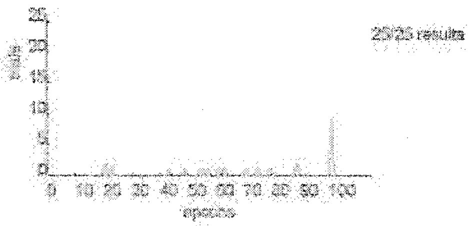
Configuration: Seeded initialization, random parent search, 100 epoch limit, 15 individuals.

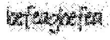
Input Image Distribution of First Symbol Found Distribution of Termination



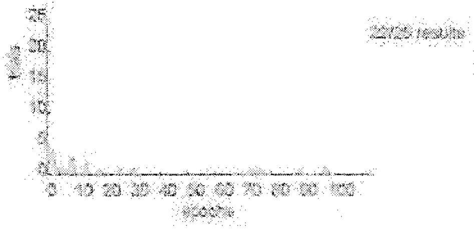
c9.jpg
fig. 50

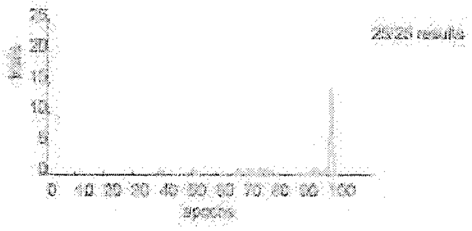






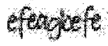
c11.jpg
fig. 51



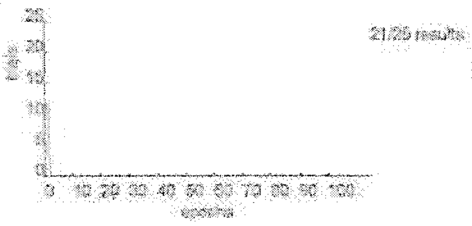


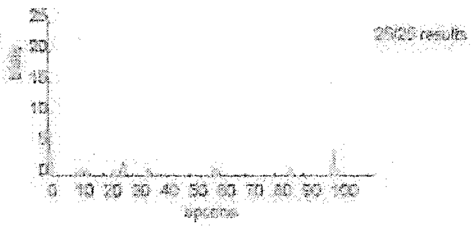
Configuration: Random initialization, full parent search, 100 epoch limit, 15 individuals.

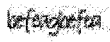
Input Image Distribution of First Symbol Found Distribution of Termination



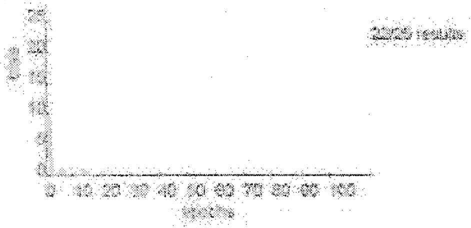
c9.jpg
fig. 52

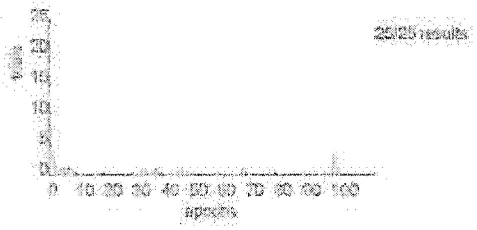






c11.jpg
fig. 53



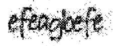


Configuration: Random initialization, random parent search, 100 epoch limit, 15 individuals.

