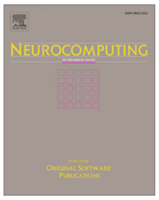
Neurocomputing 462 (2021) 534–543

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Synaptic metaplasticity for image processing enhancement in convolutional neural networks

Víctor Vives-Boix, Daniel Ruiz-Fernández *

Department of Computer Science and Technology, University of Alicante, 03690, Spain

ARTICLE INFO

Article history:
Received 24 August 2020
Revised 29 June 2021
Accepted 8 August 2021
Available online 12 August 2021
Communicated by Zidong Wang

Keywords:
Convolutional neural networks
Deep learning
Image processing
Metaplasticity
Backpropagation

ABSTRACT

Synaptic metaplasticity is a biological phenomenon shortly defined as the plasticity of synaptic plasticity, meaning that the previous history of the synaptic activity determines its current plasticity. This phenomenon interferes with some of the underlying mechanisms that are considered important in memory and learning processes, such as long-term potentiation and long-term depression. In this work, we provide an approach to include metaplasticity in convolutional neural networks to enhance learning in image classification problems. This approach consists of including metaplasticity as a weight update function in the backpropagation stage of convolutional layers. To validate this proposal, we have been used eight different award-winning convolutional neural networks architectures: LeNet-5, AlexNet, GoogLeNet, VGG16, VGG32, ResNet50, DenseNet121 and DenseNet169; trained with four different popular datasets for benchmarking: MNIST, Fashion MNIST, CIFAR-10 and CIFAR-100. Experimental results show that there is a performance enhancement for each of the convolution neural network architectures in all the datasets used.

1. Introduction

Convolutional neural networks have become the leading deep learning architecture for most image recognition, detection and classification tasks in recent years. This type of neural network was mentioned several years ago [1,2], but it became popular after the emergence of deep learning [3–5], which was powered by graphic process units (GPU) and the appearance of larger datasets [6]. Several advancements, such as the first application of maximum pooling, have also contributed to its recent popularity [7].

Current literature presents many representative enhancements related to the architecture of convolutional neural networks. One of the main contributors to these enhancements is the annual contest ImageNet Large Scale Visual Recognition Challenge or ILSVRC [8], which has become a popular way to improve and expand techniques and algorithms for both object detection and image classification tasks at large scale. This challenge has favored the appearance of some of the most used convolutional neural network architectures in recent years, with award-winning architectures such as AlexNet [9], VGGNet [10], GoogLeNet [11] or ResNet [12], among others.

However, even though learning deeper convolutional neural networks has become a trend, empirical evidences show that performance improvements cannot be accomplished by simply adding more layers [13]. This premise has generated enhancement attempts in other aspects of convolutional neural networks, such as non-linear activation functions [14–16], supervision components [17–19], regularization mechanisms [20–23] and optimization techniques [24,25].

Another popular line of research in this field is the inclusion of new bio-inspired approaches to already known methods, techniques and algorithms. Even though the concept of convolutional neural networks is already based on the visual cortex of some animals, there are several studies that tried to improve their performance by adding new bio-inspired methods or creating new neurocomputational models based on its architecture, using as reference the state of the art in neuroscience, neurobiology and other related fields [26,27].

In this work we include synaptic metaplasticity, a neurons property and a common research topic in neurobiology and neuroscience [28–32], in the backpropagation stage of convolutional neural networks. Synaptic metaplasticity directly interferes with learning and memory and it has been shown to improve performance in traditional neural networks, such as the multilayer perceptron [33]. For a deep understanding of the proposed method, we have included a brief explanation of synaptic metaplasticity

* Corresponding author.
E-mail address: druiz@dtic.ua.es (D. Ruiz-Fernández).

https://doi.org/10.1016/j.neucom.2021.08.021
0925-2312/© 2021 Published by Elsevier B.V.

in Section 2. In Section 3, we detail the proposed method where metaplasticity has been included in convolutional layers. Section 4 includes the datasets and architectures used for the experiments. Finally, Sections 5 and 6 include the results and conclusions obtained from this work, respectively.

2. Theory background

In this section we introduce the required concepts for a deep understanding of the proposed method in this work.

2.1. Metaplasticity

In a nervous system, the synapse is the connection point that allows communication between two neurons. This connection is established between the axon of the presynaptic neuron and the dendrites of the postsynaptic neuron, and the information is transmitted through molecules called neurotransmitters. The synaptic efficiency of the connection is variable and it changes depending on the previous activity of both neurons. This ability of the synapse to modulate its effectiveness is what is known as synaptic plasticity. In addition, changes in the synaptic connection efficacy may occur presynaptically to alter the properties of the neurotransmitter release or postsynaptically to modify the responsiveness to it. These changes may result in improved synaptic efficacy called long-term potentiation (LTP) or a reduction called long-term depression (LTD). Abraham [34] confirms that these long-term changes in synapse transmission properties are the basis of learning and memory, while short-term changes allow the nervous system to process and integrate information temporarily.

The concept of metaplasticity is used in neuroscience and other related fields such as psychology or physiology, to indicate a higher level of synaptic plasticity and it was initially defined for Abraham and Bear [29] as the plasticity of synaptic plasticity. Given this definition, Abraham [34] concluded that metaplasticity is highly dependent on the history of activation of the synapses; therefore, an important element in metaplasticity is the action potential. The action potential may increase (hyperpolarization) or decrease (depolarization) the postsynaptic voltage so, when the variations of the potential difference occur at the level of postsynaptic depolarization, synaptic changes are induced that intervene directly in the process that generates the metaplasticity.

Abraham [35] states that metaplasticity consists in modifying the LTP threshold according to the initial weight: the higher the weight, the higher the LTP threshold. Synaptic metaplasticity can also be observed as an induction in long-term potentiation and depression changes, enhancing the synaptic connections related to the most common events and depressing those related to less common events. Functionally, metaplasticity provides synaptic connections with the ability to integrate signals relating to plasticity across a time line. That is, metaplasticity also prevents connections from becoming too strong or too weak as a result of conventional synaptic plasticity mechanisms.

2.2. Artificial metaplasticity

The synaptic plasticity of biological neural networks has been modeled in artificial neural networks as a modification of the synaptic weights, parameters responsible for learning and performance [36]. Besides, the computational model of metaplasticity in artificial neural networks has been proposed as a modification in the way of modifying the synaptic weights so that the less frequent patterns are considered of great relevance during the learning process. Therefore, in the computational model, artificial metaplasticity is defined as a learning procedure that produces a greater modification of the synaptic weights of the less frequent patterns than of those with greater frequency, as a manner of extracting more useful information during the learning phase.

Moreover, in [37] it is shown that a feed-forward artificial neural network, such as the multilayer perceptron, trained with the backpropagation algorithm, approximates Bayes' optimal discriminant functions for both two-class and multi-class classification problems. It is also shown that the outputs of the multilayer perceptron approach the functions *a posteriori* in problems of multiclass classification, a hypothesis that is true for any network that minimizes a mean square error. This demonstration allows a direct implementation of metaplasticity as a correction factor in the synaptic weights during the learning process.

Artificial metaplasticity was first implemented by Andina [38] in a multilayer perceptron as a weighting function applied in each iteration of the backpropagation stage. The proposed weighting function was Gaussian-based to simulate the synaptic curve produced by both LTP and LTD changes due to metaplasticity and was shown to improve the multilayer perceptron performance. However, convolutional neural networks, unlike the multilayer perceptron, include a weight sharing strategy that allows the network to perform convolutions on data with the convolutional filter formed by the synaptic weights [39]. Considering this sharing strategy, we propose an artificial metaplasticity implementation in the backpropagation stage of a convolutional layer, with the aim of enhancing image processing in the early stages of the learning process before pooling operations are performed for non-linear down-sampling.

