Robustness and Performance of Deep Reinforcement Learning

Raid Rafi Omar Al-Nima[1], Tingting Han[2], Saadoon Awad Mohammed Al-Sumaidaee[3], Taolue Chen[2] and Wai Lok Woo[4]

[1] *Technical Engineering College of Mosul, Northern Technical University, Iraq*
[2] *Department of Computer Science and Information Systems, Birkbeck, University of London*
[3] *College of Engineering, Al-Mustansiriyah University, Baghdad, Iraq*
[4] *Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, UK*

Abstract

Deep Reinforcement Learning (DRL) has recently obtained considerable attentions. It empowers Reinforcement Learning (RL) with Deep Learning (DL) techniques to address various difficult tasks. In this paper, a novel approach called the *Genetic Algorithm of Neuron Coverage* (GANC) is proposed. It is motivated for improving the robustness and performance of a DRL network. The GANC uses Genetic Algorithm (GA) to maximise the Neuron Coverage (NC) of a DRL network by producing augmented inputs. We apply this method in the self-driving car applications, where it is crucial to accurately provide a correct decision for different road tracking views. We evaluate our method on the SYNTHIA-SEQS-05 databases in four different driving environments. Our outcomes are very promising - the best driving accuracy reached 97.75% - and are superior to the state-of-the-art results.

Keywords: `Deep Reinforcement Learning, Genetic Algorithm, Neuron Coverage, Road Tracking`

1. Introduction

Deep Reinforcement Learning (DRL) is a type of Machine Learning (ML). It is essentially a combination between the Reinforcement Learning (RL) concept and Deep Learning (DL) network [1]. In other words, DL is structured within the RL to address Markov Decision Process (MDP) issues [2]. DL network is an advanced version of *Artificial Neural Networks* (ANN). It is currently widely exploited in different fields such as image processing [3, 4], signal processing [5] and object tracking [6].

For safety-critical application of autonomous driving, it is extremely important to have a robust model or method, as any inappropriate behaviours can cause serious risks, damages or even catastrophes [7]. A highly accurate model may still fail easily - a slight modification in the image (e.g., colour change to a few pixels) may sometimes cause completely different classification results. For example, in self-driving cars, if a car "sees" a stop sign, it should recognise it and stop. But even a small black and white sticker on the sign could "fool" the artificial network model and prevent the car from stopping [8].

As a result, when evaluating the quality of a model, we should not only condier *performance*, but also *robustness*. Performance measures the overall accuracy of a model (accuracy being ration between the number of correctly classified cases and total number of cases), while the robustness is the ability to accept wide range of testing cases.

In this paper, we propose to increase the performance and robustness of a Deep Reinforcement Learning for Road Tracking (DRL-RT) [6] in different self-driving car environments. The robustness is enhanced by establishing new input samples that can correct wrong (or faulty) tracking predictions. Technically, the robustness is measured by the neuron coverage (NC) - the ratio of the activated neurons in a network [9]. The increasing NC in principle implies the increasing number of activated neurons in a network and it would produce high outcome values and enhance the performance of the network in general. In short, we would like to have a highly accurate model which would work well on faulty cases.

Genetic Algoritm GA is a method for solving both constrained and unconstrained optimisation problems. GA simulates the behaviour of human genes (selection, crossover and mutation of individual chromosomes) to optimise a specific fitness function. A fitness function evaluates how close a given solution is to the optimal solution of the desired problem. GA tries to search the neighbourhood for the initial solutions by heuristics method to get an optimal solution for the problem. The crossover and mutation operators guarantee that one can improve the initial solutions to get a global optimal solution [10, 11, 12]. It is suggested that the GA technique can play an effective role to optimise the NC value of the employed DRL network.

To apply the GA to our setting, we make the NC metric to be a fitness function and try to maximise its value, and then enhance the performance as a result. In the GA process, the mutation and crossover operations will produce more training samples as augmented inputs. Those training inputs will identify the incorrectly trained test cases and have very high NC value and thus will increase the robustness of the model. The newly generated training inputs are used together with the original training set in our proposed road tracking model. It has been observed that significant enhancements in performance could be obtained in the testing phase.

To wrap up, we are presenting the *Genetic Algorithm of Neuron Coverage* (GANC) approach which can enhance the performance and robustness of a DRL network. As a case study, we applied the GANC to a DRL-RT network [6] in autonomous driving scenarios.

After the introduction, the remaining sections in this paper are organised as follows: Section 2 highlights prior studies that are related to our work. Section 3 provides the theoretical background of the GANC. Section 4 illustrates the results and Section 5 concludes the main observations in this work.

2. Literature Review

Different metrics existed in several studies to measure performance and/or robustness of deep networks. We summarised these studies into two categories - those using NC as (part of) their metrics or those without. In addition, their criteria of applying the robustness and/or performance are illustrated as briefly highlighted in Table 1. Moreover, GA strategies are also surveyed this section.

Table 1: The different definitions of robustness and/or performance in literature work

	Reference	Performance Criterion	Robustness Criterion
Without NC	Sun *et al.* [13]	Percentage of covered neuron pairs	Coverage testing and adversarial percentage
	Ma *et al.* [14]	Percentage of accuracy	Applying mutation techniques and analysing the relationships between the testing data and decision boundary
	Bai *et al.* [15]	Accuracy under various training times	—
With NC	Pei *et al.* [9]	Execution time for producing difference-inducing inputs and NCs of the produced tests	DeepXplore
	Pei *et al.* [16]	Time	Employing various transformation functions, then retraining the transformed images
	Ma *et al.* [17]	—	Applying a comprehensive set of coverage criteria
	Tian *et al.* [18]	—	Employing various transformation functions, then retraining the transformed images
	Sun *et al.* [7]	—	Robustness oracle

2.1. Literature without neuron coverage

Sun *et al.* suggested four DL network measurement criteria based on the Modified Condition/Decision Coverage (MC/DC) [19], these are sign-sign cover, distance-sign cover, sign-value cover and distance-value cover. A large dataset was suggested to be used for calculating the percentage of covered neuron pairs as performance. Whereas, the robustness was measured by the coverage testing and adversarial percentage [13].

Ma *et al.* proposed DeepMutation method for DL network mutation testing. This work focused on applying different types of mutations on a DL network. The percentage of accuracy was used to evaluate the performance. Whilst, applying mutation techniques and analysing the relationships between the testing data and decision boundary were utilised to assess the robustness [14].

Bai *et al.* generated adversarial samples for white-box Deep Q Network (DQN) in terms of pathfinding training. The authors exploited a DQN to automatically find paths of a robot by searching for an optimal (shortest) path. In addition, noise obstacles were constructed and tested for the pathfinding. The robot could not follow the optimal path as the training efficiency was reduced.

Only performance was considered here by computing the accuracy under various training times [15].

2.2. Literature with neuron coverage

Pei *et al.* established Deepxplore method for automatic whitebox testing. The DeepXplore has complicated architecture as multiple DL networks have to be used. Seeds of inputs were required from the testing samples. Determined threshold was suggested to detect the active neurons. The performance was evaluated for the DeepXplore by exploiting two metrics: the execution time for producing difference-inducing inputs and the NCs of the produced tests. On the other hand, the DeepXplore itself was proposed as a robustness method [9].

Similarly, Pei *et al.* employed fixed threshold to explore the effective DL neurons. In this work a VERIVIS framework was adopted as a blackbox model. The authors described the ability of determining safety DL properties from various attackers. This work was focused on a blackbox testing. Time was considered as a performance metric. In contrast, the robustness was enhanced by employing various transformation functions, and then retraining the transformed images [16].