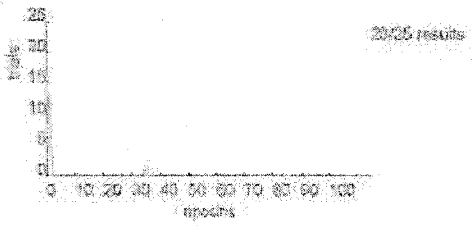
Input Image

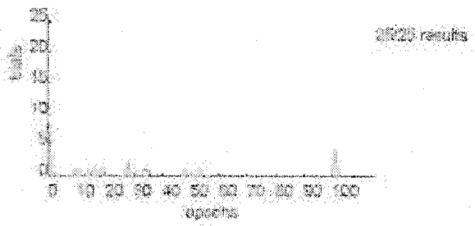
Distribution of First Symbol Found

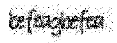
Distribution of Termination



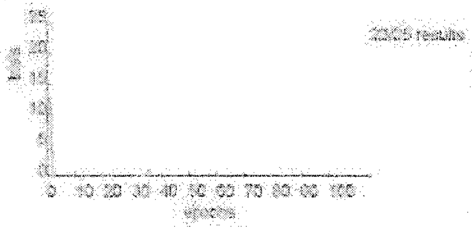
c9.jpg
fig. 54

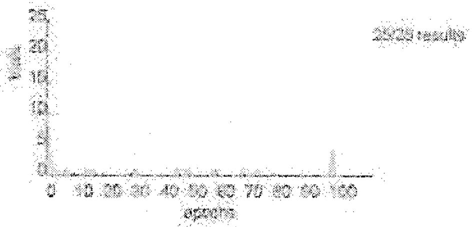






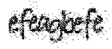
c11.jpg
fig. 55



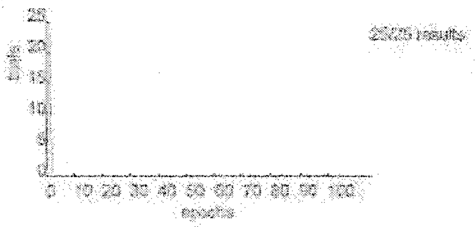


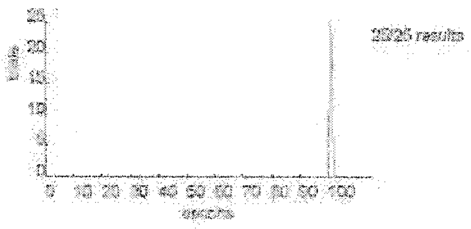
Configuration: Seeded initialization, full parent search, 100 epoch limit, 20 individuals.

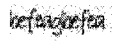
Input Image Distribution of First Symbol Found Distribution of Termination



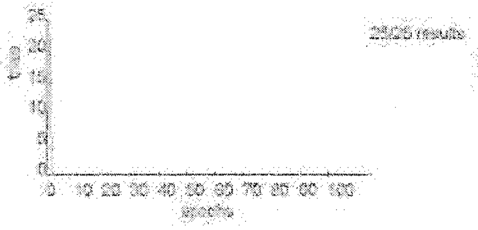
c9.jpg
fig. 56

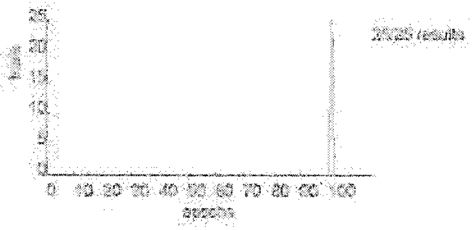






c11.jpg
fig. 57



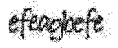


Configuration: Seeded initialization, random parent search, 100 epoch limit, 20 individuals.

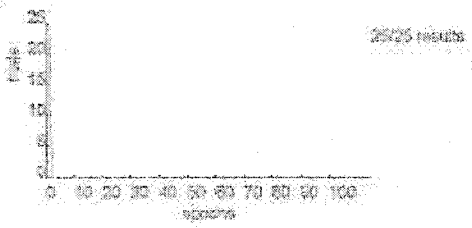
Input Image

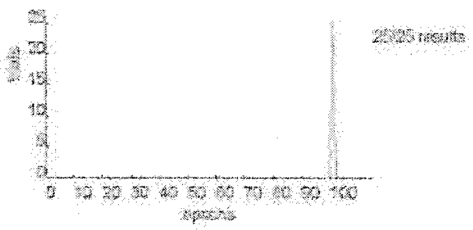
Distribution of First Symbol Found

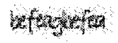
Distribution of Termination



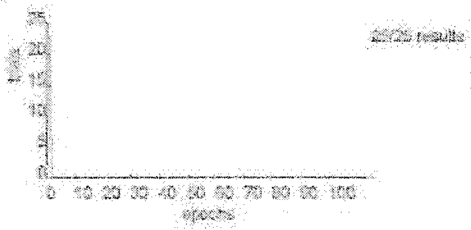
c9.jpg
fig. 58

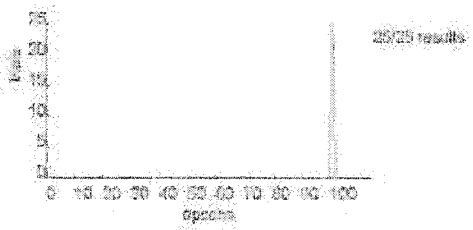






c11.jpg
fig. 59



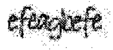


Configuration: Random initialization, full parent search, 100 epoch limit, 20 individuals.

