# Artificial Neural Networks: the missing link between curiosity and accuracy

Giorgia Franchini[1], Paolo Burgio[1], and Luca Zanni[1]

University of Modena and Reggio Emilia

**Abstract.** Artificial Neural Networks, as the name itself suggests, are biologically inspired algorithms designed to simulate the way in which the human brain processes information. Like neurons, which consist of a cell nucleus that receives input from other neurons through a web of input terminals, an Artificial Neural Network includes hundreds of single units, artificial neurons or processing elements, connected with coefficients (weights), and are organized in layers. The power of neural computations comes from connecting neurons in a network: in fact, in an Artificial Neural Network it is possible to manage a different number of information at the same time. What is not fully understood is which is the most efficient way to train an Artificial Neural Network, and in particular what is the best mini-batch size for maximize accuracy while minimizing training time. The idea that will be developed in this study has its roots in the biological world, that inspired the creation of Artificial Neural Network in the first place.
Humans have altered the face of the world through extraordinary adaptive and technological advances: those changes were made possible by our cognitive structure, particularly the ability to reasoning and build causal models of external events. This dynamism is made possible by a high degree of curiosity. In the biological world, and especially in human beings, curiosity arises from the constant search of knowledge and information: behaviours that support the information sampling mechanism range from the very small (initial mini-batch size) to the very elaborate sustained (increasing mini-batch size).
The goal of this project is to train an Artificial Neural Network by increasing dynamically, in an adaptive manner (with validation set), the mini-batch size; our hypothesis is that this training method will be more efficient (in terms of time and costs) compared to the ones implemented so far.

**Keywords:** artificial neural network, stochastic gradient, mini-batch size increasing

## 1 Introduction

Artificial Neural Networks (ANNs) are biologically inspired algorithms designed to simulate the way in which the human brain processes information. ANNs collect their knowledge by detecting the patterns and relationships in data, and

learn (or are trained) through experience, not from programming. An ANN is formed by hundreds of single units, artificial neurons or processing elements (PE), connected with coefficients (weights), which constitute the neural structure and are organised in layers. The power of neural computations comes from connecting neurons in a network. Each PE has weighted inputs, activation function, and one output like in Figure 1. The behaviour of a neural network is

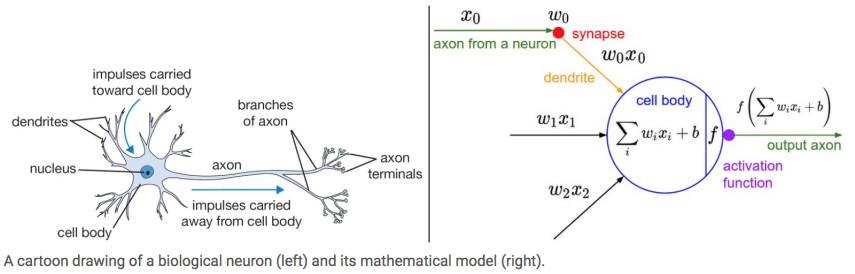


Fig. 1: Neuron and mathematical model

determined not only by the activation functions of its neurons, by the loss function, and by the architecture itself but also by the number of information that it processes simultaneously. The weights are the adjustable parameters and, in that sense, a neural network is a parametrized system. The weighed sum of the inputs constitutes the activation of the neuron. The activation signal is passed through activation function to produce a single output of the neuron. Activation function introduces non-linearity to the network. During training, the inter-unit connections are optimized until the error in predictions is minimized. Once the network is trained and tested, it can be given new input information to predict the output. The various applications of ANNs can be summarised into classification or pattern recognition, prediction, and in modelling.

### 1.1 ANN history and biologically inspiration

The first abstract model of Artificial Neural Networks (ANN) was proposed by McCulloch and Pitts in 1943. They considered the neuron as a binary device, which can be assimilated to a logic unit which compute a logical function of its inputs. The neuron of McCulloch and Pitts may thus be found in only one of two possible states $\{0, 1\}$. It may receive inputs from exciting synapses which all have the same value. If the sum of the inputs exceeds a certain threshold, the neuron is activated, otherwise it is not. McCulloch and Pitts succeeded in demonstrating that a network of neurons of this type could compute any finite logical expression. This result had a profound influence, in what for the field it

showed for the first time that a network of extremely simple elements possessed an enormous computing power, a power which derived exactly from the presence of numerous elements and from their interactions [7]. On the other hand ANNs, as the name suggests, are biologically inspired, in particular biological neurons consist of a cell nucleus, which receives input from other neurons through a web of input terminals, or branches, called dendrites.