Ma *et al.* presented a DeepGauge model for thorough gauging the robustness of DL networks. This method considers two levels of coverage criteria - at neuron-level and at layer-level. For the neuron-level coverage criteria, various coverages were denoted: the k-multisection coverage, the strong neuron activation coverage and the neuron boundary coverage. For the layer-level coverage criteria, multiple coverages were defined: the Top-k NC and the Top-k neuron patterns. Testing samples were considered for all the above criteria. Only the robustness was considered in this paper [17].

Tian *et al.* proposed Deeptest model as an automatic DL network testing method for autonomous driven cars. In this study it has been illustrated that increasing NCs would increase the safety-critical of the DL system. DeepTest investigated many erroneous cases that could lead to series driving cars collisions. In this paper, only robustness was considered, which can be enhanced by applying different types of image transformation functions, then retraining the resulted transformed images [18]. In this way, different corner cases would be accepted.

Sun *et al.* explained Concolic testing for DL systems called DeepConcolic. It is capable to collect coverage requirements as inputs with the heuristic executions. Different types of coverage areas were employed. These are the activation patterns to determine the activation ReLU nodes, formalising test coverage criteria by exploiting a fragment of Quantified Linear Arithmetic over Rationals (QLAR) and test coverage metrics to be utilised as a proxy metric for the safety confidence. Furthermore, five different specific coverage requirements were applied. These are the Lipschitz continuity [20], NC [9], MC/DC [19] and neuron boundary coverage [17]. The DeepConcolic also provided adversarial testing samples. A robustness oracle was suggested in this work as a defensive method for checking the robustness [7].

2.3. Literature with genetic algorithm

Mangano provided fundamentals of the GA and in addition illustrated the history, theory, application, implementation and future expectations of GAs [21].

Mishra *et al.* reviewed another version of the GA known as Multi-Objective Genetic Algorithm (MOGA). The MOGA concentrates on employing more than one fitness function - at least two fitness functions are required. The GA here can deal with more than a single search space at a time [22].

Vuolio *et al.* explained a GA based model selection for the recognition of carbide-based hot metal desulfurisation. A single hidden layer feedforward neural network was utilised. A simultaneous variable optimisation and selection for the number hidden neurons was applied in the GA [23].

Połap suggested an adaptive method for the combinations between a cascade of the convolutional classifiers and GA. Analyses of microscopy images were focused. GA was exploited as an indicator for the classifier selection of a neural network [24].

Leonori *et al.* illustrated optimization strategies by GAs. Microgrid energy management systems were studied. In fact, a microgrid fuzzy logic is approached. A combination between the GA and Fuzzy Inference System (FIS) was investigated in terms of optimization [25].

2.4. Contribution of this paper

From the literature review, it can be seen that there is no study yet on the DRL that can measure the robustness *and* then enhance the performance. In this paper, a novel approach termed the GANC is designed based on the GA and NC of a DRL. The proposed GANC optimises a robustness measurement called the NC by applying the GA. The outcomes are generating new augmented input images, which are used to enhance the DRL performances of faulty input samples.

3. Theoretical Background

First of all, a graphical abstract that can clearly describe the general suggested structure of this work is given in Fig. 1. In this section we introduce the theoretical background of the DRL, NC and GA.

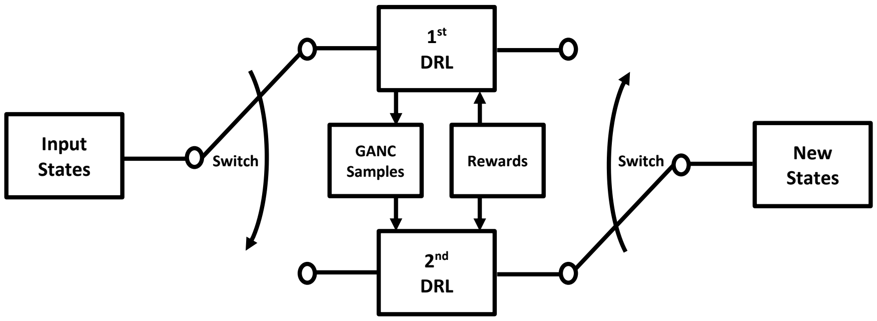

Figure 1: A graphical abstract that can clearly describe the general suggested structure of this work

3.1. Deep Reinforcement Learning

The DRL is a combination between the RL and DL network. Principally, there are two essential types of RL methods - policy search and value function. The first type refers to methods that consider searching for the optimal policy π^*. The second type refers to methods that consider investigating the optimal value-state function $V^*(s)$ [2]. In this work, the employed DRL networks are based on the policy search. According to the MDP concept, the DRL collects input images as current states S_t and by taking the rewards R, it generates actions A which predict new states S_{t+1}. Consequently, the new states can be transitioned again in the inputs as current states.

The essential equation of the policy search is demonstrated as:

$$\pi^* = \underset{\pi}{argmax}\ \mathbb{E}[R \mid \pi] \tag{1}$$

where π is the policy and $\mathbb{E}$ is the expected return value [26].

3.2. Neuron Coverage

ANNs compose of small units called neurons or nodes [27]. Each neuron can be considered as an artificial node. It analyses input values and produces an output decision [28].

In a neuron network, given a threshold *thr* and a testing set T, if the output of a neuron for any determined input is greater than *thr*, the neuron is considered as activated, and vice versa. Formally, a neuron n is activated, if $out(n, \mathbf{x}) > thr$ for any testing input $\mathbf{x} \in T$, where $out(n, \mathbf{x})$ is a function that returns the outcome value of a neuron n given the input $\mathbf{x}$.

Intuitively, the *NC* is defined as the ratio between the number of activated neurons and the number of all employed neurons [18]. Specifically,

$$NCov(T) = \frac{|\{n \in N \mid \forall \mathbf{x} \in T.\ out(n, \mathbf{x}) > thr\}|}{|N|} \tag{2}$$

where $N = \{n_1, n_2, ...\}$ is a set of neurons in the DL network and $NCov(T)$ represents the NC for the given set of test inputs $T = \{\mathbf{x}_1, \mathbf{x}_2, ...\}$ [9]. Conventionally, $|\cdot|$ stands for the cardinality of a set, i.e., the number of members in the set and the boldface $\mathbf{x}$ means it is a vector. This function will be used later as the fitness function for the GA.

We will use the NC to measure robustness. By maximising the NC, both robustness and performance can be enhanced. This type of work has not been studied before for DRL networks and we will consider the DL network as a closest network type.

A DL network would be more robust if the overall network is more activated. The performance or the overall accuracy of a DL network can be enhanced by increasing the NC [18]. Similar method of employing a fixed threshold to explore the effective DL network neurons is used in [16]. We will explore how to maximise the NC using GAs in the following section.

3.3. NC as a fitness function for the GA

GA is an intelligent optimisation technique that simulates human gene behaviours and seeks the best solution for a certain fitness function. GA first selects two sets of numbers (mimicking personal chromosomes) called *generations* and considers them as parents. It then performs operations of *crossover* and/or *mutation*, with the generated coded numbers as *children*. The children that are closest to the fitness function are selected as the new parents. The algorithm repeats until it converges.

One of our contributions is to adapt the definition of NC as Eq. (3) so that it also takes small embedded values $p_{i,j,k}$ into consideration. These values are added to the original input vector and make slight change to the original image view. The new input has higher NC value than the original input as the GA will maximise the NC of the new input.

$$NewCov(T,P) = \frac{|\{n \in N \mid \forall \mathbf{x} \in T, \mathbf{p} \in P.\ out(n, \mathbf{x}+\mathbf{p}) > thr\}|}{|N|} \tag{3}$$

where $NewCov$ represents the new coverage value, T, N, *thr* are as before and P is the set of embedded inputs. $\mathbf{x} \in T$ is a test input vector (sample) and $\mathbf{p} \in P$ is an embedded vector.

The GA is to maximise the fitness function of Eq. (3) and get the embedded $p_{i,j,k}$ values to be added to the DRL inputs.