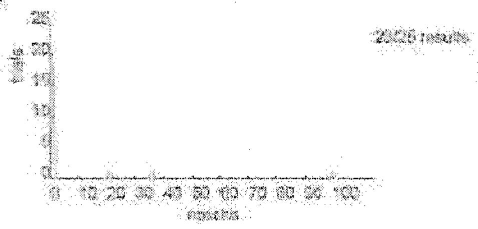
Input Image

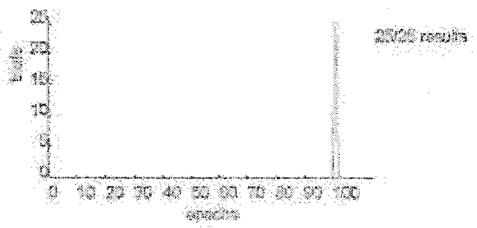
Distribution of First Symbol Found

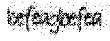
Distribution of Termination



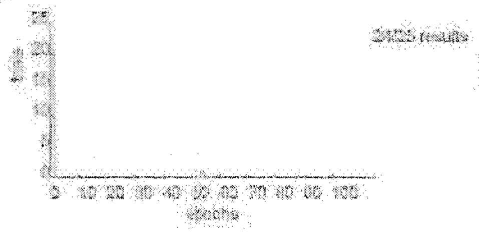
c9.jpg
fig. 60

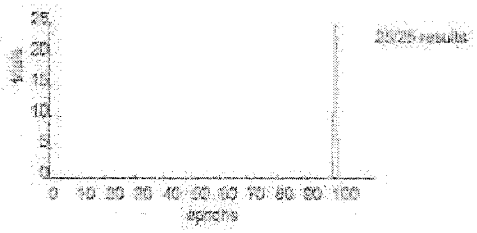






c11.jpg
fig. 61



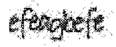


Configuration: Random initialization, random parent search, 100 epoch limit, 20 individuals.