3. Proposed method

This section includes the detailed explanation of the proposed method, where metaplasticity has been added in the backpropagation stage of convolutional layers with the aim of enhancing learning in convolutional neural networks.

3.1. Forward propagation in a convolutional layer

Convolutional layers in a convolutional neural network are formed by an input map I, convolutional filters K and a bias b. In image processing, inputs have also a height H, a width W and one or three channels C, depending on whether the input images are greyscale or RGB respectively. The convolution procedure generates the following output for RGB input images:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1}\sum_{c=1}^{C} K_{m,n,c} \cdot I_{i+m,j+n,c} + b, \tag{1}$$

or the following output when the input image is in greyscale:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} K_{m,n} \cdot I_{i+m,j+n} + b, \tag{2}$$

where i and j are the iterators for each element in the input map and m and n are the iterators for every element with dimensions k_1 and k_2 in the convolutional filter. At each location, the product between each element of the convolutional filter and the input feature map element is calculated. Results for every product are then summarized to obtain the output at the appropriate location.

Convolutional layer units also have receptive fields in the input feature map and they are connected to the same number of adjacent neurons in the input layer. So, the convolution equation for an input x is given by:

$$x^l_{i,j} = \mathrm{rot}_{180^\circ}\{w^l_{m,n}\} * o^{l-1}_{i,j} + b^l_{i,j} = \sum_m \sum_n w^l_{m,n} * o^{l-1}_{i+m,j+n} + b^l_{i,j}, \tag{3}$$

where $x^l_{i,j}$ is the convolved input vector at the layer l plus the bias $b, w^l_{m,n}$ is the flipped kernel element at the layer $l, o^l_{i,j}$ is the output vector being $o^l_{i,j} = f(x^l_{i,j})$ and $f(\cdot)$ the activation function applied to the convolved input vector $x^l_{i,j}$. Filter kernels are often initialized without flipping the convolution operation but, mathematically, when a convolution is performed without flipping the kernel it is defined as a cross correlation operation [40]. Then, for a total of P predictions, the predicted outputs y_p and the targeted values t_p, the mean squared error, also abbreviated as MSE, is given by:

$$E = \frac{1}{2}\sum_p (t_p - y_p)^2. \tag{4}$$

Learning is then achieved by adjusting the weights so that the predicted output y_p is as close as possible or equals to the targeted value t_p so, during the backward propagation stage, the weights change following the gradient descent direction of the error E.

3.2. Backpropagation in a convolutional layer

The backward operation is also a convolution for both deltas and weights, so two updates are performed during this stage as in Fig. 1. During the forward propagation stage, a convolution guarantees that a pixel w in the weight kernel, contributes in the products between each element of the filter kernel and the corresponding element of the input feature map, meaning that the pixel w affects to every other element in the output feature map. Then, when there is a convolution between the input feature map with dimension $H \times W$, being H the input image height and W the width, and the weight kernel with dimension $k_1 \times k_2$, an output feature of size $(H - k_1 + 1)$ by $(W - k_2 + 1)$ is produced. Thus, the gradient for each weight can be obtained by applying the chain rule as follows:

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \frac{\partial E}{\partial x^l_{i,j}}\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \delta^l_{i,j}\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}}, \tag{5}$$

where $x^l_{i,j}$ is equivalent to $\sum_m\sum_n w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l$. Expanding this part of the equation results in:

$$\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} = \frac{\partial}{\partial w^l_{m',n'}}\left(\sum_m\sum_n w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l\right). \tag{6}$$

Further expanding this summations and taking the partial derivatives for all the components, results in zero values for all except the components where $m = m'$ and $n = n'$ in $w^l_{m,n} o^{l-1}_{i+m,j+n}$ [41]:

$$\begin{aligned}\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} &= \frac{\partial}{\partial w^l_{m',n'}}\left(w^l_{0,0} o^{l-1}_{i+0,j+0}\right) + \cdots + w^l_{m',n'} o^{l-1}_{i+m',j+n'} + \cdots + b^l \\ &= \frac{\partial}{\partial w^l_{m',n'}}\left(w^l_{m',n'} o^{l-1}_{i+m',j+n'}\right) = o^{l-1}_{i+m',j+n'}.\end{aligned} \tag{7}$$

Then, substituting Eq. (7) in Eq. (5) results in the following equation:

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \delta^l_{i,j} o^{l-1}_{i+m',j+n'} = \mathrm{rot}_{180^\circ}\{\delta^l_{i,j}\} * o^{l-1}_{m',n'}, \tag{8}$$

where the dual summation is a result of sharing weights in the convolutional neural network. The summations represent a collection from all the outputs gradients $\delta^l_{i,j}$ in a layer l. Then, by using the gradients corresponding to the filter maps, a cross correlation is performed that is transformed into a convolution by flipping the delta matrix, both horizontally and vertically, in the same manner that the filters were flipped while performing the forward propagation phase.

During the reconstruction process, the deltas are provided by the equation:

$$\delta^l_{i,j} = \frac{\partial E}{\partial x^l_{i,j}}, \tag{9}$$

which defines how affects to the loss E the changes made in a single pixel $x_{i',j'}$ within the input feature map. Introducing the summations and using the chain rule results in the following equation:

$$\frac{\partial E}{\partial x^l_{i,j}} = \sum_{i,j\in Q} \frac{\partial E}{\partial x^{l+1}_Q}\frac{\partial x^{l+1}_Q}{\partial x^l_{i',j'}} = \sum_{i,j\in Q} \delta^{l+1}_Q \frac{\partial x^{l+1}_Q}{\partial x^l_{i',j'}}, \tag{10}$$

where Q represents the output region composed of pixels that are affected by the pixel $x_{i',j'}$ in the input feature map, so this equation can also be represented as:

$$\begin{aligned}\frac{\partial E}{\partial x^l_{i,j}} &= \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \frac{\partial E}{\partial x^{l+1}_{i'-m,j'-n}}\frac{\partial x^{l+1}_{i'-m,j'-n}}{\partial x^l_{i',j'}} \\ &= \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n}\frac{\partial x^{l+1}_{i'-m,j'-n}}{\partial x^l_{i',j'}}\end{aligned} \tag{11}$$

In region Q, height ranges from $i' - 0$ to $i' - (k_1 - 1)$ and width ranges from $j' - 0$ to $j' - (k_2 - 1)$. These two values can be simply represented by $i' - m$ and $j' - n$ in the summation as the iterators m and n are in ranges from $0 \leqslant m \leqslant k_1 - 1$ and $0 \leqslant n \leqslant k_2 - 1$. In Eq. (11), $x^{l+1}_{i'-m,j'-n}$ is equivalent to $\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} o^l_{i'-m'+m,j'-n'+n} + b^{l+1}$ and expanding this part of the equation results in:

$$\begin{aligned}\frac{\partial x^{l+1}_{i'-n,j'-n}}{\partial x^l_{i',j'}} &= \frac{\partial}{\partial x^l_{i',j'}}\left(\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} o^l_{i'-m+m',j'-n+n'} + b^{l+1}\right) \\ &= \frac{\partial}{\partial x^l_{i',j'}}\left(\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} f\left(x^l_{i'-m+m',j'-n+n'}\right) + b^{l+1}\right)\end{aligned} \tag{12}$$