## 2 Machine Learning and large scale optimization problems

The promise of artificial intelligence has been a topic of both public and private interest for decades. Advances based on such techniques may be in store in the future, many researchers have started to doubt these classical machine learning approaches, choosing instead to focus their efforts on the design of systems based on statistical techniques, such as in the rapidly evolving and expanding field of machine learning. Machine learning and the intelligent systems that have been born out of it have become an indispensable part of modern society. One of the pillars of machine learning is mathematical optimization, which, in this context, involves the numerical computation of parameters for a system designed to make decisions based on yet unseen data. That is, based on currently available data, these parameters are chosen to be optimal with respect to a given learning problem (and a given loss or cost function) [1]. A loss function or cost function, in this context, is a function that maps values of variables onto a real number representing some "cost" associated with the event.
For example the following optimization problem: which minimizes the sum of cost functions over samples from a finite training set composed by sample data $a_i \in \mathbb{R}^d$ and class label $b_i \in \{\pm 1\}$ for $i \in \{1 \dots n\}$, appears frequently in machine learning:

$$\min F(x) \equiv \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{1}$$

where $d$ is the sample size, $n$ is the number of samples, and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is the cost function corresponding to a training set element.
For example in the logistic regression case we have:

$$f_i(x) = \log \left[1 + exp(-b_i a_i^T x)\right]$$

We are interested in finding $x$ that minimizes (1).
For given $x$, computing $F(x)$ and $\nabla F(x)$ is prohibited, due to the large size of the training set. When $n$ is large, Stochastic Gradient Descent (SGD) method and its variants have been chosen as the main approaches for solving (1).

### 2.1 Stochastic Gradient Methods and mini-batch

We define generalized SGD method as Algorithm 1. The algorithm merely presumes that three computational tools exist: (i) a mechanism for generating a

realization of a random variable $\xi_k$; (ii) given an iterate $x_k \in \mathbb{R}^d$ and the realization of $\xi_k$, a mechanism for computing a stochastic vector $g(x_k, \xi_k) \in \mathbb{R}^d$; and (iii) given an iteration number $k \in \mathbb{N}$, a mechanism for computing a scalar learning rate $\eta_k > 0$.

**Algorithm 1** Stochastic Gradient Descent (SGD) Method

1: Choose an initial iterate $x_1$.
2: **for** $k = 1, 2, \ldots$ **do**
3: Generate a realization of the random variable $\xi_k$.
4: Compute a stochastic vector $g(x_k, \xi_k)$.
5: Choose a learning rate $\eta_k > 0$.
6: Set the new iterate as $x_{k+1} \leftarrow x_k - \eta_k g(x_k, \xi_k)$.
7: **end for**

The generality of Algorithm 1 can be seen in various ways. First, the value of the random variable $\xi_k$ needs only be viewed as a seed for generating a stochastic direction; as such, a realization of it may represent the choice of a single training sample as in the simple SGD method, in particular: in the $k-th$ iteration of SGD, a random index of a training sample $i_k$ is chosen from $\{1, 2, \ldots, n\}$ and the iterate $x_k$ is updated by

$$x_{k+1} = x_k - \eta_k \nabla f_{i_k}(x_k)$$

where $\nabla f_{i_k}(x_k)$ denotes the gradient of the $i_k - th$ component function at $x_k$, therefore in this case $g(x_k, \xi_k) = \nabla f_{i_k}(x_k)$.
In another case, the value of the random variable $\xi_k$ may represent a set of samples as in the **mini-batch** SGD method: one can employ a mini-batch approach in which a small subset of samples, call it $S_k \in \{1, ..., n\}$, is chosen randomly at each iteration, leading to

$$x_{k+1} \leftarrow x_k - \frac{\eta_k}{|S_k|} \sum_{i \in S_k} \nabla f_i(x_k). \tag{2}$$

This is the case in our approach, where $g(x_k, \xi_k) = \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(x_k)$.
In literature we can find several papers in which two techniques are used to increase accuracy: decaying the learning rate and increasing the batch size. In general, it is common practice to decay the learning rate. But with an increasing batch size, we can usually obtain the same learning curve on both training and test sets by increasing the batch size during training instead. In the state of the art, we can find different works that use this technique but with random criteria and without the use of the validation set [6].