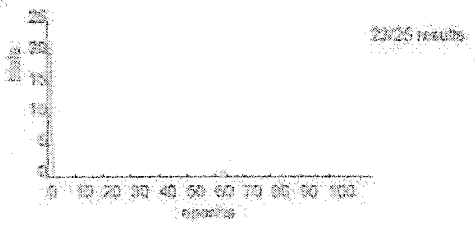
Input Image

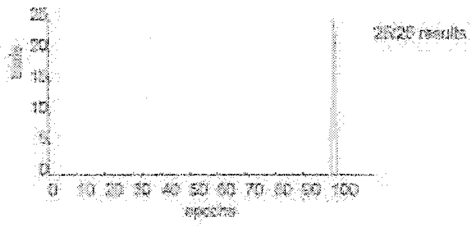
Distribution of First Symbol Found

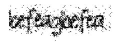
Distribution of Termination



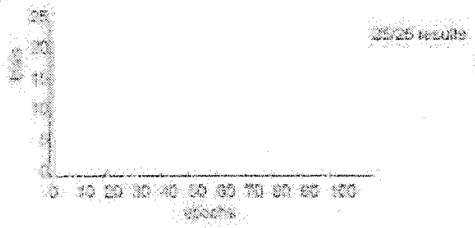
c9.jpg
fig. 62

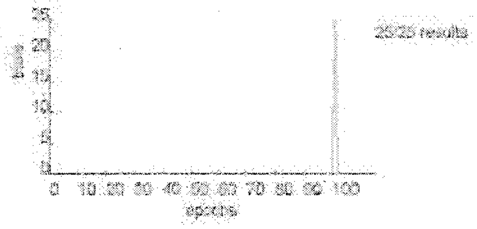




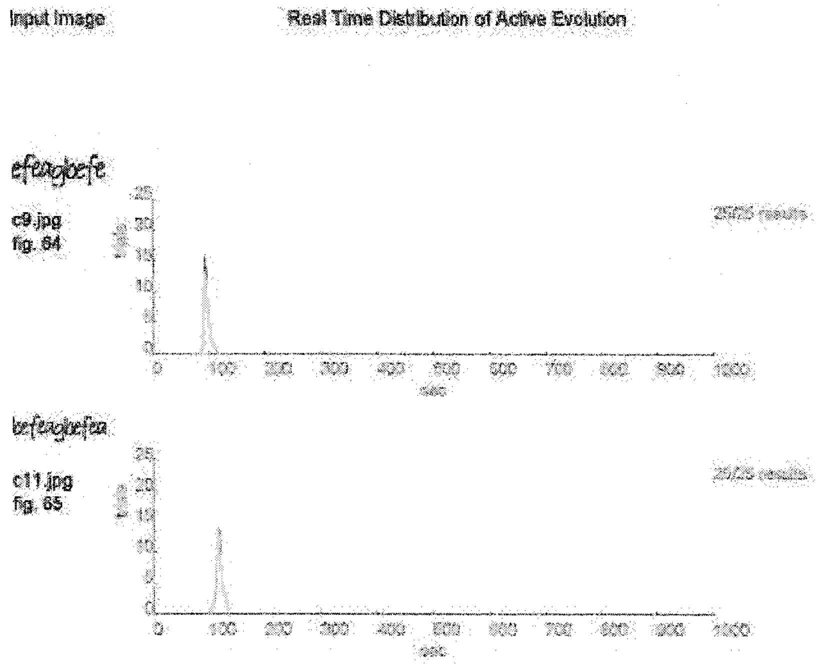


c11.jpg
fig. 63

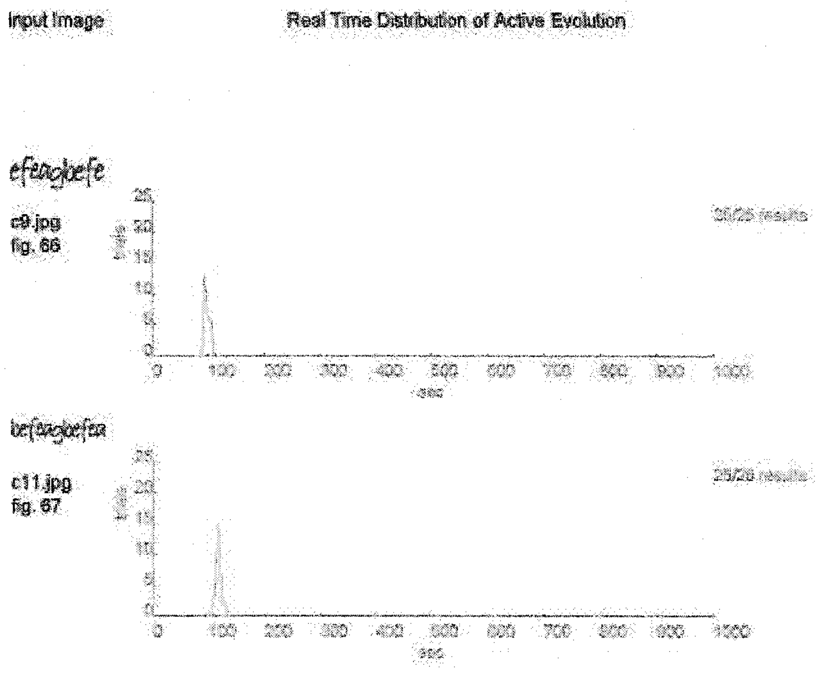




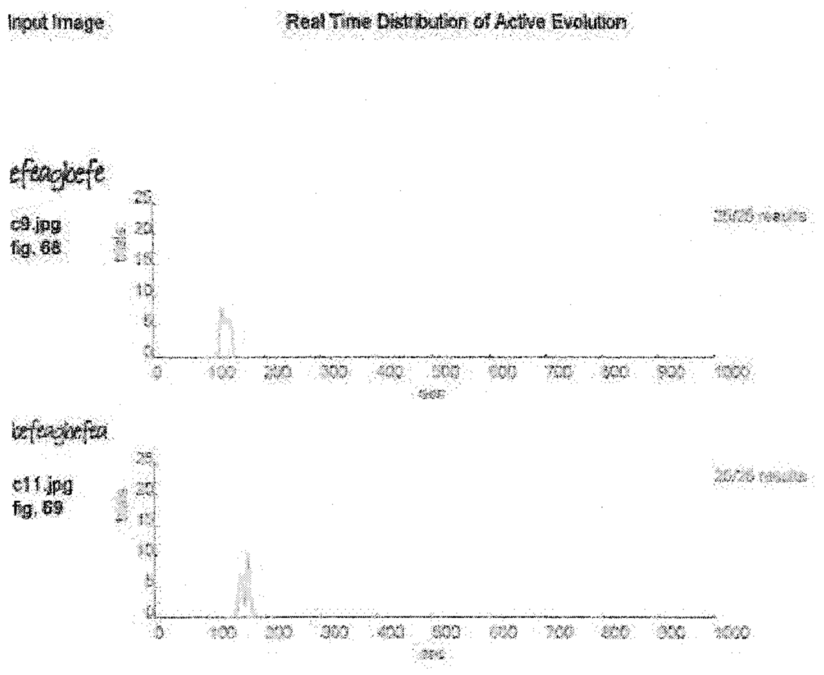
Configuration: Seeded initialization, full parent search, 100 epoch limit, 5 individuals.



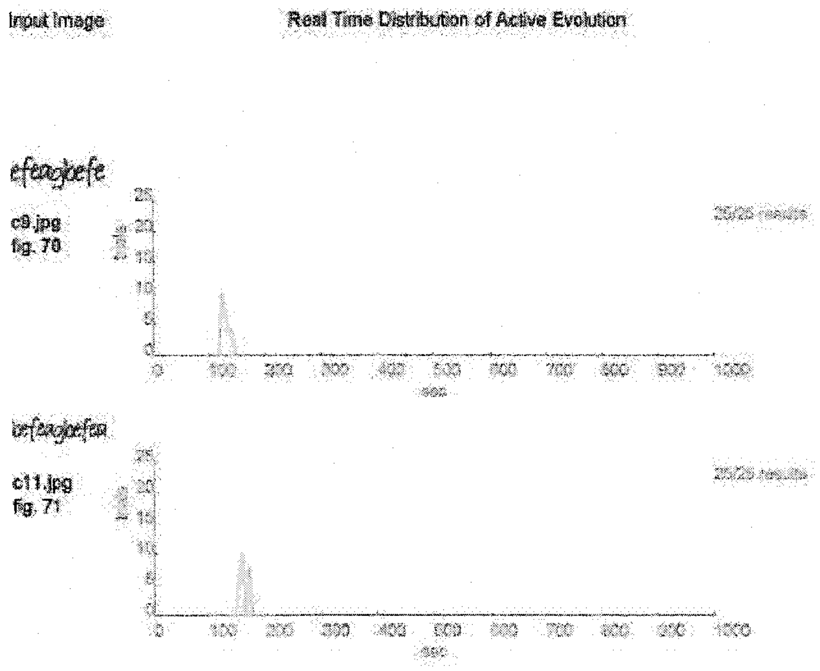
Configuration: Seeded initialization, random parent search, 100 epoch limit, 5 individuals.