Expanding the summation further in Eq. (11) and using again the partial derivatives, zero values are obtained except for the components where $m' = m$ and $n' = n$ [41], function $f(x^l_{i'-m+m',j'-n+n'})$ becomes $f(x^l_{i',j'})$ and $w^{l+1}_{m',n'}$ becomes $w^{l+1}_{m,n}$ in the relevant part of the expanded summations:

$$\begin{aligned}\frac{\partial x^{l+1}_{i'-n,j'-n}}{\partial x^l_{i',j'}} &= \frac{\partial}{\partial x^l_{i',j'}}\left(w^{l+1}_{m',n'} f\left(x^l_{0-m+m',0-n+n'}\right) + \cdots + w^{l+1}_{m,n} f\left(x^l_{i',j'}\right) + \cdots + b^{l+1}\right) \\ &= \frac{\partial}{\partial x^l_{i',j'}}\left(w^{l+1}_{m,n} f\left(x^l_{i',j'}\right)\right) = w^{l+1}_{m,n}\frac{\partial}{\partial x^l_{i',j'}}\left(f\left(x^l_{i',j'}\right)\right) = w^{l+1}_{m,n}\left(f'\left(x^l_{i',j'}\right)\right)\end{aligned} \tag{13}$$

Substituting in Eq. (11) then results in:

$$\frac{\partial E}{\partial x^l_{i',j'}} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} w^{l+1}_{m,n} f'\left(x^l_{i',j'}\right) \tag{14}$$

For backpropagation we obtain a convolution expressed as a cross-correlation operation with a flipped kernel as in the following equation:

$$\begin{aligned}\frac{\partial E}{\partial x^l_{i',j'}} &= \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} w^{l+1}_{m,n} f'\left(x^l_{i',j'}\right) \\ &= \mathrm{rot}_{180^\circ}\left\{\sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} w^{l+1}_{m,n}\right\} f'\left(x^l_{i',j'}\right) \\ &= \delta^{l+1}_{i',j'} * \mathrm{rot}_{180^\circ}\{w^{l+1}_{m,n}\} f'\left(x^l_{i',j'}\right)\end{aligned} \tag{15}$$

Metaplasticity can finally be added to this equation by adding a weighting function related to the probability distribution of the input vectors and applied to the cost function to be minimized. In order to model the probability density function of the input pat-

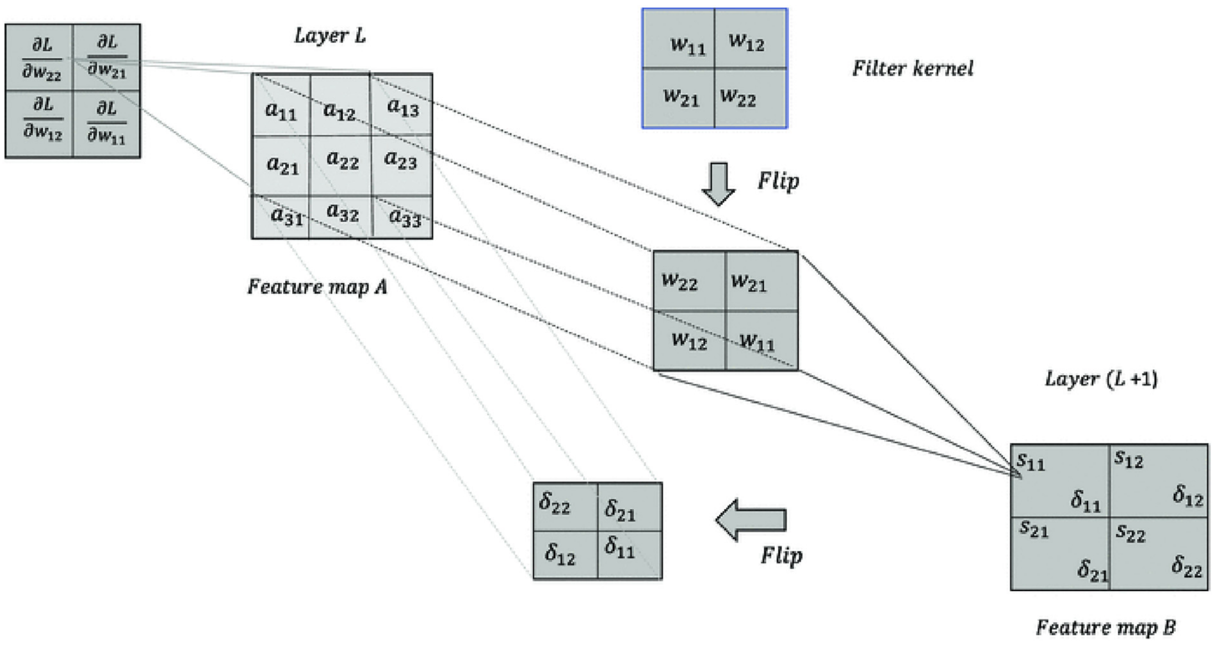


Fig. 1. Backpropagation through the convolutional layer where a convolution operation is expressed as cross-correlation operation with a flipped kernel [40].

terns, a Gaussian function representing the LTP and LTD threshold modification due to the patterns frequency, has been added in a similar way than for the multilayer perceptron in [42]:

$$f^*(x^l_{i',j'}) = \frac{A}{\sqrt{(2\pi)^N} \cdot e^{B\sum_{z=1}^{N}(x^l_{i',j'})^2_z}}, \tag{16}$$

where $f^*(\cdot)$ is the weighting function, A and B are the metaplasticity parameters representing the LTP and LTD thresholds for each initial weight and N is the number of elements in the input vector. The metaplasticity parameters have been established following the practical assumptions in [38,42]. These assumptions relate synaptic metaplasticity with the Shannon's information theory, which stays that the most uncommon patterns have a higher impact in both learning and memory procedures. In addition, previous studies have demonstrated that a classifier with backpropagation is itself an estimator of the *a posteriori* probability of the class of the input pattern [43]. Thus, metaplasticity parameters A and B have been normalized to [-1,1] and established as follows: parameter A gets higher values for infrequent patterns, meaning that the value will be closer to 1; while parameter B gets lower values for the same infrequent patterns and the value will be closer to -1. This function is applied during the training phase affecting the weights in each iteration based on an estimation of the real distribution of training patterns. For each component of the weight matrix, the weight reinforcement for each training pattern is then applied as:

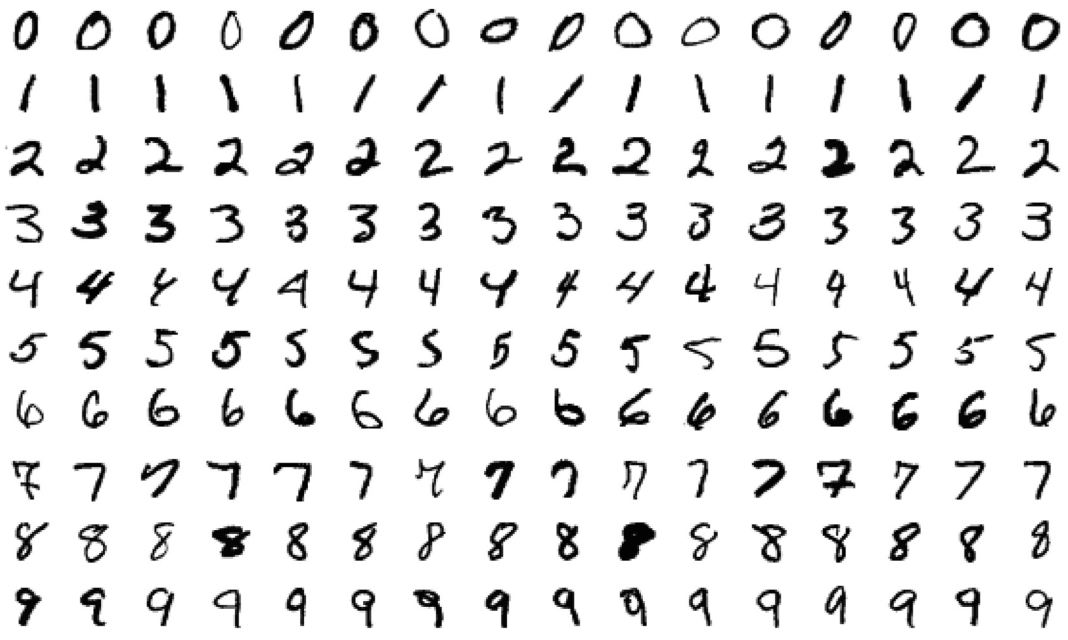


Fig. 2. Examples of the MNIST handwritten dataset.

$$w^l_{i,j}(t+1) = w^l_{i,j}(t) - \eta \frac{1}{f^*(x^l_{i,j})} \frac{\partial E}{\partial x^l_{i,j}}, \tag{17}$$

where η is the learning rate parameter. This weight update function has been replaced in the backpropagation stage in the convolutional layers of the convolutional neural networks for image classification and the resulting implementation has been tested with multiple popular architectures and datasets.

4. Experiments

This section describes the experimental setup including the convolutional neural networks architectures where the implementation has been tested, datasets used for the experiment and the implementation details where the executions have been performed.

4.1. Datasets

To find out the empirical observations addressed in this work, we have conducted the experiments on different popular datasets such as MNIST [44], Fashion-MNIST [45] CIFAR-10 and CIFAR-100 [46].

4.1.1. MNIST handwritten digit dataset

The MNIST dataset of handwritten digits is widely used for training and testing deep learning models and consists of a training set of 60,000 samples and a testing set of 10000. Digits are normalized in a bounding box of 28 $\times$ 28 pixels and all digits are centered in the image like in Fig. 2.

4.1.2. Fashion-MNIST

Fashion-MNIST is a dataset of Zalando's article images consisting of 60,000 images for training 10,000 for testing. Each example is a 28 $\times$ 28 greyscale image with a label from 10 classes. Fashion-MNIST was intended to be used as a direct replacement for the original MNIST dataset for evaluating machine learning algorithms and techniques, so this is the reason that this dataset has the same image size and structure of training and testing splits than MNIST. Fig. 3 includes some examples where each class takes three rows.

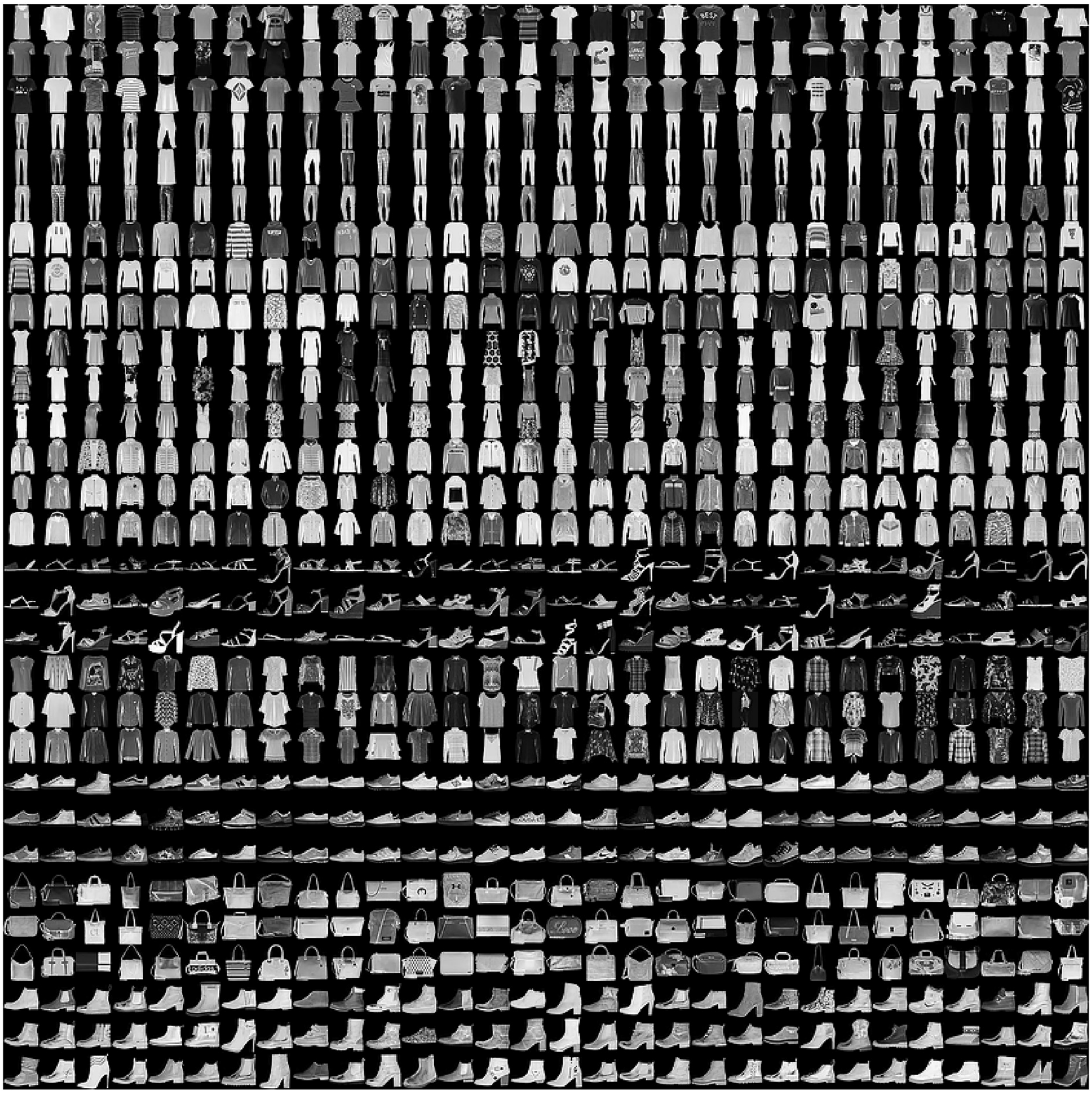


Fig. 3. Examples of the Fashion MNIST dataset [45].

4.1.3. CIFAR-10

The CIFAR-10 dataset consists of 60,000 32×32 color images labeled from 10 classes and including 6,000 images per class. There are 50,000 training images and 10,000 test images, and it is divided into five training batches and one test batch, both with 10,000 images. The test batch includes 1,000 randomly selected images from each class and the training batches contain the remaining images in a random order. Between them, the training batches contain exactly 5,000 images from each class. Fig. 4 includes examples in each row of airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships and trucks.

4.2. CIFAR-100

The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. This dataset includes 500 training images and 100 testing images per class. The 100 classes are also grouped into 20 super-classes and each image comes with a *fine* label, that represents the class to which it belongs, and a *coarse* label, that represents the super-class [46].

