Hindawi
Complexity
Volume 2019, Article ID 2952304, 16 pages
https://doi.org/10.1155/2019/2952304

Research Article

Hybridizing Evolutionary Computation and Deep Neural Networks: An Approach to Handwriting Recognition Using Committees and Transfer Learning

Alejandro Baldominos, Yago Saez, and Pedro Isasi

Computer Science Department, Universidad Carlos III de Madrid. Avenida de la Universidad, 30. 28911 Leganes, Madrid, Spain

Correspondence should be addressed to Alejandro Baldominos; abaldomi@inf.uc3m.es

Received 3 December 2018; Revised 12 February 2019; Accepted 3 March 2019; Published 26 March 2019

Academic Editor: Michele Scarpiniti

Neuroevolution is the field of study that uses evolutionary computation in order to optimize certain aspect of the design of neural networks, most often its topology and hyperparameters. The field was introduced in the late-1980s, but only in the latest years the field has become mature enough to enable the optimization of deep learning models, such as convolutional neural networks. In this paper, we rely on previous work to apply neuroevolution in order to optimize the topology of deep neural networks that can be used to solve the problem of handwritten character recognition. Moreover, we take advantage of the fact that evolutionary algorithms optimize a population of candidate solutions, by combining a set of the best evolved models resulting in a committee of convolutional neural networks. This process is enhanced by using specific mechanisms to preserve the diversity of the population. Additionally, in this paper, we address one of the disadvantages of neuroevolution: the process is very expensive in terms of computational time. To lessen this issue, we explore the performance of topology transfer learning: whether the best topology obtained using neuroevolution for a certain domain can be successfully applied to a different domain. By doing so, the expensive process of neuroevolution can be reused to tackle different problems, turning it into a more appealing approach for optimizing the design of neural networks topologies. After evaluating our proposal, results show that both the use of neuroevolved committees and the application of topology transfer learning are successful: committees of convolutional neural networks are able to improve classification results when compared to single models, and topologies learned for one problem can be reused for a different problem and data with a good performance. Additionally, both approaches can be combined by building committees of transferred topologies, and this combination attains results that combine the best of both approaches.

1. Introduction

Deep Learning encompasses a broad set of techniques that are able to infer "deep" models to solve diverse machine learning problems. From these techniques, convolutional neural networks (CNNs) are probably the most well-known, extensively studied and widely used. CNNs are a type of neural network that most typically comprises two different parts: first, convolutional layers are in charge of automatically extracting relevant features from the input; then, fully-connected layers are responsible for performing supervised learning. While other architectures exist, such as residual networks [1] or fully convolutional networks [2], the one described is the most commonly found in the literature.

The advantage of CNNs is that all the network parameters, from both the feature extractor and the classifier, can be learned using backpropagation. CNNs have been widely used for computer vision, achieving an outstanding performance, but have also been applied to a variety of problems with remarkable success: natural language processing, signal classification, human activity recognition, or even music generation.

While CNNs have been proved to work well in a very diverse number of domains, one downside is that the network topologies can be very complicated, involving a large number of hyperparameters from both the convolutional layers, the fully-connected layers, and the training process itself. Generally, the topologies are manually designed, by

either choosing the most suitable one after some trial-and-error or using domain-specific expertise to infer what would constitute a good topology for the problem. Unfortunately, the first approach is very time-consuming, while the second one requires an expertise which may not always be available.

While there are no analytic procedures for automatically determining the optimal CNN topology and hyperparameters for a certain problem, in recent years some works have focused on developing mechanisms to automatically search for them. An important subset of these mechanisms involve neuroevolution, a concept that arose in the late 1980s to apply evolutionary computation for optimizing some aspects of neural networks and that, only in very recent years, with the improvement of hardware technology and efficient GPU-based deep learning frameworks, is starting to be applied to deep and convolutional neural networks.

In this work, we will first use a previously described evolutionary algorithm [3] to optimize the topology of convolutional neural networks. This procedure includes mechanisms specifically devoted to preserving the diversity of the population during evolution. Besides, in this paper, we will focus in the study on two aspects that arise naturally from the neuroevolutionary process itself.

The first of such aspects is the fact that evolutionary computation can output not just one solution, but rather a whole population of candidate solutions resulting from the optimization procedure. In this paper take advantage of the evolutionary process in order to obtain not just an optimized CNN but rather a set of outperforming CNNs that can be combined within an ensemble. This whole process is enriched by the diversity-enhancing techniques mentioned before, which will lead to a higher variance in the models conforming the ensemble.

Regarding the second aspect, it must be noticed that neuroevolution is a very expensive task in computational terms. This high cost is due to the fact that, in every generation of the genetic algorithm, each candidate CNN topology must be evaluated, thus requiring first learning its parameters using some training data and then computing its performance (fitness) using some validation data. For this reason, in this paper, we study the process of transferring the topologies optimized for one domain to a different domain. A successful outcome in this topology transfer learning task would make neuroevolution a more appealing tool since most of the knowledge could be reused between different problems.