Configuration: Random initialization, full parent search, 100 epoch limit, 5 individuals.



Configuration: Random initialization, random parent search, 100 epoch limit, 5 individuals.



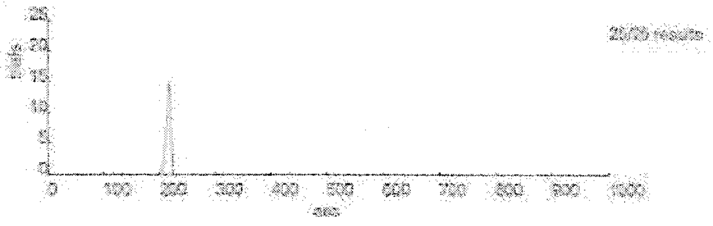
Analysis of Numerous Trials on Each Input Image

Input Image

Real Time Distribution of Active Evolution

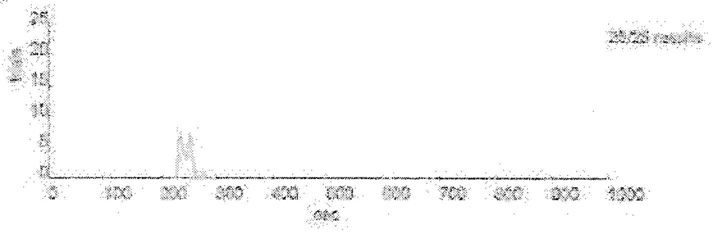
efeagbefe

c9.jpg
fig. 72

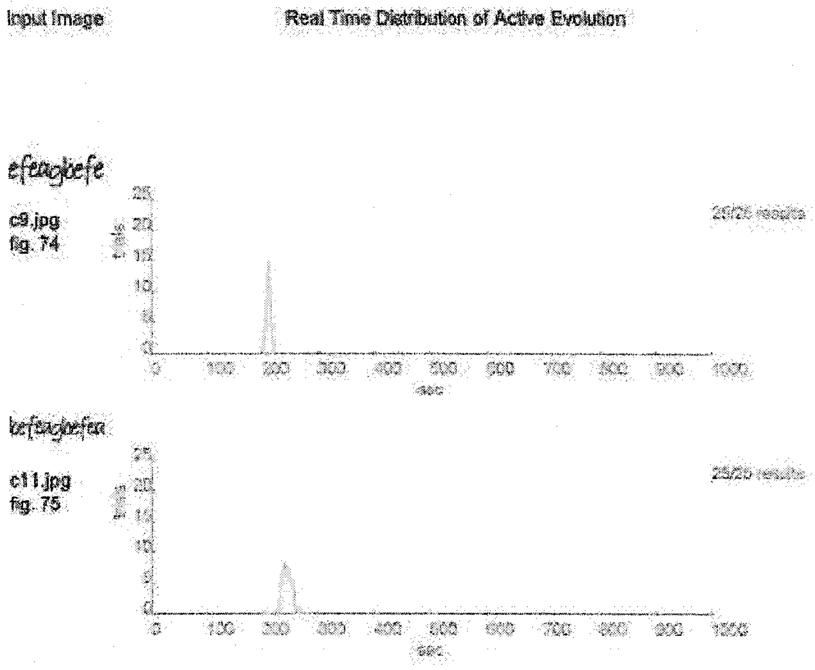


befeagbefea

c11.jpg
fig. 73

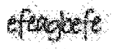


Configuration: Seeded initialization, full parent search, 100 epoch limit, 10 individuals.

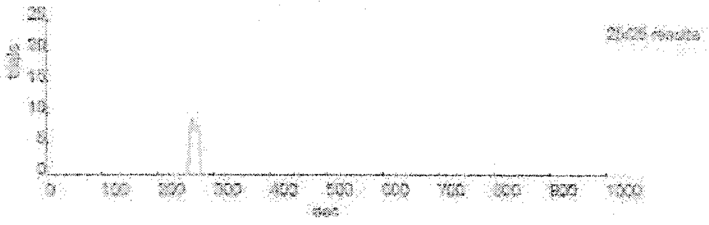


Configuration: Seeded initialization, random parent search, 100 epoch limit, 10 individuals.