## 3 The idea: curiosity to improve accuracy

Consider the iteration

$$x_{k+1} \leftarrow x_k - \bar{\eta} g(x_k, \xi_k) \tag{3}$$

where the stochastic directions are computed for some $\tau > 1$ as

$$g(x_k, \xi_k) := \frac{1}{n_k} \sum_{i \in \mathcal{S}_k} \nabla f_i(x_k; \xi_{k,i}) \text{ with } n_k := |\mathcal{S}_k| = \lceil \tau^{k-1} \rceil. \tag{4}$$

That is, consider a mini-batch SGD iteration with a fixed learning rate in which the mini-batch size used to compute unbiased stochastic gradient estimates increases geometrically, or in an other manner, as a function of the iteration counter $k$.
Returning to the concept of biological inspiration that justifies the idea of ANNs, the new approach suggested here can be inserted in this context, considering the information sampling mechanism used by the animals in general, and humans in particular. During our limited existence, humans have alterated the face of the world; these extraordinary advances are made possible by our cognitive structure, particularly the ability to reason and build causal models of external events. This dynamism is made possible by our high degree of curiosity. Many animals, and especially humans, seem constantly to seek knowledge and information in behaviours ranging from the very small (initial mini-batch size) to the very elaborate sustained (increasing mini-batch size). In neuroscience research, the most commonly considered exploration strategies are based on random action selection or automatic biases toward novel, surprising or uncertain events. Actions extremely driven by randomness, novelty, uncertainty or surprise are valuable for allowing agents to discover new tasks. However, these actions have an important limitation: they do not guarantee that an agent will learn. The mere fact that an event is novel or surprising does not guarantee that it contains regularities that are detectable, generalizable or useful. Therefore, heuristics based on novelty (in our case a big mini-batch size from the beginning) can guide efficient learning in small and closed spaces, where the number of tasks is small, but are very inefficient in large open ended spaces, where they only allow the agent to collect very sparse data and risk trapping him in unlearnable tasks. This motivates the search for additional solutions that use more targeted mechanisms designed to maximize learning per se (in our case a dynamic change of the mini-batch size) [3]. Similarly, we find interesting the concept of novelty search and its application in Evolutionary Neural Network to create increasing complex structures [5].

## 4 Numerical experiment

The *database* MNIST (Modified National Institute of Standards and Technology database) is a large collection of handwritten digits, commonly used for testing different systems that process images. The *database* MNIST contains $60,000$ images for network training and $10,000$ images used to test the network: the *test*

*set*. The $60,000$ initial images are further subdivided into $55,000$ images that represents the real *training set* and $5,000$ images that represents the *validation set*, used, in our case, to have a dynamic criterion to evaluate the performance of the algorithm.
The images are in gray-scale (0-255) centred in a box of $28 \times 28$ pixels.

### 4.1 The problem

The idea of building a classifier for the database MNIST, is to use a Convolution Neural Network CNN by creating a multiple classifier for the ten digits and a binary classifier to distinguish between the eight digit and the others.

### 4.2 Convolutional Neural Network

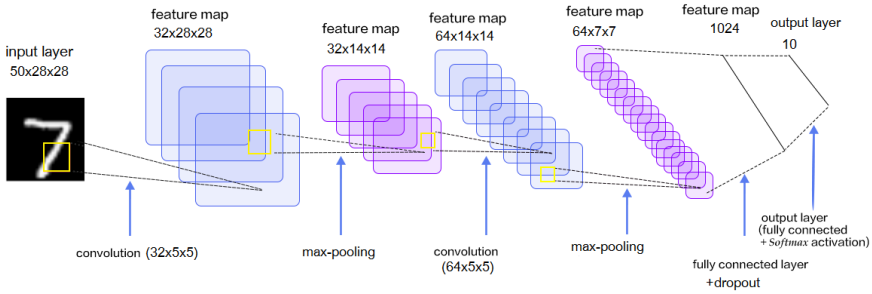


Fig. 2: CNN with 10 outputs

As you can see from Figure 2 the network takes as input an image of $28 \times 28$ pixels and processes it through five layers and then reaches an output layer composed of ten or two neurons: the numbers from 0 to 9 in the multiple classifier case and 8 or not8 in the binary case.
Internal processing takes place through manipulations of the image: the convolution will calculate 32 or 16 categories for each patch of $5 \times 5$, the image is reduced to a size $14 \times 14$ pixels and then the ReLU (Rectified Linear Unit) function is applied for activation. In the next layer there are similar operations but the image is further reduced to a size of $7 \times 7$ pixels, in the penultimate layer we have a level fully connected with $1,024$ or $2,048$ neurons to allow the processing of the whole image. The images at this point are transformed into vectors, multiplied by the weight matrix, added to the bias and the ReLU is applied again. To reduce the phenomenon of overfitting, we apply a dropout, that is, with a given probability, we eliminate one or more connections of the network, and we use different probabilities. To reach the final layer of output, we