3.4. GA to generate more training samples

We have a set of training samples $\mathbf{X}_{train} = \{x^1_{train}, x^2_{train}, ..., x^n_{train}\}$ and a set of testing data samples $\mathbf{X}_{test} = \{x^1_{test}, x^2_{test}, ..., x^m_{test}\}$ to begin with. It is worth mentioning that the desired output values can easily be collected from the original training and testing information.

After training the DRL, the testing data are evaluated. We call it a *faulted testing sample* if the predicted output is different from the desired output. Faulted testing samples are crucial as they pick out the weakness of the existing training samples and model. To enhance a DRL's performance, we have to focus on the faulted test samples, as the testing faults are mainly caused by the lack of representation in the training set.

In this paper, we will use the GA to generate augmented training samples that can correct faulty testing outcomes. Thus, a relationship between original training samples and faulted testing cases is required in order to produce more effective training vectors. Some training samples could not enforce a DL network to obtain desired testing outputs. In other words, there is a potential relationship between the failed training samples and their desired testing outputs. We first need to detect this relationship and determine the failed training inputs, then, we need to establish other training vectors that can effectively enforce the DL network during the testing phase.

We call training samples *effective* if without these samples the model would make wrong predicted outputs in the testing phase. Those new effective training samples add diversity to the model and decrease the generalisation error.

In [29], Zhang et al. defined the "similarity" between training data set and and a testing sample by the following distance metric.

$$D(x^j_{test}, \mathbf{X}_{train}) := \frac{1}{k} \sum_{i=1}^{k} \left\| h(x^j_{test}) - h(x^{\pi_j(i)}_{train}) \right\| \qquad (4)$$

where k represents the k-nearest neighbour, x^j_{test} is the j-th testing data, $||.||_p$ is the p-norm, $h(x)$ is the employed DL network and $\pi_j : [n] \to [n]$ is a permutation that $\{\pi_j(1), \pi_j(2), ..., \pi_j(n)\}$ is an ascending ordering of training samples based on the Minkowski norms distance between a training data sample and the j-th testing data sample in the ℓ_p space [29].

This inspires the idea of investigating the relationships between faulted testing inputs and their nearest training inputs which have same targets or desired outputs. We further narrow down the relationship between faulted testing inputs and their nearest training inputs if they have the same outputs. As a result, we modified Eq. (4), by adding a condition that the nearest training and testing samples have the same targeted values as shown below:

$$D(x^j_{test}, \mathbf{X}_{train}) := \{\frac{1}{k}\sum_{i=1}^{k}\left\|h(x^j_{test}) - h(x^{\pi_j(i)}_{train})\right\| \mid x^j_{test} and\ x^{\pi_j}_{train} \Rightarrow t_j\} \quad (5)$$

where t_j represents the target of the faulted j-th testing.

It is important to consider such distance to assign nearest training vectors to the testing inputs that caused faulted outcomes. These training vectors are then modified in order to improve the accuracy of overall system.

3.5. How GA works in our setting

In our work, the GA is adapted to find new training inputs by searching for embedded input values that can maximise the NC of the network. Thus, we choose gene (or chromosomes) size to be the number of input training samples. As each chromosome is a combination between a known training sample and additional embedded values, the embedded values are initialized by random numbers of zeros and ones. Then, the GA algorithm iterates until it converges to the optimum, that is, augmented input sample is produced. All new input chromosomes are assessed by the NC fitness function. The results are ranked from highest to lowest NC fitness function values. We select the two input chromosomes that have the highest NC values to be parents, as those two chromosomes are closer to the optimum solution than others. Consequently, crossover and mutation processes can be implemented in each iteration. Following the behaviour of human genes, the probability of implementing the crossover process is high, whilst the probability of performing the mutation process is low. Scatter crossover method is applied between the two chromosome parents, and random binary numbers are generated to determine the genes exchanging locations. An example of the employed scatter crossover is given in Fig. 2.

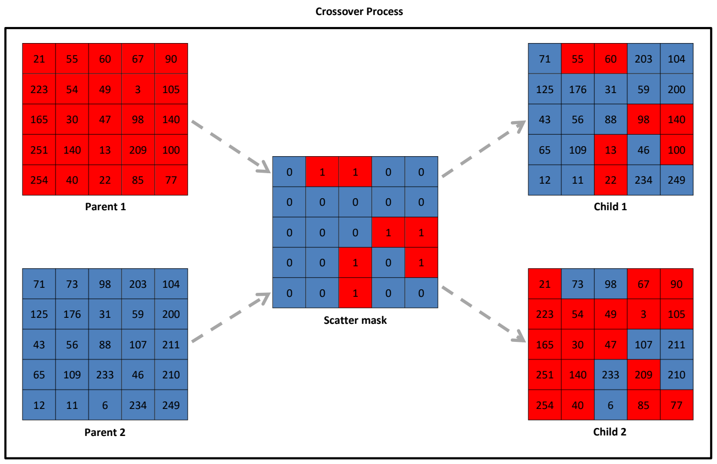

Figure 2: An example of the employed scatter crossover method, where two parent chromosomes exchanging their values according to a scatter mask and producing two offspring children

Fundamentally, the two parent chromosomes are exchanging their values according to a scatter mask of random binary numbers. That is, first chromosome child is produced from positioning Parent 1's values in the locations of logic '1' and Parent 2's values in the locations of logic '0'. Similarly, the second chromosome child is produced from positioning Parent 2's values in the locations of logic '1' and Parent 1's values in the locations of logic '0'. This process is repeated with different scatter masks to establish another offspring population of children.

Uniform mutation process is also performed by establishing a mask of random binary numbers. Because the first parent chromosome is closer to the goal than the second chromosome, the mutation process is applied to the second parent chromosome to enhance its outcome. An example of the employed uniform mutation is given in Fig. 3.

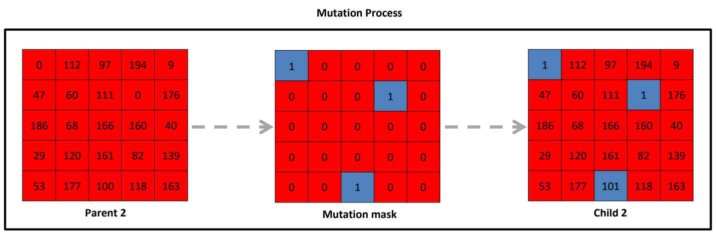

Figure 3: An example of the employed uniform mutation method, where the second parent chromosome adjusting some values according to a mask of random numbers and producing a new chromosome child

Principally, Parent 2 adjusts some of its values according to the random binary mutation mask. The logical '1' assigned locations in a mutation mask refers the values that have corresponding positions in Parent 2. These values are slightly adjusted, increased by 1, as these adjustment in pixel values may increase the number of activated neurons and then optimise the NC function.

The generated chromosome children are evaluated for the fitness function again. Repeatedly, the full process of selecting new parents, implementing crossover between determined chromosomes and performing mutations for the second parents is iterated to obtain the maximum NC function value and collect new input sample(s).

3.6. The GANC Algorithm

In this work, the GANC algorithm is adopted. The visualization of the algorithm is shown in Fig. 4. A pseudo code of overall GANC process can be expressed as given in **Algorithm 1**.