4.3. Convolutional neural network architectures

The experiments for MNIST, Fashion-MNIST and CIFAR-10 datasets have been performed with four award-winning convolutional neural networks: LeNet-5 [47], AlexNet [9], GoogLeNet [11] and VGG16 [10]. The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, two fully-connected layers and a softmax classifier with about 60,000 parameters. GoogLeNet is a 22-layer architecture which has inception modules and a single fully-connected layer as output with almost 7 million parameters. The

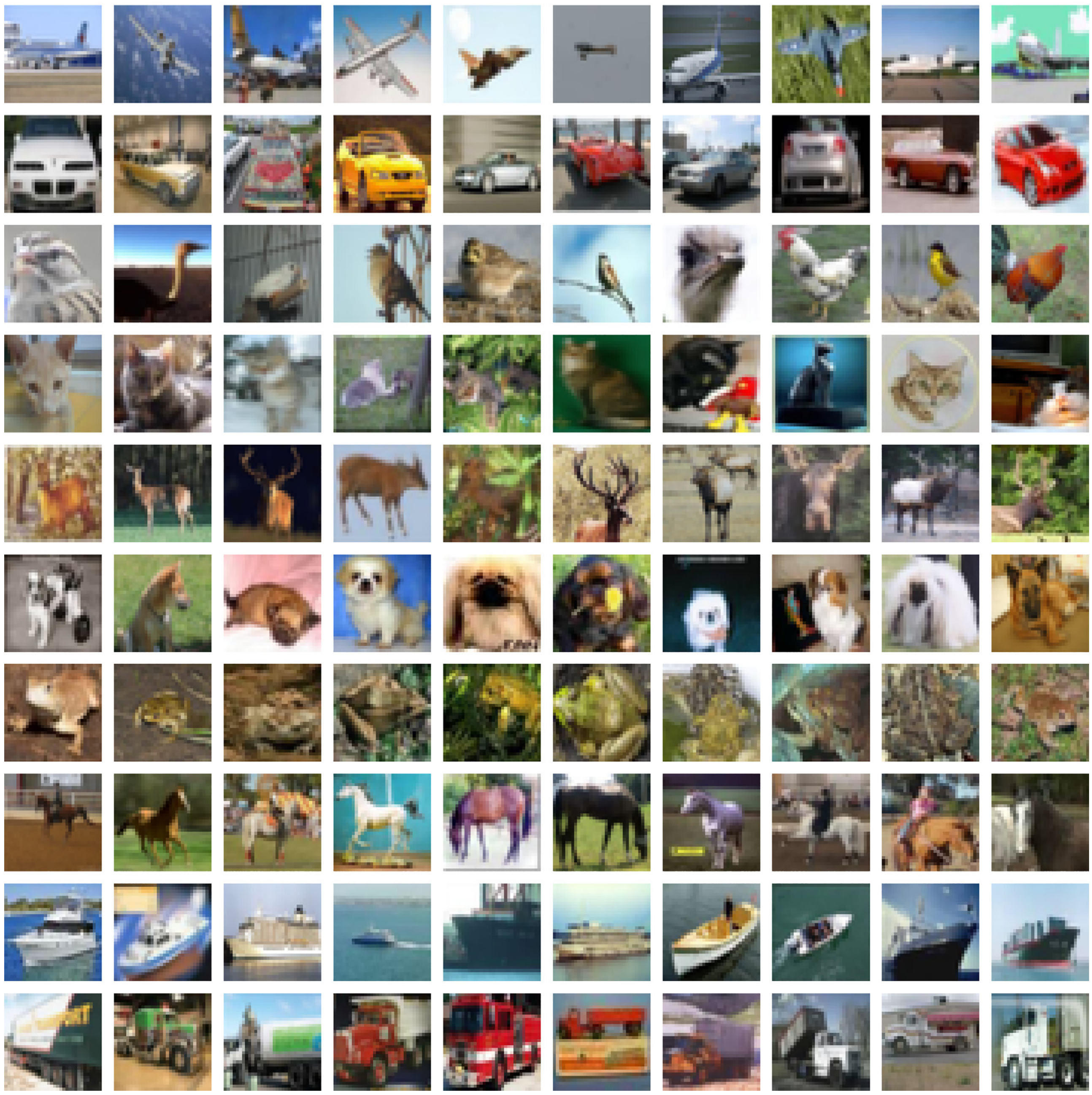


Fig. 4. Examples of the CIFAR-10 dataset [46].

AlexNet architecture comprises 5 convolutional layers, 3 max pooling layers, and 3 fully connected layers with 61 million parameters. Lastly, VGG16 has 13 convolutional layers and 3 fully connected layers with 138 million parameters.

Experiments for CIFAR-100 have been performed with larger convolutional neural networks architectures such as ResNet50 [12], VGG32 [10], DenseNet121 and DenseNet169 [48]. The ResNet50 architecture consists of five stages with convolution and identity blocks where each block type has 3 convolutional layers, with over 23 million trainable parameters. The VGG32 architecture is the same as VGG16 with the difference of having 19 layers deep instead of 16. DenseNet121 architecture consists of dense blocks and transition layers other than common convolutional and pooling layers, forming over 7 millions trainable parameters. DenseNet169 has the same architecture than DenseNet121 but with 169 layers deep instead of 121.

There are still larger convolutional neural networks architectures such as Inception-ResNet-V2 [49], Xception [50] or ResNeXt [51], among other novel approaches based on these architectures. However, due to the scope of this work and the available hardware resources detailed in next section, the architectures used in this work are sufficient for the validation of the synaptic metaplasticity in the early learning phase of convolutional neural networks.

4.4. Implementation details

Implementation for every neural network architecture has been performed by using Keras [52], a high-level neural networks API written in Python and running on Tensorflow [53], an open source software library for machine learning. Results in this work have been obtained on a system with GPU NVIDIA GeForce GTX 970, Intel Core i7 4790K CPU, 16.0 GB RAM and Windows 10 Pro. The implemented software has been published in a GitHub repository and offered as a service through an API REST and Docker containers.[1]

5. Results

In this work we have focused on the initial phase of the convolutional neural network modifying the convolutional layers for each architecture defined in the previous section. The motivation of the proposed method is to improve learning while performing convolutions in the early phase of any architecture, by adding in every convolutional layer the foundations of synaptic metaplasticity. The evaluation metrics for each architecture include accuracy, F1 score, precision and recall. Accuracy is the ratio of correctly predicted observation to the total observations; precision relates the number of correctly predicted positives versus the total of predicted positives; recall is the number of correctly predicted positive observations divided by the number of all observations in actual class; F1 score is the weighted average of both precision and recall; and last, the error loss by using the mean squared error function.

Convolutional neural network architectures with synaptic metaplasticity in their convolutional layers have been named with the prefix Am, shortly meaning artificial metaplasticity. For every architecture results are shown in Table 1 for the MNIST handwritten digits dataset, Table 2 for the Fashion-MNIST dataset, Table 3 for the CIFAR-10 dataset and Table 4 for the CIFAR-100 dataset. Results obtained show a notable improvement for each architecture and for each dataset. Every metric has been improved when using synaptic metaplasticity in convolutional layers and it can also be observed in Fig. 5 that, for ResNet, VGG and DenseNet architectures, the convergence rate increases rapidly and reaches high accuracy percentages in just a few epochs. In addition, Fig. 6 shows how metaplasticity parameters A and B adjust their values based on how common the input parameters are when using the AlexNet architecture with artificial metaplasticity. First epochs show the ignorance of the values for the network but, it can also be observed how over time the input values become more frequent and A and B parameters get closer to zero.