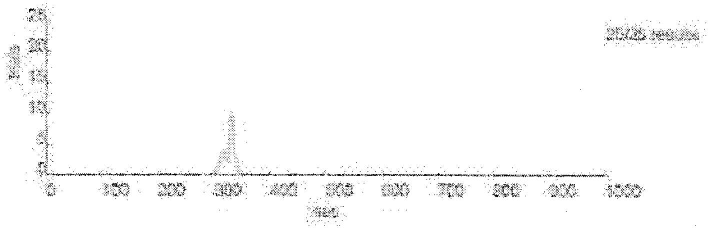
Input Image Real Time Distribution of Active Evolution



c9.jpg
fig. 76

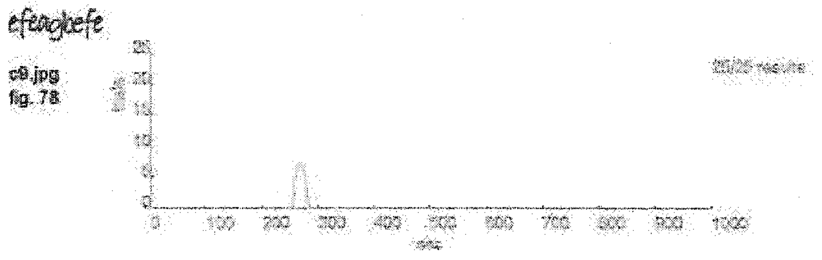


c11.jpg
fig. 77

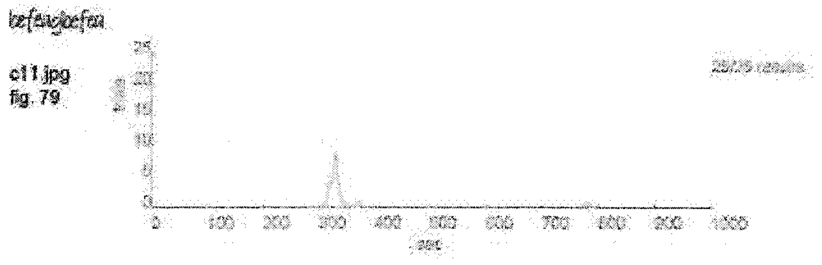


Configuration: Random initialization, full parent search, 100 epoch limit, 10 individuals.

Input Image Real Time Distribution of Active Evolution



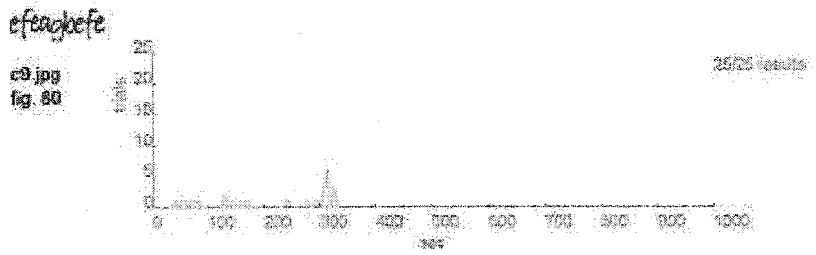
c9.jpg
fig. 78

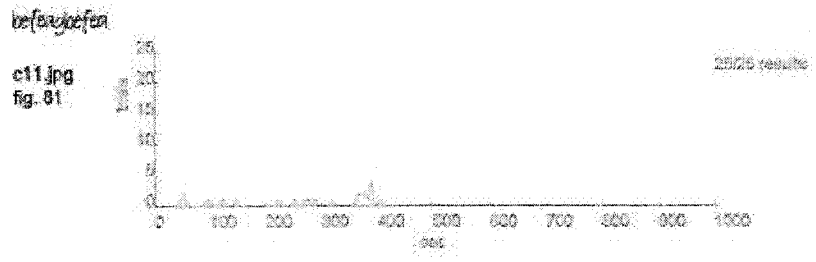


c11.jpg
fig. 79

Configuration: Seeded initialization, full parent search, 100 epoch limit, 15 individuals.

Input Image Real Time Distribution of Active Evolution





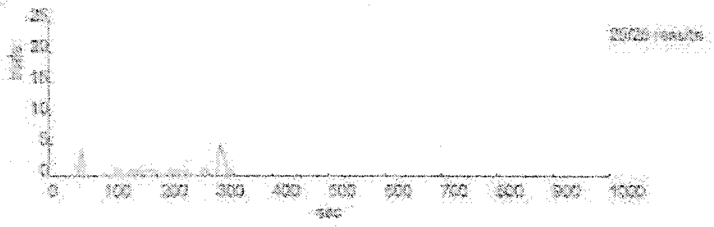
Configuration: Seeded initialization, random parent search, 100 epoch limit, 15 individuals.

Input Image

Real Time Distribution of Active Evolution

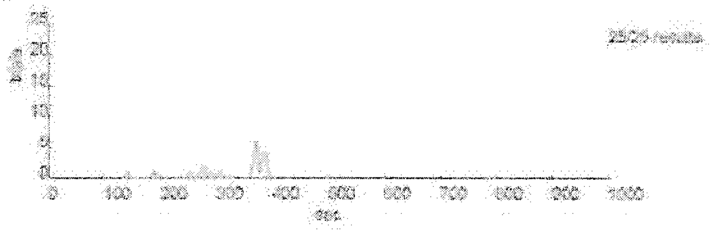
efeagbefe

c9.jpg
fig. 82



befeagbefea

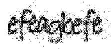
c11.jpg
fig. 83



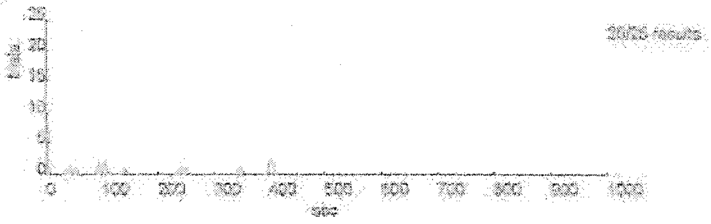
Configuration: Random initialization, full parent search, 100 epoch limit, 15 individuals.