The first round starts with the train samples from which the first DRL (1^{st} DRL or DRL_1) is derived. It then evaluating the testing samples and compares the output results with the desired targets. Subsequently, the fault

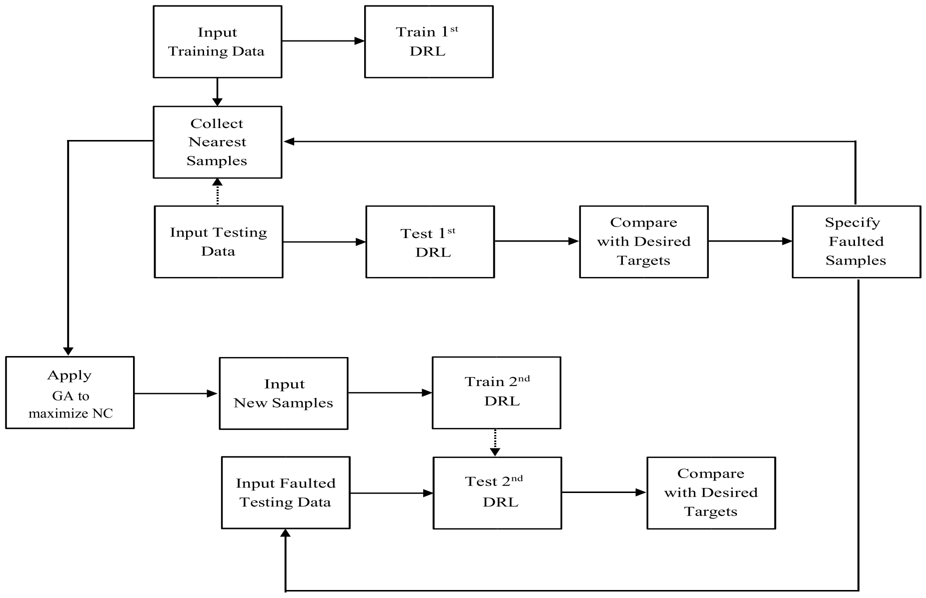

Figure 4: The visualization of the GANC algorithm

Algorithm 1 : The GANC Process

```
Round 1:
1. Train $DRL_1$ from the training samples $\mathbf{X}_{train}$
2. Evaluate $DRL_1$ by the testing samples $\mathbf{X}_{test}$
3. Identify the faulted samples $\mathbf{X}_{fault}$ according to [29]:
       $D(x^j_{test}, \mathbf{X}_{train}) := \{\frac{1}{k}\sum_{i=1}^{k}\left\|h(x^j_{test}) - h(x^{\pi_j(i)}_{train})\right\| \mid x^j_{test}\ and\ x^{\pi_j}_{train} \Rightarrow t_j\}$
4. Apply the GA to produce new inputs $\mathbf{X}_{new}$ for the faulted samples by optimising the NC function:
       $NewCov(T, P) := \frac{|\{n \in N \mid \forall \mathbf{x} \in T, \mathbf{p} \in P.\ out(n, \mathbf{x} + \mathbf{p}) > thr\}|}{|N|}$
Round 2:
5. Train $DRL_2$ for the newly generated inputs $\mathbf{X}_{new}$
6. Re-evaluate $\mathbf{X}_{fault}$ with $DRL_2$
```

tested inputs are identified and their nearest *training* samples with the same targets are found. The algorithm then applies the GA approach to generate new input (augmented) samples.

The second round starts by training a second DRL by the newly generated (augmented) inputs for only the faulted cases. It re-evaluates the faulted tested samples.

The purpose of the second DRL (2^{nd} DRL or DRL_2) is to focus on the faulted cases only. It will enhance the performance of the 1^{st} DRL. This has been confirmed by all the experiments in Section IV. In the testing phase the 1^{st} DRL produces the output according to the provided input. It collects the reward, which could be positive or negative (for a correct or an incorrect outcome, respectively). When a negative reward is received, the 2^{nd} DRL will be trained and carries out the testing process again and tries to produce the correct output.

Advantages: The GANC approach will construct new training input patterns that can provide maximum NC values in the network. Generally, it has the following advantages:

1. Maximising the NC of the DRL, which can enhance the performance [18].
2. Generating many different inputs ($x_{i,j,k}+p_{i,j,k}$) after each time of running the GA. It can be guaranteed that the new inputs have higher NC values than the original inputs. In other words, the new inputs have higher numbers of activated neurons than the original inputs. Therefore, the new inputs can be more effective than the original inputs.
3. Training vectors can be used instead of testing vectors, the latter might be not available to provide comprehensive measuring. This overcomes previous studies, where the robustness of a DL network was always measured based on the testing inputs only.
4. Other than the DRL, this approach can also be applied to any multi-layer neural network such as Multi-Layer Perceptron (MLP) [30], Convolution Neural Network (CNN) [3, 4] and Auto-Encoder Network (AEN) [5].

Principally, the GA can effectively be applied for the proposed GANC approach. There are other newest optimisation methods such as the Whale Optimization Algorithm (WOA) [31], Polar Bear Optimization Algorithm (PBOA) [32] and Red Fox Optimization Algorithm (RFOA) [33]. These methods can work well with the small data sizes and they are still be under the development [31, 32, 33]. The WOA was utilized for producing augmentation inputs as in the GA. Unfortunately, the WOA failed on generating the new inputs because of the required big data size. On the other hand, the GA is more flexible and applicable even with a large-scale size of data.

3.7. Exploited DRL-RT Model:

In this paper, we used the DRL-RT which was proposed in [6] by the same authors. This approach can produce road tracking actions by analysing the road state and getting advantages from the tracking rewards. Fig. 5 shows the DRL-RT approach.

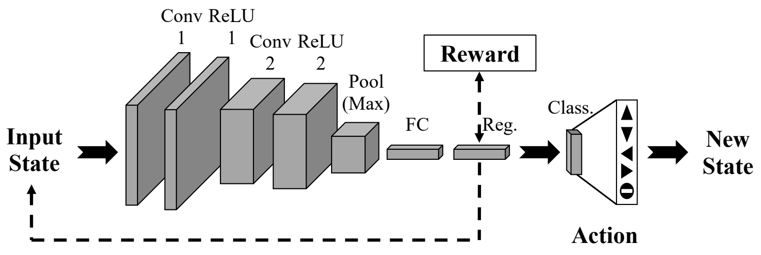

Figure 5: The DRL-RT network as approached in [6]. It composes of convolution layers, two ReLU layers, a maxpooling layer, a fully connected layer, a regression layer and an action or classification layer

The theoretical concepts of the main analysis layers (convolution, ReLU, pooling and fully connected) in the DRL-RT network were stated in [34].

In the first and third layers, the collected information will be converted to feature maps. The feature map is defined as a convoluted 2D image with a kernel of weights. The following general equation represents the operations in a convolution layer:

$$z_{u,v,c^l} = B_{c^l} + \sum_{i=-k_h^l}^{k_h^l} \sum_{j=-k_w^l}^{k_w^l} \sum_{c^{l-1}=1}^{C^{l-1}} W^{c^l}_{i+k_h^l, j+k_w^l, c^{l-1}} z_{u+i, v+j, c^{l-1}} \tag{6}$$

where z_{u,v,c^l} is a convolution layer outcome, (u,v) is the assigned pixel, c^l is the channel number of the convolution layer, $W^{c^l}_{i,j,c^{l-1}}$ is the components of the kernel weights, B_{c^l} is the channel bias of the convolution layer, k_h^l and k_w^l are respectively the height and width of the kernel weights of the convolution layer, C is the number of channels and it is here equal to 3 as we are using three channels of coloured images, $l-1$ is the previous layer, and l is the current layer (the convolution layer) [35].

A ReLU activation function is applied in the second and fourth layers. Equation (7) can represent the ReLU function:

$$o_{u,v,c^l} = f(z_{u,v,c^l}) = \max(0, z_{u,v,c^l}) \tag{7}$$

where o_{u,v,c^l} is a ReLU layer outcome and max is the maximum operation [36].