For the MNIST and Fashion MNIST data sets, the improvement rate is not considerably high since they are data sets with which it is easy to obtain high metrics, but even so, it can be seen that an improvement is obtained in all the architectures. For more complex data sets such as CIFAR-10, all architectures obtain excellent improvement rates, increasing the accuracy for VGG16 by 1.6%, for GoogLeNet by 0.92%, for AlexNet by 0.82% and for LeNet-5 by 2.31%. Deeper convolutional neural network architectures used in the challenging CIFAR-100 dataset also show a great improvement in both metrics and convergence rates (see Fig. 5). Every architecture used for CIFAR-100 improves between a 1% and a 2% approx-

Table 1
Results for MNIST dataset.

CNN	accuracy	f1-score	precision	recall	loss
LeNet-5	0.9886	0.9889	0.9894	0.9885	0.0344
AmLeNet-5	0.9910	0.9911	0.9912	0.9904	0.0291
AlexNet	0.9928	0.9929	0.9939	0.9919	0.0337
AmAlexNet	0.9934	0.9936	0.9943	0.9930	0.0298
GoogLeNet	0.9929	0.9929	0.9930	0.9929	0.0290
AmGoogLeNet	0.9935	0.9934	0.9939	0.9929	0.0226
VGG16	0.9910	0.9908	0.9911	0.9905	0.0353
AmVGG16	0.9933	0.9933	0.9935	0.9931	0.0303

Table 2
Results for Fashion MNIST dataset.

CNN	accuracy	f1-score	precision	recall	loss
LeNet-5	0.9014	0.9025	0.9132	0.8924	0.2802
AmLeNet-5	0.9031	0.9035	0.9136	0.8940	0.2761
AlexNet	0.9119	0.9131	0.9224	0.9042	0.2796
AmAlexNet	0.9163	0.9167	0.9243	0.9094	0.2714
GoogLeNet	0.9189	0.9190	0.9219	0.9163	0.2898
AmGoogLeNet	0.9217	0.9222	0.9254	0.9192	0.2773
VGG16	0.9245	0.9252	0.9289	0.9216	0.2555
AmVGG16	0.9269	0.9274	0.9301	0.9249	0.2399

Table 3
Results for CIFAR-10 dataset.

CNN	accuracy	f1-score	precision	recall	loss
LeNet-5	0.6992	0.6968	0.7695	0.6387	0.9242
AmLeNet-5	0.7223	0.7227	0.7963	0.6635	0.8509
AlexNet	0.7564	0.7590	0.7776	0.7419	0.9948
AmAlexNet	0.7646	0.7676	0.7843	0.7519	0.9636
GoogLeNet	0.7166	0.7180	0.7561	0.6847	0.9345
AmGoogLeNet	0.7258	0.7304	0.7689	0.6967	0.8888
VGG16	0.7840	0.7866	0.8025	0.7718	0.8645
AmVGG16	0.8000	0.8017	0.8174	0.7870	0.7760

Table 4
Results for CIFAR-100 dataset.

CNN	accuracy	f1-score	precision	recall	loss
ResNet50	0.4561	0.4641	0.4958	0.4372	3.8865
AmResNet50	0.5539	0.5664	0.6029	0.5350	2.3659
VGG32	0.5881	0.5967	0.6177	0.5777	3.004
AmVGG32	0.5950	0.6039	0.6237	0.5861	2.8506
DenseNet121	0.6056	0.6191	0.6541	0.5887	2.286
AmDenseNet121	0.6249	0.6379	0.6930	0.5922	1.778
DenseNet169	0.6169	0.6309	0.6647	0.6014	2.219
AmDenseNet169	0.6304	0.6449	0.6918	0.6053	1.8431

[1] https://github.com/vvives/metaplasticity-rest-api

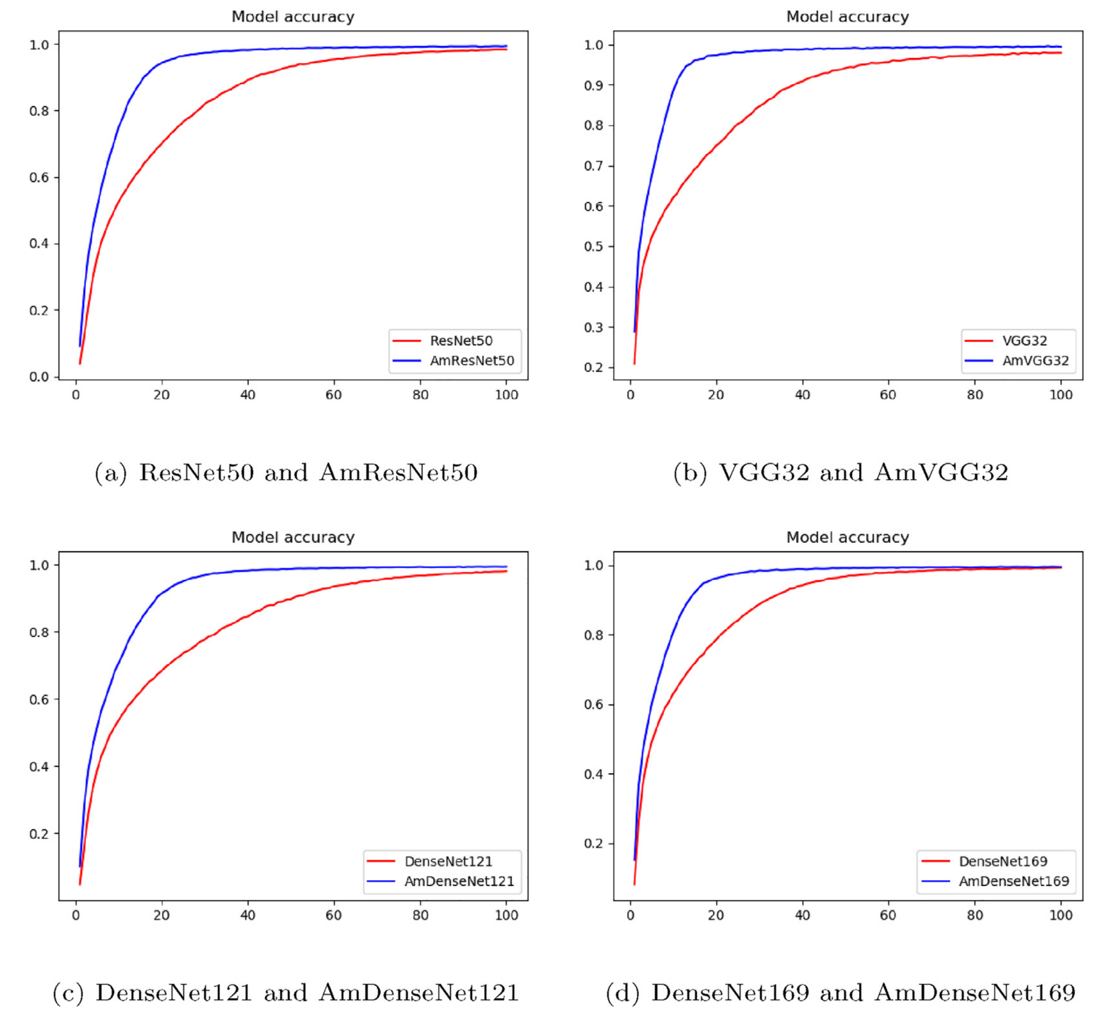


Fig. 5. Training accuracy and convergence rate comparison for convolutional neural network architectures when artificial metaplasticity is included (*Am* prefix) for the CIFAR-100 dataset.