apply the Softmax regression function or multinomial logistic regression, which leads us to identify the label among one of the ten possible.
For the training phase, we use the Adam optimizer (*Adaptive Moment estimation*), a method for stochastic optimization that requires only first-order information, then the gradient, and that uses little memory. The method calculates adaptive updates of various parameters by estimating the first and second moments of the gradients [2].
In this way, we test different architectures of the network with the proposed new technique. In fact, changing the number of neurons in some layers and the probability with which we make dropout changes the number of parameters on which we are optimizing. Therefore, using the same database, we create a problem of multiple classification and a problem of binary classification.
The code contains within it a criterion, based on the verification of accuracy on the validation set, called "early stopping", to decide whether to block the process or continue it up to the maximum number of epochs [4]. We set the maximum number of epochs to $20,000$ and we ask that if the accuracy does not improve on the validation set for a number of epochs equal to 15 the exit from the training cycle is forced regardless of the number of epochs passed. At this point, the algorithm returns the number of epochs, and then compares the network so far trained (in the best case) with the test set to obtain success rates and errors committed.

### 4.3 The new idea in practice

The idea we tested is to progressively increase the size of the mini-batch. Virtually every 100 epochs (in this paper we consider epoch only the training over one mini-batch) the neural network is evaluated on the validation set and in case of 15 successive comparisons without improvement the exit from the epoch cycle is forced. The new idea, on the other hand, is to dynamically increase the sample size if no improvement is observed after 10 checks; the increase is according to the law $h = h * 2$. At this point the algorithm proceeds normally but with an increased mini-batch and if it reaches 15 total checks without improvements the exit from the epoch cycle is forced. Finally, the best result obtained from the network is evaluated on the test set.

### 4.4 Results

Since the first simulations we have noticed an improvement in performance, about a third of errors less, but associated with an inevitable increase in time, about twice. Although theoretically the algorithm can increase sample size indefinitely, from simulations, it is observed that typically the process stops around 400/800 items per sample (mini-batch size), in the multiple classifier case, and 200/400 items per sample in the binary case by forcing the exit from the loop without reaching the maximum iterate number. The table 1 report the results of accuracy and the number of errors for two different CNNs in which all the

parameters are set in the same way and the results are about the multiple classifier in the case of a CNN $32 \times 64 \times 1,024 \times 10$ with a $0,5$ dropout probability; all the results correspond to the results with the test images. The only changes are related to the size of the sample: in the STATIC case this size is always fixed at 50 and the exit from the loop is forced after 15 evaluations on the validation set in which no improvement is observed; in the DYNAMIC version, after 10 no improvement the sample size is dynamically increased; in this case too, when the 15 valuation is reached without improvement, the exit from the cycle is forced.

| DYNAMIC | STATIC |
|---|---|
| 99.27 | 99.34 |
| 99.24 | 99.27 |
| 99.35 | 99.01 |
| 99.4 | 99.12 |
| 99.42 | 99.1 |

| DYNAMIC | STATIC |
|---|---|
| 73 | 76 |
| 66 | 73 |
| 65 | 99 |
| 60 | 88 |
| 58 | 90 |

Table 1: Experiment 1: Accuracy and error number

Because of the stochastic nature of the method, the results undergo oscillations; so 5 simulations with the same criteria were repeated. In the STATIC case the average accuracy is 99.15 for an average error number of 85 while in the DYNAMIC case the average accuracy is 99.36 for an average error number of 64, thus recording an average improvement of over 20 errors.
For the robustness analysis we consider the standard deviations of the stochastic processes, in particular the standard deviation of the accuracy in the DYNAMIC case is 0.0789 and in the STATIC case is 0.1341. For this reason we can consider the new method to be more robust. We have the same result for the error numbers: in fact the standard deviation for the DYNAMIC case is 5.8566 and 10.6630 for the STATIC case. As already mentioned the times for the DYNAMIC version are dilated; in particular, the DYNAMIC network requires an average time doubled compared to the STATIC version, so it is natural to wonder if it is not enough simply to increase the sample size from the beginning so as to allow the network more time per period. The answer is no. In fact, from an experiment with an initial mini-batch of 400 it is observed that the training takes longer than the DYNAMIC version for an average accuracy of 99.2 and a number of average errors equal to 80; this data is slightly better than the 50 mini-batch version but not comparable to the improvements obtained from the dynamic version.

In a second experiment we consider the binary case with the same architecture of the CNN: $32 \times 64 \times 1,024 \times 2$ with a $0,5$ dropout probability and we compare the STATIC-50 (static case with a mini-batch size of 50 samples), the STATIC-400 (static case with a mini-batch size of 400 samples), and the DYNAMIC case.