A pooling layer is used in the fifth part of the DRL-RT. In general, the pooling layer can be applied according to the following equation:

$$q_{a^l,b^l,c} = \max_{0 \le a < p_h, 0 \le b < p_w} o_{a^l \times p_h + a,\ b^l \times p_w + b,\ c} \tag{8}$$

where $q_{a^l,b^l,c}$ is a pooling layer outcome, $0 \le a^l < p_h^l$, p_h^l is the height of the resulting feature maps, $0 \le b^l < p_w^l$, p_w^l is the width of the resulting feature maps, $0 \le c < C^l = C^{l-1}$, p_h and p_w are respectively the width and height of the feature map sub-areas that require pooling [37].

Subsequently, the fully connected layer is used to match between the designed number of subjects and the data of the pooling layer. Equation (10) demonstrates the fully connected layer processes:

$$g_r = \sum_{a=1}^{m_1^{l-1}} \sum_{b=1}^{m_2^{l-1}} \sum_{c=1}^{m_3^{l-1}} W^l_{a,b,c,r} (\mathbf{Q}_c)_{a,b}\ , \qquad \forall 1 \le r \le m^l \tag{9}$$

where g_r is a fully connected layer outcome, m_1^{l-1} and m_2^{l-1} are the width and height of a feature map in the previous layer (the pooling layer) respectively,

m_3^{l-1} is the number of produced feature maps in the pooling layer, $W^l_{a,b,c,r}$ is the connection weights between the fully connected layer and the pooling layer, $\mathbf{Q}_c$ are the pooling layer outputs, and m^l is the number of designed subjects [38].

The computations of the regression layer in the suggested DRL-RT network are based on the Mean Squared Error (MSE). The main MSE equation is illustrated as:

$$MSE = \frac{1}{n}\sum_{r=1}^{n}(t_r - g_r)^2 \tag{10}$$

where n is the number of computed values and t is the desired output values [39]. If the regression output values close to the desired code values, positive rewards are produced. Otherwise, negative rewards are generated.

Finally, the classification layer translates the regression information into actions by converting the obtained values into their assigned classes.

4. Results

4.1. General Parameters:

Four databases from SYNTHIA [40] are used. The selected databases are constructed under different environment conditions: (1) spring, (2) fog, (3) rain and (4) heavy-rain. Moreover, their segmented images, which are provided by the same database, are found to be useful for manually determining the appropriate code of each track.

The input of the DRL-RT network is an image of a car facing view, and it is considered as a current state. The input image size has been prepared as $254 \times 427 \times 3$ pixels. Then, the next layers are arranged as follows: a convolution layer of 5 filters with a filter size of 10×10 pixels, a ReLU layer, a convolution layer of 5 filters with a filter size of 5×5 pixels, a ReLU layer, a maximum pooling layer of a filter size equal to 3×3 pixels with a stride of 3 pixels, a fully connected layer, a regression layer and a classification layer [6]. In the regression layer, a series of directional road tracking codes are generated. Successful codes in this layer produce positive rewards, whereas, unsuccessful codes generate negative rewards. That is, the correct tracking is considered as (+1) and the incorrect tracking is considered as (1). Correct and incorrect tracks are specified by comparing the regression layer outputs with the desired tracking codes as illustrated in [6]. The network is propagated, forwarded and backwarded the information for updating the network weights, during the training phase till obtaining as many positive rewards as possible. Given the codes in the regression layer, it is the classification layer's task to generate a new action. The employed DRL-RT network is based on the policy search as highlighted in [6].

4.2. Practical Experiments:

Following [6], the number of employed frames in each environments are: 264, 284, 268 and 248 images for the environments of spring, fog, rain and heavy-rain, respectively. Two scenarios were considered in [6]:

(1) Exploiting the four environments altogether. The frames are randomly partitioned to the 2/3 of all the frames for training, and the remaining 1/3 frames were used for testing.

(2) Separately training and testing each environment, where the odd-indexed images are utilised for the training phase and the even-indexed images are utilised for the testing phase.

The number of training steps can be computed from the number of targets, which refer to the positive rewards. To clarify, the number of training steps in the 1^{st} DRL and 2^{nd} DRL are equal to the number of their targets, where these targets are already lead to the positive rewards of the new RL states. Achieving all targets in the training phase mean collecting the highest number of rewards. The number of training steps in the 1^{st} DRL that are required for collecting positive rewards can be considered for the experiment of scenario 1 as 709 steps and for the experiments of scenario 2 as 132, 142, 134 and 124 steps for the environments of spring, fog, rain and heavy-rain, respectively. The number of training steps in the 2^{nd} DRL that are required for collecting positive rewards can be considered for the experiment of scenario 1 as 1120 steps and for the experiments of scenario 2 as 150, 210, 340 and 270 steps for the environments of spring, fog, rain and heavy-rain, respectively. In each experiment, trainings were mainly implemented two times, for the 1^{st} DRL and 2^{nd} DRL, and they were repeated offline many times until correct actions were provided for road tracking.

In training phases, the following training parameters are utilized: Adaptive Moment Estimation (ADAM) optimizer [41], mini batch size equal to 128, squared gradient decay factor (β_2) equal to 0.99, gradient decay factor (β_1) equal to 0.9 and learning rate equal to 0.0003.

As mentioned, in the testing phase the original DRL-RT generates its output according to the afforded input. It receives the testing rewards. The positive reward indicates the correct tracking, whilst, the negative reward signifies the incorrect tracking. The supported DRL carries out the testing process when negative rewards are collected.

4.3. Prior Results:

The prior results of testing the original DRL-RT are quite interesting, but reasonable. The driving accuracy attained its highest value of 93.94% by using the spring environment database, which is expected, as the spring images are the clearest. The fog environment database obtained a high driving accuracy of 93.66%. Although the overall views are blurred, the road tracking can still be recognised and decided based on the views, with slightly lower accuracy than the spring views. As for the rain environment, the accuracy was 89.55% and this was due to the noise effects of rain drops on image views. Finally, the inferior driving accuracy of 84.68% was recorded for the heavy-rain environment database as the amount of rain drops (or noise) significantly increased here.

4.4. GANC Parameters and NC Results:

Our suggested GANC method is applied to the DRL-RT with the following specifications:

1. Population size equal to 10.

2. Gene size is similar to the network input size ($254 \times 427 \times 3$ pixels).
3. Rank selection.
4. Scatter crossover with the probability $P_c < 0.8$.
5. Uniform mutation with the probability $P_m < 0.3$.
6. Number of iterations equal to 20.

GANC parameters are reasonably chosen, in addition of considering the limitations of the employed computer which has the following facilities: laptop of type Hewlett-Packard (HP), processor of type Intel Core i7-2620M, processor speed of 2.70 GHz and Random Access Memory (RAM) of 8GB RAM. To illustrate, GANC parameters are selected as follows: increasing the population size to more than 10 required more memory to be implemented, the gene size fixed to the similar network input size of $254 \times 427 \times 3$ pixels, the rank selection was found to be operative as it considers nearest samples in addressing the fitness function, the scatter crossover with a high probability could provide the effective implementations (it was even the default type in the employed software program), the uniform mutation with a low probability could also offer the effective execution and appropriate number of iterations was found to be 20 as increasing this number would resulted in further increasing the execution time.

Uncertain environments with a large-scale number of images (total of 2,184 images) have been considered in Scenario 1 experiments.

Fig. 6 shows the progress of NCs along the number of iterations for the different databases and experiments in the GANC. In all cases the NC increases during the GANC implementations, because GANC aims to maximise the NC, which is the fitness function (Eq. (3)). The results confirm the GANC's capability of increasing the NCs.

This figure also shows that all the NCs started in relatively small values (0.44 - 0.56). As the DRL network has a huge number of neurons, the NC always stayed as small values even if the number of activated neurons is reasonably big.