Input Image

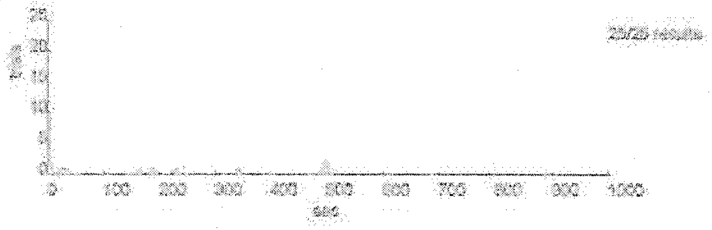
Real Time Distribution of Active Evolution



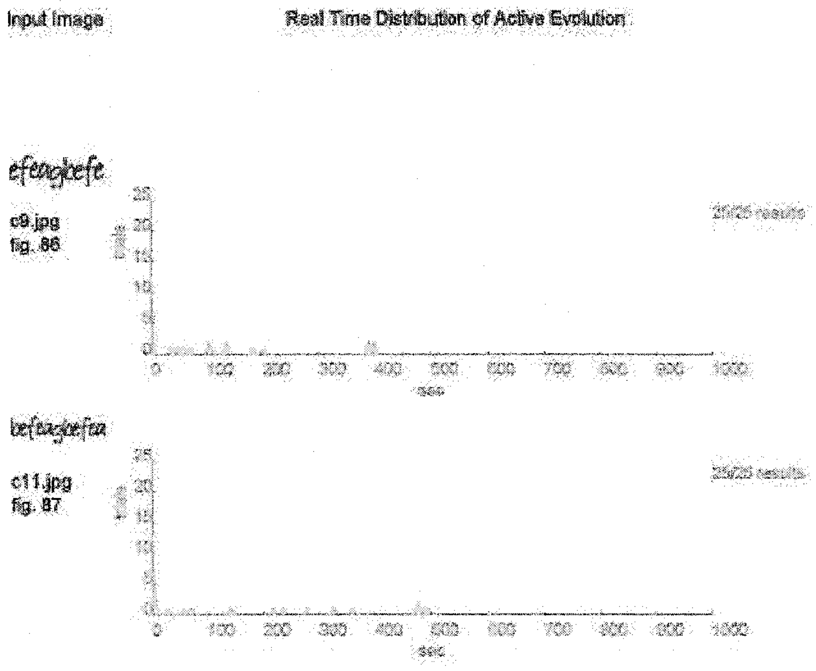
c9.jpg
fig. 84



c11.jpg
fig. 85



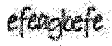
Configuration: Random initialization, random parent search, 100 epoch limit, 15 individuals.



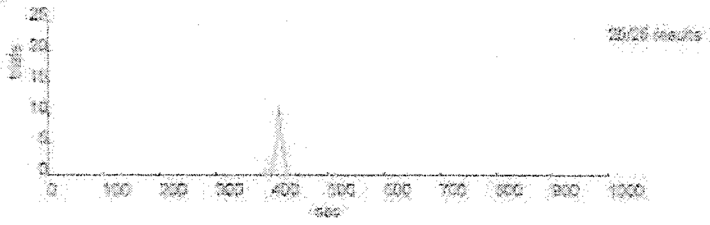
Configuration: Seeded initialization, full parent search, 100 epoch limit, 20 individuals.