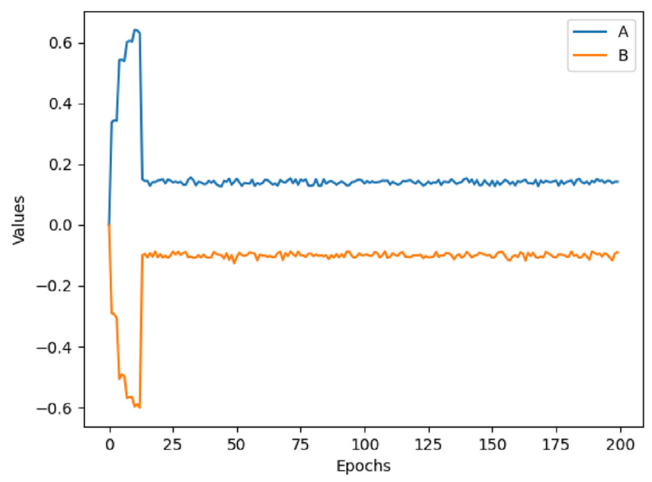


Fig. 6. Metaplasticity parameters A and B adjustment while training an AlexNet with artificial metaplasticity using the MNIST dataset.

imately in every metric, while the ResNet50 architecture has the best improvement with an accuracy increase of 9.78%, a F1-score increase of 10.24%, a precision increase of 10.71% and a recall increase of 9.78%. Results are not compared with other works because the main objective is not to obtain the best metrics in current literature, but rather to improve the performance of any architecture that uses convolutional layers with artificial metaplasticity.

6. Conclusions

In this work, synaptic metaplasticity, a biological phenomenon that directly interferes with learning and memory, has been included in convolutional neural networks. We propose an approach that consists of simulating the metaplasticity thresholds produced by long-term and long-depression changes during the learning transfer process between neurons, as a weight update function in the backpropagation stage of convolutional layers. This approach has an impact in the early learning phases of convolutional neural networks when image processing and feature maps creation tasks are performed.

Results prove a considerable improvement when this phenomenon has been included in the convolutional layers of some popular convolutional neural network architectures. The best improvement was for the residual neural network ResNet-50 with approximately a 10% improvement for every metric: accuracy, F1-score, precision and recall. Including metaplasticity in the early phases of learning has also shown to improve by far the convergence rates for every convolutional neural network architecture. In conclusion, synaptic metaplasticity is a biological phenomenon that interferes with learning and memory that, included in convolutional neural network architectures, improves both performance and convergence rate.

CRediT authorship contribution statement

Víctor Vives-Boix: Methodology, Software, Writing - original draft. **Daniel Ruiz-Fernández:** Conceptualization, Methodology, Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Computation 1 (4) (1989) 541–551, https://doi.org/10.1162/neco.1989.1.4.541.

[2] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Handwritten digit recognition with a back-propagation network, Advances in Neural Information Processing Systems 2 (1990) 396–404. url:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.5076..

[3] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (7) (2006) 1527–1554, https://doi.org/10.1162/neco.2006.18.7.1527.

[4] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507, https://doi.org/10.1126/science.1127647.

[5] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Advances in Neural Information Processing Systems (2007) 153–160.

[6] L. Deng, D. Yu, Deep learning: Methods and applications, Now Publishers Inc, vol. 7, 2013. arXiv:1309.1501, doi:10.1561/2000000039..

[7] M. Ranzato, F.J. Huang, Y.L. Boureau, Y. LeCun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2007, https://doi.org/10.1109/CVPR.2007.383157, url:http://www.cs.nyu.edu/yann.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision 115 (3) (2015) 211–252. arXiv:1409.0575, doi:10.1007/s11263-015-0816-y..

[9] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90, https://doi.org/10.1145/3065386, url:http://code.google.com/p/cuda-convnet/.

[10] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, ICLR 2015 – Conference Track Proceedings, 2015. arXiv:1409.1556v6. url:http://www.robots.ox.ac.uk/..

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 07-12-June, IEEE Computer Society, 2015, pp. 1–9. arXiv:1409.4842, doi:10.1109/CVPR.2015.7298594..

[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2016-Decem, IEEE Computer Society, 2016, pp. 770–778. arXiv:1512.03385, doi:10.1109/CVPR.2016.90..

[13] L. Shen, Z. Lin, Q. Huang, Relay Backpropagation for Effective Learning of Deep Convolutional Neural Networks, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9911 LNCS 467–482 (2015), arXiv:1512.05830, url:http://arxiv.org/abs/1512.05830.

[14] R. ZahediNasab, H. Mohseni, Neuroevolutionary based convolutional neural network with adaptive activation functions, Neurocomputing 381 (2020) 306–313, https://doi.org/10.1016/j.neucom.2019.11.090.

[15] D. Kim, J. Kim, J. Kim, Elastic exponential linear units for convolutional neural networks, Neurocomputing 406 (2020) 253–266, https://doi.org/10.1016/j.neucom.2020.03.051.

[16] M. Tanaka, Weighted Sigmoid Gate Unit for an Activation Function of Deep Neural Network, Pattern Recognition Letters 135 (2020) 354–359, https://doi.org/10.1016/j.patrec.2020.05.017, arXiv:1810.01829.

[17] W. Burton, C. Myers, P. Rullkoetter, Semi-supervised learning for automatic segmentation of the knee from MRI with convolutional neural networks, Computer Methods and Programs in Biomedicine 189 (2020), https://doi.org/10.1016/j.cmpb.2020.105328 105328.

[18] Y. Gao, L. Gao, X. Li, X. Yan, A semi-supervised convolutional neural network-based method for steel surface defect recognition, Robotics and Computer-Integrated Manufacturing 61 (2020), https://doi.org/10.1016/j.rcim.2019.101825 101825.

[19] M. Chung, J. Lee, M. Lee, J. Lee, Y.G. Shin, Deeply self-supervised contour embedded neural network applied to liver segmentation, Computer Methods and Programs in Biomedicine 192 (2020), https://doi.org/10.1016/j.cmpb.2020.105447 105447.

[20] J. Zhang, C. Ma, J. Liu, G. Shi, Penetrating the influence of regularizations on neural network based on information bottleneck theory, Neurocomputing 393 (2020) 76–82, https://doi.org/10.1016/j.neucom.2020.02.009.

[21] Q. Wu, Y. Sun, H. Yan, X. Wu, ECG signal classification with binarized convolutional neural network, Computers in Biology and Medicine 121 (2020), https://doi.org/10.1016/j.compbiomed.2020.103800 103800.

[22] X. Li, Y. Grandvalet, F. Davoine, A baseline regularization scheme for transfer learning with convolutional neural networks, Pattern Recognition 98 (2020), https://doi.org/10.1016/j.patcog.2019.107049 107049.

[23] Z. Mushtaq, S.F. Su, Environmental sound classification using a regularized deep convolutional neural network with data augmentation, Applied Acoustics 167 (2020), https://doi.org/10.1016/j.apacoust.2020.107389 107389.