Table 2: Accuracy and error number

| DYNAMIC | STATIC-50 | STATIC-400 | DYNAMIC | STATIC-50 | STATIC-400 |
|---|---|---|---|---|---|
| 99.88 | 99.71 | 99.72 | 12 | 29 | 28 |
| 99.77 | 99.55 | 99.75 | 23 | 45 | 25 |
| 99.81 | 99.77 | 99.8 | 19 | 23 | 20 |
| 99.87 | 99.69 | 99.82 | 13 | 31 | 18 |
| 99.81 | 99.79 | 99.77 | 19 | 21 | 23 |

Table 3: Time in seconds

| DYNAMIC | STATIC-50 | STATIC-400 |
|---|---|---|
| 140 | 113 | 149 |
| 80 | 60 | 236 |
| 86 | 106 | 159 |
| 145 | 72 | 170 |
| 77 | 127 | 213 |

In the STATIC-50 case the average accuracy is 99.7 for an average error number of 30 with an average time of 96, in the STATIC-400 case the average accuracy is 99.77 for an average error number of 23 with an average time of 185, while in the DYNAMIC case the average accuracy is 99.81 for an average error number of 17 with an average time of 106, thus recording an average improvement of over 30% errors.
For the robustness analysis we consider the standard deviations of the stochastic processes. In particular the standard deviation of the accuracy in the DYNAMIC case is 0.046, in the STATIC-50 case is 0.0944 and in the STATIC-400 case 0.0396. For this reason we notice that the new method is more robust than the STATIC-50 case but less robust than the STATIC-400 case. This is reasonable because with an increase of the mini-batch size the stochastic process is more stable. We have the same result for the error numbers: in fact the standard deviation for the DYNAMIC case is 4.6043, 9.4446 for the STATIC-50 case and 3.9623 for the STATIC-400 case.
For completeness we consider also the measure of the accuracy every time the mini-batch size increases. For example, in the DYNAMIC case with a final accuracy of 99.88, the accuracy with a 50 mini-batch size is 99.68, with a 100 mini-batch size is 99.82 and, finally, with a 200 mini-batch size is 99.88, similarly in the other cases.

In the last experiment we still consider the binary case with another architecture of the CNN: $16 \times 64 \times 2,048 \times 2$ with a $0,1$ dropout probability and we compare the STATIC-50 (static case with a mini-batch size of 50 samples), the STATIC-400 (static case with a mini-batch size of 400 samples), and the DYNAMIC.

We report only the average results: in the STATIC-50 case the average accuracy is 99.62 for an average error number of 38 with an average time of 158, in

the STATIC-400 case the average accuracy is 99.72 for an average error number of 28 with an average time of 174, while in the DYNAMIC case the average accuracy is 99.75 for an average error number of 20 with an average time of 199, thus recording an average improvement of over 30% errors.

## 5 Conclusions

In conclusion, this new method, inspired by the cognitive processes of living beings, leads to a considerable increase in the accuracy, without increasing the computational cost. The computational costs for each epoch are not increased because the networks maintain the same characteristics and the test to find out if it is necessary to increase the size of the mini-batch is done using the test that is already performed for early stopping. The fact that there is not a similar increase in the accuracy associated with a larger starting mini-batch size underscores how effectiveness actually lies in choosing an adaptive dimension during the network learning process.

## Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 Programme under the CLASS Project (https://class-project.eu/), grant agreement n 780622.
This work was partially supported also by INdAM-GNCS (Research Projects 2018).

## References

1. *Optimization Methods for Large-Scale Machine Learning*, L. Bottou, F. E. Curtis and J. Nocedal; arXiv:1606.04838.
2. *Adam: A method for stochastic optimization*, Diederik P. Kingma and J. Ba; arXiv:1412.6980, 2014.
3. *Information seeking, curiosity and attention: computational and neural mechanisms*, J. Gottlieb, P.-Y. Oudeyer, M. Lopes, and A. Baranesend; NIH Public Access.
4. *Deep learning*, Y. LeCun, Y. Bengio, and G. Hinton; Nature, 2015.
5. *Abandoning Objectives: Evolution Through the Search for Novelty Alone*, J. Lehman, K.O. Stanley; Evolutionary Computation, 2011.
6. *Don't Decay the Learning Rate, Increase the Batch Size*, S. L. Smith, P.-J. Kindermans, C. Ying, Q. V. Le; ICLR 2018 Conference.
7. *Complex system and cognitive process*, R. Serra and G. Zanarini; Ed. Springer-Verlad.