Fig. 7 shows an example of a new input image that generated by the GANC

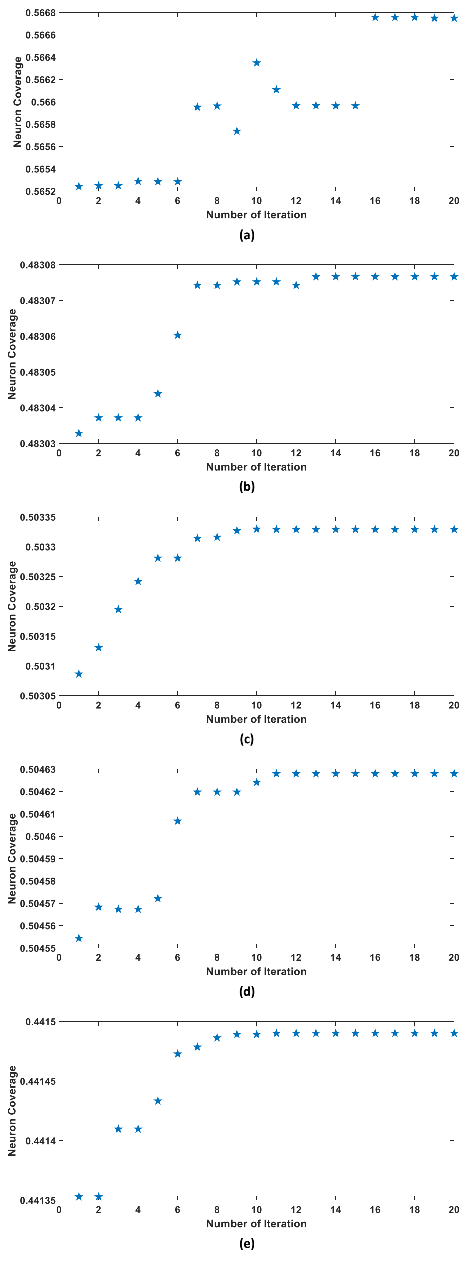

Figure 6: Examples of the NC progress in the GANC for the: (a) spring, (b) fog, (c) rain and (d) heavy-rain environment and (e) all environments

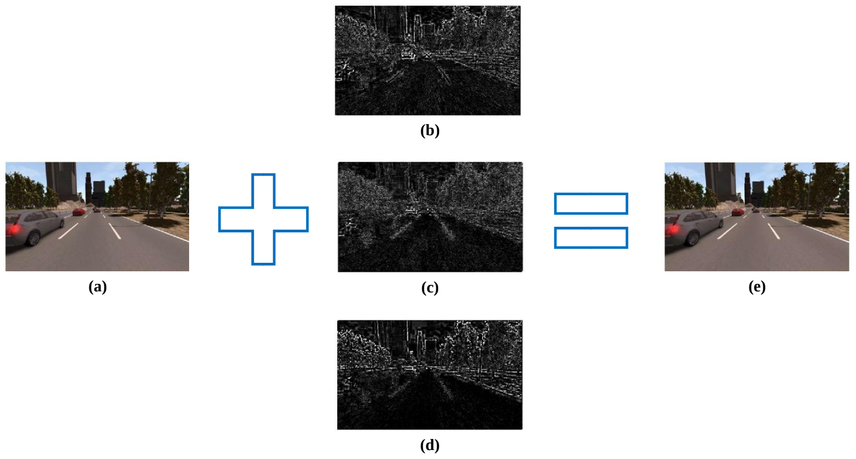

Figure 7: An example of a new input image generated by the GANC method, where the leftmost one is an original training input image, the middle three are a produced embedded input image for red, green and blue channel (N.B., images have been adjusted to be clearer), respectively, and the rightmost one is a resulted new input sample.

method. In this figure, the established embedded values for each coloured channel are added to a training frame. Consequently, a new input image is generated. The new produced images are exploited later to address faulted testing outcomes.

Moreover, multiple measurements have been applied between the generated augmented images and their original images. These measurements are the Structural SIMilarity (SSIM) Index [42], Mean Absolute Percentage Error (MAPE) [**?**] and Difference Entropy (DE) [43]. The obtained average results are given in Table 2.

All results in this table yield that the generated augmented images have the same characteristics of their original images as the recorded average results of SSIM, MAPE and DE measurements show small differences between the augmented and original images. The augmented images have slightly changing pixel values compared to the original images and these values have been produced by increasing the NC values of the employed network. Increasing the NC values

Table 2: The average results of multiple measurements between the generated augmented images and their original images

Measurement	Results for Scenario 1	Results for Scenario 2			
		SPRING	**FOG**	**RAIN**	**HEAVY-RAIN**
SSIM	0.29	0.15	0.45	0.47	0.27
MAPE	0.15	0.26	0.18	0.1	0.13
DE	3.85	3.74	4.37	3.38	3.81

principally means increasing the activated neurons in the network and this can lead to enhance its performance.

Fig. 8 shows the relationships between the number of activated and deactivated neurons for different threshold values. The effects of the threshold values are evaluated for the first convolution layer, first ReLU layer, second convolution layer, second ReLU layer and maxpooling layer of the DRL-RT. It can be argued that using a threshold for determining the activated neurons cannot be justified as a good basis as explained in [17].

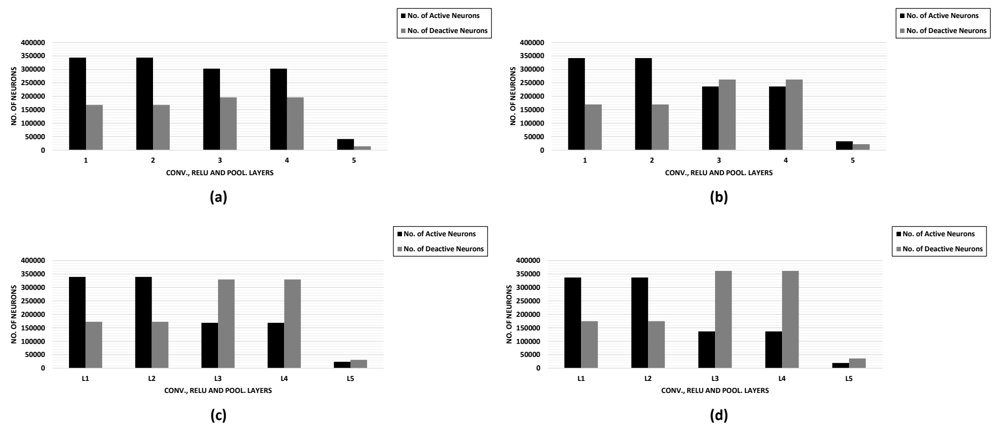

Figure 8: The relationships between the number of activated and deactivated neurons for certain threshold values *thr* that are employed in [17]: (a) $thr = 0$, (b) $thr = 0.2$, (c) $thr = 0.5$ and (d) $thr = 0.75$

To overcome the confusion of selecting the best threshold and to provide a good basis for determining the activated neurons, the threshold value has been assigned to zero in this work. This makes more sense because the deactivated neurons generate zero or negative values and the activated neurons produce other values. Any positive value more than zero may affect the output decision of the DL network, while he zeros and negative values are usually be excluded by the ReLU and Maxpooling layers.

Fig. 8 shows that for the first convolution, ReLU and maxpooling layer (C/R/M, for short) of the DRL-RT, the effects of threshold values are very small. Whereas for the second C/R/M layer, the effects of threshold values are very obvious. This means the later layers are more sensitive to the number of activated and deactivated neurons than the earlier layers. It is because that the earlier layers are mainly doing early processing of filtering and rectifying the internal informations. Using a zero threshold value produces reasonable relationships between the activated and deactivated neurons. On the other hand, using other threshold values generates bigger numbers of deactivated neurons than activated neurons and this is not feasible for any DL network.