[24] Y. Guo, J. Chen, Q. Du, A. Van Den Hengel, Q. Shi, M. Tan, Multi-way backpropagation for training compact deep neural networks, Neural Networks 126 (2020) 250–261, https://doi.org/10.1016/j.neunet.2020.03.001.

[25] Z. Gao, Y. Li, Y. Yang, X. Wang, N. Dong, H.D. Chiang, A GPSO-optimized convolutional neural networks for EEG-based emotion recognition, Neurocomputing 380 (2020) 225–235, https://doi.org/10.1016/j.neucom.2019.10.096.

[26] B. Wei, K. Hao, L. Gao, X. song Tang, Y. Zhao, A biologically inspired visual integrated model for image classification, Neurocomputing 405 (2020) 103–113. doi:10.1016/j.neucom.2020.04.081..

[27] Q. Wei, N. Kasabov, M. Polycarpou, Z. Zeng, Deep learning neural networks: Methods, systems, and applications, Neurocomputing 396 (2020) 130–132, https://doi.org/10.1016/j.neucom.2019.03.073.

[28] M.F. Bear, R.C. Malenka, Synaptic plasticity: LTP and LTD, Current Opinion in Neurobiology 4 (3) (1994) 389–399, https://doi.org/10.1016/0959-4388(94)90101-5.

[29] W.C. Abraham, M.F. Bear, Metaplasticity: The plasticity of synaptic plasticity, Trends in Neurosciences 19 (4) (1996) 126–130, https://doi.org/10.1016/S0166-2236(96)80018-X.

[30] B. Mockett, C. Coussens, W.C. Abraham, NMDA receptor-mediated metaplasticity during the induction of long-term depression by low-frequency stimulation, European Journal of Neuroscience 15 (11) (2002) 1819–1826, https://doi.org/10.1046/j.1460-9568.2002.02008.x, url:http://doi.wiley.com/10.1046/j.1460-9568.2002.02008.x.

[31] V. Baione, D. Belvisi, A. Cortese, I. Cetta, M. Tartaglia, E. Millefiorini, A. Berardelli, A. Conte, Cortical M1 plasticity and metaplasticity in patients with multiple sclerosis, Multiple Sclerosis and Related Disorders 38 (2020), https://doi.org/10.1016/j.msard.2019.101494 101494.

[32] A. Piva, L. Caffino, L. Padovani, N. Pintori, F. Mottarlini, G. Sferrazza, G. Paolone, F. Fumagalli, C. Chiamulera, The metaplastic effects of ketamine on sucrose renewal and contextual memory reconsolidation in rats, Behavioural Brain Research 379 (2020), https://doi.org/10.1016/j.bbr.2019.112347 112347.

[33] A. Marcano-Cedeño, J. Quintanilla-Domínguez, D. Andina, Breast cancer classification applying artificial metaplasticity algorithm, Neurocomputing 74 (8) (2011) 1243–1250, https://doi.org/10.1016/j.neucom.2010.07.019.

[34] W.C. Abraham, Activity-dependent regulation of synaptic plasticity (metaplasticity) in the hippocampus, The Hippocampus: Functions and Clinical Relevance (1996) 15–26..

[35] W.C. Abraham, Metaplasticity: Tuning synapses and networks for plasticity, Nature Reviews Neuroscience 9 (5) (2008) 387–399, https://doi.org/10.1038/nrn2356.

[36] J.R. Pelaez, M.G. Simoes, Computational model of synaptic metaplasticity, in: Proceedings of the International Joint Conference on Neural Networks, vol. 1, IEEE, 1999, pp. 6–11. doi:10.1109/ijcnn.1999.831446..

[37] D.W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, B.W. Suter, The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function, IEEE Transactions on Neural Networks 1 (4) (1990) 296–298, https://doi.org/10.1109/72.80266.

[38] D. Andina, F.J. Ropero-Peláez, On the biological plausibility of artificial metaplasticity learning algorithm, Neurocomputing 114 (2013) 32–35, https://doi.org/10.1016/j.neucom.2012.09.028.

[39] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, https://doi.org/10.1038/nature14539.

[40] S. Pattanayak, S. Pattanayak, Convolutional Neural Networks, in: Pro Deep Learning with TensorFlow, Apress, 2017, pp. 153–221. doi:10.1007/978-1-4842-3096-1_3..

[41] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning arXiv:1603.07285. url:http://arxiv.org/abs/1603.07285..

[42] D. Andina, S. Torres-Alegre, M.J. Alarcón, J.I. Seijas, M. De-Pablos-Álvaro, Robustness of artificial metaplasticity learning algorithm, Neurocomputing 151 (P1) (2015) 49–54, https://doi.org/10.1016/j.neucom.2014.07.075.

[43] D. Andina, A. Álvarez-Vellisco, J. Aleksandar, J. Fombellida, Artificial metaplasticity can improve artificial neural networks learning, Intelligent Automation and Soft Computing 15 (4) (2009) 683–696, https://doi.org/10.1080/10798587.2009.10643057.

[44] Y. LeCun, C. Cortes, MNIST handwritten digit database (2010). url:http://yann.lecun.com/exdb/mnist/..
[45] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms arXiv:1708.07747. url:http://arxiv.org/abs/1708.07747..
[46] A. Krizhevsky, V. Nair, G. Hinton, CIFAR-10 and CIFAR-100 datasets (2009). url: https://www.cs.toronto.edu/kriz/cifar.html..
[47] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2323, https://doi.org/10.1109/5.726791.
[48] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely Connected Convolutional Networks, Proceedings – 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 2017-Janua (2016) 2261–2269. arXiv:1608.06993. url:http://arxiv.org/abs/1608.06993..
[49] K. He, X. Zhang, S. Ren, J. Sun, Identity Mappings in Deep Residual Networks, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9908 LNCS (2016) 630–645, arXiv:1603.05027, url:http://arxiv.org/abs/1603.05027.
[50] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, vol. 2017-Janua, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1800–1807. arXiv:1610.02357, doi:10.1109/CVPR.2017.195..
[51] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, vol. 2017-Janua, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 5987–5995. arXiv:1611.05431, doi:10.1109/CVPR.2017.634..
[52] Keras: the Python deep learning API. url:https://keras.io/..
[53] TensorFlow. url:https://www.tensorflow.org/..

Victor Vives-Boix received the B.S. and M.S. degrees in computer science and engineering from the University of Alicante (Spain) in 2015 and 2017, respectively. He is currently an Associate Lecturer in the Department of Computer Science and Technology at the University of Alicante and is also pursuing his PhD in Computer Science and Artificial Intelligence in the same university. His research interests include machine learning, deep learning, computational medicine and bioinformatics.

Prof. Daniel Ruiz Fernández, Eng, PhD. He is computer engineer and works as a professor at the University of Alicante (Spain). He has been developing his research activity in the field of biomedical engineering for more than 15 years, more specifically in the areas of machine learning, telemedicine and clinical decision support systems (using artificial intelligence and machine learning). In these areas, he is the author of more than a hundred contributions in high impact journals and conferences. Prof. Ruiz Fernández is director of the Bioinspired Engineering and Informatics for Health (IBIS) group at the University of Alicante and academic director of the degree in Biomedical Engineering at the same university; he is also secretary of the Doctoral Commission of the Faculty of Health Sciences of the University of Alicante. Likewise, he is a member of different national and international societies such as the Spanish Society of Biomedical Engineering, the Spanish Society of Health Informatics, IEEE Computer Society and the IEEE Engineering in Medicine and Biology Society.