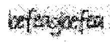
Input Image

Real Time Distribution of Active Evolution

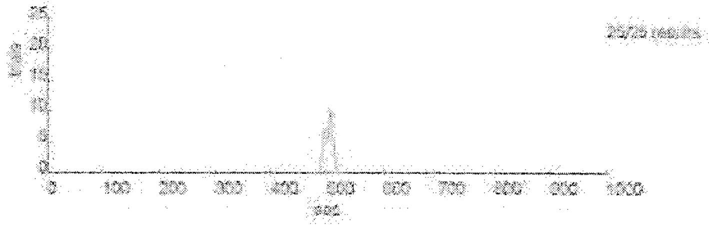


c9.jpg
fig. 88



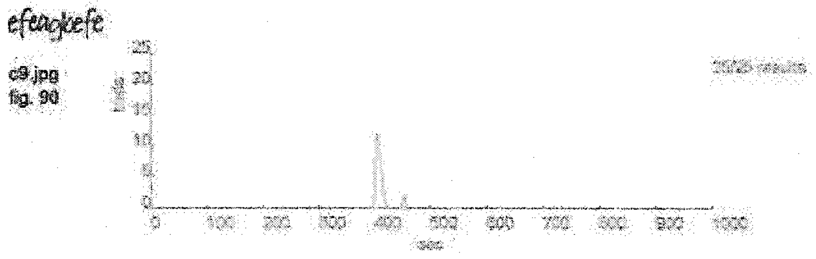


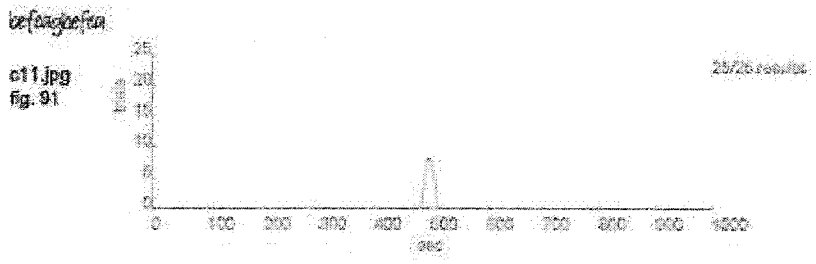
c11.jpg
fig. 89



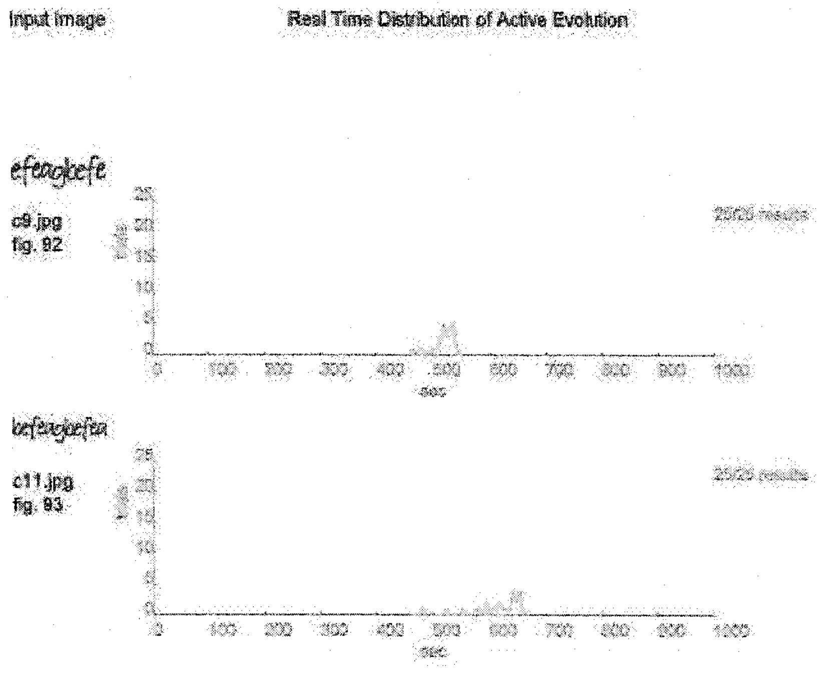
Configuration: Seeded initialization, random parent search, 100 epoch limit, 20 individuals.

Input Image Real Time Distribution of Active Evolution





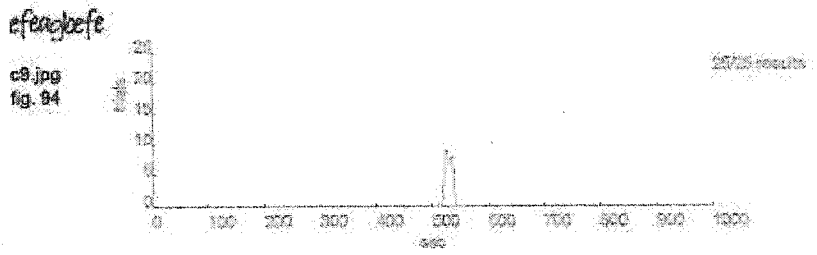
Configuration: Random initialization, full parent search, 100 epoch limit, 20 individuals.

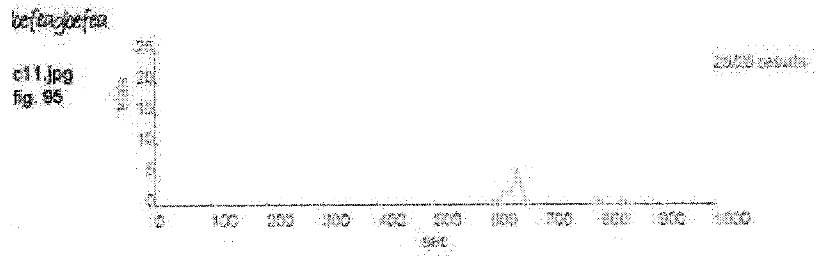


Configuration: Random initialization, random parent search, 100 epoch limit, 20 individuals.

Input Image

Real Time Distribution of Active Evolution


c9.jpg
fig. 94
25/25 results
sec


c11.jpg
fig. 95
25/25 results
sec