4.5. Enhanced Performances:

After applying the proposed GANC to the employed databases, the driving accuracies have been enhanced in all the applied experiments as given in Fig. 9. That is, the driving accuracy is increased from 93.94% to 97.73% for the spring environment, from 93.66% to 95.07% for the fog environment, from 89.55% to 93.28% for the rain environment, from 84.68% to 88.71% for the heavy-rain environment and from 95.49% to 97.75% for all the environments. The attained results are clearly indicating that the GANC approach can successfully enhance the DRL outcomes.

4.6. Comparisons:

For comparison purposes, recent augmentation suggestions with their employed networks have been explored and simulated. The databases of the two scenarios, which has been explained in the Practical Experiments subsection,

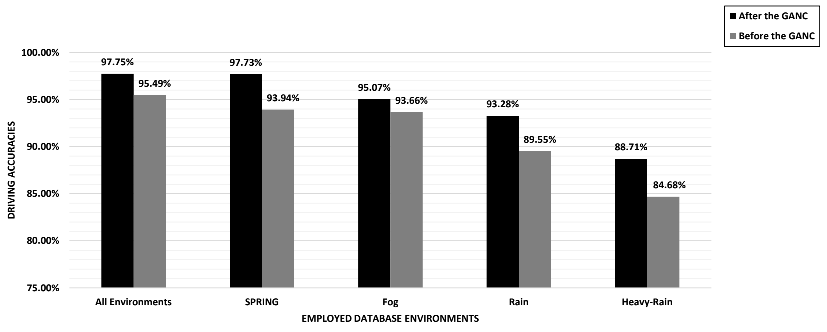

Figure 9: The results of driving accuracies before and after using the GANC approach

have been applied to the road tracking application by using similar characteristics to the compared augmentation studies. Such interesting results have been obtained as shown in Table 3.

Table 3: Comparisons with different augmentation methods

Reference	**Augmentation Method**	**Accuracy for Scenario 1**	**Accuracy for Scenario 2**			
			SPRING	**FOG**	**RAIN**	**HEAVY-RAIN**
Wang and Perez (2017) [44]	Augmentation network	14.08%	11.36%	13.38%	11.94%	17.74%
	Traditional methods	80.28%	88.64%	89.44%	91.79%	62.90%
Marcus *et al.* (2017) [45]	Augmentor	84.79%	93.94%	93.66%	90.30%	87.10%
Taylor and Nitschke (2017) [46]	Cropping	66.20%	75.76%	62.68%	67.91%	63.71%
Lu (2019) [47]	Random transformations	21.69%	91.67%	90.14%	83.58%	87.90%
Andriyanov and Andriyanov (2020) [48]	Various transformations	81.69%	84.09%	83.80%	81.34%	78.23%
Our Approach	GANC	**97.75%**	**97.73%**	**95.07%**	**93.28%**	**88.71%**

From this table, it can be observed that inferior performances have been reported for [44] with its suggested augmentation and classification networks. This is expected as the augmentation network produces augmentation images by randomly selecting two different samples of inputs from each class. Since, road tracking images are dependent on training certain views, randomly choosing two different views as inputs reduces the accuracy of road tracking decision. On the other hand, using traditional augmentation methods of (shifting, zooming in/out, rotating, flipping, distorting and shading with a hue) and classification network that are employed in [44] generates reasonable driving accuracy results. Similarly, the proposed augmentor method with the CNN in [45] could produce reasonable deriving accuracy performances. Because this method probabilistically employs elastic transforms and rotation process to generate the augmentation images. The cropping augmentation method with the CNN which was exploited in [46] attained moderate performances. This method works based on cropping parts of images and this could accordingly affect the road tracking processes. The random transformations of (rotation, translation and scaling) with the DL that are suggested in [47] have reported a bad result for scenario 1 and good results for scenario 2. This is due to the number of employed samples in scenario 1 and scenario 2. That is, the influence of random transformations is negatively increased when the number of samples is increased. Various transformations (such as changing colour and adding noises) with the network that were utilized in [48] recorded analysable driving performances. That is, the clear environment of spring attains the best percentage and this result reasonably decreases according to the applied noisy environments of fog, rain and heavy-rain. Also, all environments in Scenario 1 could reach comparable results to spring, fog and rain in Scenario 2. Obviously, best results have been benchmarked for our proposed GANC approach, where highest driving accuracies have been recorded for all employed databases.

The times of compared augmentation methods are also be considered. As mentioned, the employed computer has the characteristics of (HP laptop, Intel Core i7-2620M processor, 2.70 GHz processor speed and RAM of 8GB). Furthermore, spring environment has been selected as a basis for constructing the time comparisons. Classical augmentations of traditional methods [44], augmentor [45], cropping [46], random transformations [47] and various transformations [48] attained around 2.22, 14.80, 5.06, 4.58 and 4.57 seconds, respectively. These small time values are expected as they use classical augmentation methods. On the other hand, the complex augmentations of the augmentation network [44] and proposed GANC obtained around 143.85 and 2440.75, respectively. These high time values are also expected as they use complex augmentation methods by exploiting DL techniques. This can be considered as the main drawback of our GANC approach as it requires observing and collecting the number of active neurons inside a DRL network for the NC calculations.

Testing demo examples from the four databases are given in Fig. 10.

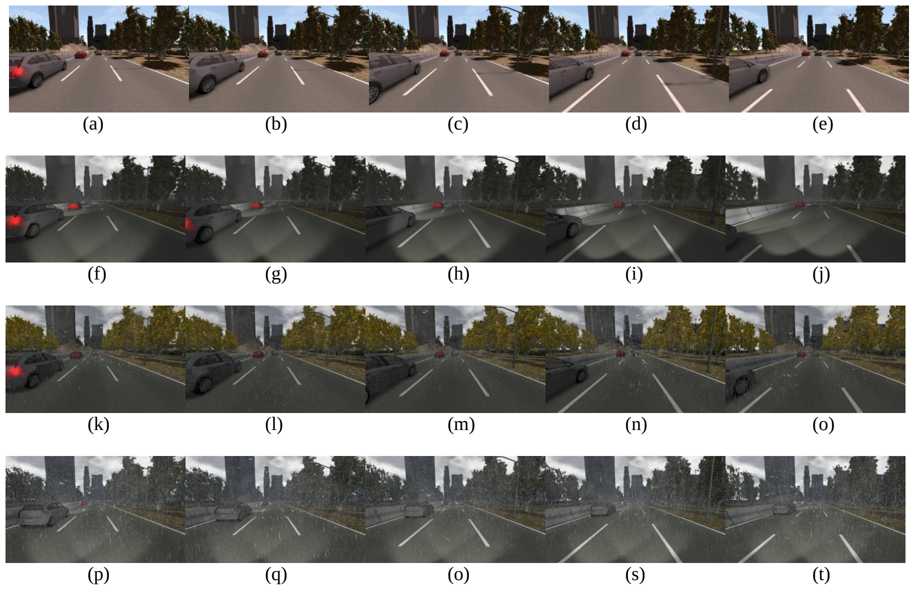

Figure 10: Testing demo examples from the four employed environments where: the first row represents the spring environments, the second, third and fourth row represents the fog, rain, and heavy-rain environments

5. Conclusion

In this paper, a novel approach named the GANC was presented based on the GA, NC and DRL. GANC optimises the NC by establishing embedded information within the inputs. The new inputs will as a result increase the NC of a DRL network. Then these inputs were used to enhance the performance of faulty testing samples. Road tracking for self-driving car applications with different experiments and environments was employed to evaluate the suggested approach. The driving accuracies were benchmarked to 93.94%, 93.66%, 84.68% and 95.49% by using the DRL-RT for the spring, fog, rain, heavy-rain and all the environments, respectively. The performances have been enhanced to 97.73%, 95.07%, 93.28%, 88.71% and 97.75% after applying the GANC for the spring, fog, rain, heavy-rain and all the environments, respectively. The proposed approach has successfully improved the robustness and performance of the employed DRL for various environments and experiments.

Acknowledgment

This work is supported by EPSRC grants (EP/P015387/1, EP/P00430X/1), Birkbeck BEI School Project (ARTEFACT), Guangdong Science and Technology Department grant (No. 2018B010107004), China.

[1] D. Knemeijer, Stripai: Determining the suitability of implementing deep reinforcement learning principles into new domains, MSc thesis, Faculty of Science, Utrecht University (2019).

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Processing Magazine 34 (6) (2017) 26–38.

[3] M. M. M. AL-Hatab, R. R. O. Al-Nima, I. Marcantoni, C. Porcaro, L. Burattini, Classifying various brain activities by exploiting deep learning techniques and genetic algorithm fusion method, TEST Engineering & Management 83 (2020) 3035–3052.

[4] M. M. AL-Hatab, R. R. O. Al-Nima, I. Marcantoni, C. Porcaro, L. Burattini, Comparison study between three axis views of vision, motor and pre-frontal brain activities, Journal of Critical Reviews 7 (5) (2020) 2598–2608.

[5] A. S. Anaz, R. R. O. Al-Nima, M. Y. Al-Ridha, Multi-encryptions system based on autoencoder deep learning network, Solid State Technology 63 (6) (2020) 3632–3645.

[6] R. R. O. Al-Nima, T. Han, T. Chen, Road tracking using deep reinforcement learning for self-driving car applications, in: International Conference on Computer Recognition Systems, Springer, 2019, pp. 106–116.

[7] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, D. Kroening, Concolic testing for deep neural networks, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ACM, 2018, pp. 109–119.

[8] C. Wang, Research and application of traffic sign detection and recognition based on deep learning, in: 2018 International Conference on Robots & Intelligent System (ICRIS), IEEE, 2018, pp. 150–152.

[9] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: Automated whitebox testing of deep learning systems, in: proceedings of the 26th Symposium on Operating Systems Principles, ACM, 2017, pp. 1–18.

[10] L. Davis, Handbook of genetic algorithms.

[11] M. Mitchell, An introduction to genetic algorithms, MIT press, 1998.

[12] M. Kumar, M. Husian, N. Upreti, D. Gupta, Genetic algorithm: Review and application, International Journal of Information Technology and Knowledge Management 2 (2) (2010) 451–454.

[13] Y. Sun, X. Huang, D. Kroening, Testing deep neural networks, arXiv preprint arXiv:1803.04792.

[14] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, et al., Deepmutation: Mutation testing of deep learning systems, in: 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2018, pp. 100–111.

[15] X. Bai, W. Niu, J. Liu, X. Gao, Y. Xiang, J. Liu, Adversarial examples construction towards white-box q table variation in dqn pathfinding training, in: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), IEEE, 2018, pp. 781–787.

[16] K. Pei, Y. Cao, J. Yang, S. Jana, Towards practical verification of machine learning: The case of computer vision systems, arXiv preprint arXiv:1712.01785.

[17] L. Ma, F. Juefei-Xu, J. Sun, C. Chen, T. Su, F. Zhang, M. Xue, B. Li, L. Li, Y. Liu, et al., Deepgauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems, arXiv preprint arXiv:1803.07519 7.

[18] Y. Tian, K. Pei, S. Jana, B. Ray, Deeptest: Automated testing of deep-neural-network-driven autonomous cars, in: Proceedings of the 40th international conference on software engineering, ACM, 2018, pp. 303–314.

[19] J. H. Kelly, S. V. Dan, J. C. John, K. R. Leanna, A practical tutorial on modified condition/decision coverage, Tech. Rep. Nasa.

[20] R. Balan, M. Singh, D. Zou, Lipschitz properties for deep convolutional networks, arXiv preprint arXiv:1701.05217.

[21] S. Mangano, An introduction to genetic algorithm implementation, theory, application, history and future potential.

[22] E. A. K. Mishra, E. Y. Mohapatra, E. A. K. Mishra, Multi-objective genetic algorithm: A comprehensive survey, International Journal of Emerging Technology and Advanced Engineering 3 (2) (2013) 81–90.

[23] T. Vuolio, V.-V. Visuri, A. Sorsa, S. Ollila, T. Fabritius, Application of a genetic algorithm based model selection algorithm for identification of carbide-based hot metal desulfurization, Applied Soft Computing (2020) 106330.

[24] D. Połap, An adaptive genetic algorithm as a supporting mechanism for microscopy image analysis in a cascade of convolution neural networks, Applied Soft Computing (2020) 106824.

[25] S. Leonori, M. Paschero, F. M. F. Mascioli, A. Rizzi, Optimization strategies for microgrid energy management systems by genetic algorithms, Applied Soft Computing 86 (2020) 105903.

[26] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, A brief survey of deep reinforcement learning, arXiv preprint arXiv:1708.05866.

[27] L. V. Fausett, P. Hall, Fundamentals of neural networks: architectures, algorithms, and applications, Prentice-Hall Englewood Cliffs, 1994.

[28] R. R. O. Al-Nima, Signal processing and machine learning techniques for human verification based on finger textures, PhD thesis, School of Engineering, Newcastle University (2017).

[29] H. Zhang, H. Chen, Z. Song, D. Boning, I. S. Dhillon, C.-J. Hsieh, The limitations of adversarial training and the blind-spot attack, arXiv preprint arXiv:1901.04684.

[30] R. R. Al-Nima, S. Dlay, W. Woo, A new approach to predicting physical biometrics from behavioural biometrics, World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering 8 (11) (2014) 1996–2001.

[31] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in engineering software 95 (2016) 51–67.

[32] D. Połap, et al., Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism, Symmetry 9 (10) (2017) 203.

[33] D. Połap, M. Woźniak, Red fox optimization algorithm, Expert Systems with Applications 166 114107.

[34] R. R. Omar, T. Han, S. A. M. Al-Sumaidaee, T. Chen, Deep finger texture learning for verifying people, IET Biometrics 8 (2019) 40–48(8).

[35] E. Simo-Serra, S. Iizuka, K. Sasaki, H. Ishikawa, Learning to simplify: fully convolutional networks for rough sketch cleanup, ACM Transactions on Graphics (TOG) 35 (4) (2016) 121.

[36] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[37] J. Wu, Introduction to convolutional neural networks, National Key Lab for Novel Software Technology. Nanjing University. China.

[38] D. Stutz, Neural codes for image retrieval, Proceedings of the Computer VisionECCV (2014) 584–599.

[39] Ş. Sağiroğlu, N. Özkaya, An intelligent face features generation system from fingerprints, Turkish Journal of Electrical Engineering & Computer Sciences 17 (2) (2009) 183–203.

[40] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, A. M. Lopez, The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3234–3243.

[41] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[42] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE transactions on image processing 13 (4) (2004) 600–612.

[43] Y. K. Kumar, Comparison of fusion techniques applied to preclinical images: Fast discrete curvelet transform using wrapping technique & wavelet transform., Journal of Theoretical & Applied Information Technology 5 (6).

[44] J. Wang, L. Perez, The effectiveness of data augmentation in image classification using deep learning, arXiv preprint arXiv:1712.04621.

[45] M. D. Bloice, C. Stocker, A. Holzinger, Augmentor: an image augmentation library for machine learning, arXiv preprint arXiv:1708.04680.

[46] L. Taylor, G. Nitschke, Improving deep learning using generic data augmentation, arXiv preprint arXiv:1708.06020.

[47] Y. Lu, Food image recognition by using convolutional neural networks (cnns), arXiv preprint arXiv:1612.00983v2.

[48] N. Andriyanov, D. Andriyanov, The using of data augmentation in machine learning in image processing tasks in the face of data scarcity, in: Journal of Physics: Conference Series, Vol. 1661, IOP Publishing, 2020, p. 012018.