The main contribution of this paper is therefore to address these two aspects, showing how ensembles of neuroevolved and diverse CNNs can significantly outperform individual models and how the knowledge acquired during evolution can be then transferred to a different problems with successful results.

The remainder of this paper is structured as follows: first, Section 2 introduces the context of this paper, describes related works, and elaborates on the contribution of this paper when compared with these works. The proposal of this paper is described in Sections 3 and 4, which describe the procedure for building committees of CNNs and for performing topology transfer learning and report and discuss the results attained after a systematic evaluation. Finally, Section 5 provides some conclusive remarks about the work carried out in this paper.

2. Background and Related Work

CNNs were first introduced by LeCun et al. in 1998 [4, 5] as an approach to achieve outperforming classification of different types of documents and information, such as images, speech, or time series. The most common CNN architecture comprises two distinct parts: first, a sequence of convolutional layers is in charge of automatically extracting relevant features from input data. This stage is known as "feature learning" or "representation learning" and replaces the procedure in which an expert or group of experts perform manual feature engineering to convert some unstructured information into a set of valuable features. Then, once these features have been extracted, the second part of the CNN, often known as "dense layers", will be in charge of performing classification. This classifier module will commonly comprise a fully-connected feedforward or recurrent neural network. The backpropagation mechanism is used both for learning the parameters of the dense layers (as in classical neural networks) and also for the convolutional layers, in order to minimize a loss function defined over the output of the neural network and the expected classification output.

Nowadays, CNNs have become one of the most widespread techniques in the field known as "deep learning". Many frameworks have arisen in recent years in order to easily train CNN models over very different types of input data, supporting a large variety of tensor algebra operations and automatic differentiation. Examples of widely used deep learning frameworks are Theano [6], Caffe [7], or TensorFlow [8] and some other libraries that have been also published to ease the design of convolutional neural networks, such as Lasagne [9] or Keras [10]. CNNs are becoming ubiquitous at solving a variety of artificial intelligence problems due to the availability of high-end hardware (mostly graphics processor units, GPUs, and specific hardware for tensor processing) and the ease of implementation provided by such frameworks and libraries. Nevertheless, the right topology and hyperparameters of a CNN for solving a problem are still a challenging design decision which in some cases requires expert knowledge and in most of the cases expensive trial-and-error to be done.

The problem of designing the topology of a neural network is not new: it goes back as far as the mid-1980s, when the backpropagation algorithm was introduced by Rumelhart el al. [11]. Backpropagation enabled a fast and reliable way of learning the parameters of a neural network, and it was a key discovery for settling the field of neural networks; however, the topology of the network should be known prior to the start of the training process. Unfortunately, no analytic procedure exists which determines the optimal topology of a neural network, and the common way to design it involves either trying different architectures until one satisfies our expected quality or reusing topologies that have been proved to be successful for very similar problems.

However, by the end of the 1980s, a new approach arose: neuroevolution. This paradigm uses evolutionary algorithms in order to optimize the topology, and in some cases also the weights, of a neural network. Evolutionary algorithms are a set of biologically inspired, metaheuristic search techniques that can optimize a population of individuals in order to maximize a certain fitness metric. In neuroevolution, the population comprises the description of the neural network topologies and/or hyperparameters, and the fitness function is a certain machine learning metric to be optimized (e.g., accuracy, precision or recall, F1 score, etc.).

Neuroevolution has been applied successfully to determine optimal neural network topologies for almost three decades. However, its application to CNNs has been marginal, and only in very recent years a significant amount of works have been published proposing different approaches to perform an evolutionary optimization of CNN topologies. A very recent survey has been provided by Stanley et al. early in 2019 [12].

One of the earliest approaches was proposed by Koutník et al. [13] in 2014, where an architecture of four convolutional layers with max-pooling and a small recurrent network with three hidden units is fixed. All parameters are evolved by encoding them in a real-valued genome. However, as it can be seen, this work do not perform architecture optimization.

Other work was presented in 2015 by Verbancsics and Harguess [14], proposing an update to HyperNEAT [15] to support the evolution of CNNs, by learning the weights of a feature extractor consisting on convolutional layers. Also this year, Young et al. [16] introduced MENNDL (multi-node evolutionary neural networks for deep learning), where a genetic algorithm was used to optimize only six hyperparameters of a CNN, focusing on high performance computing. A more recent version was presented in 2017 [17], where the number of evolved hyperparameters was increased to eleven.

Some relevant works were also published during 2016. It is the case of the work by Loshchilov and Hutter [18], where an evolution strategy is used to evolve 19 hyperparameters from a CNN, most of which belong to the learning process, using an architecture with a fixed number of layers. Other work was published by Fernando et al. [19], where they propose the introduction of a differentiable compositional pattern producing network (DPPN), a type of network which can be evolved using approaches based on augmenting topologies. An interesting novelty of this approach is that the topology and the initial weights can be evolved altogether and later optimized using backpropagation.

Most works in this field started to arise in 2017. A representative example is GeNet, introduced by Xie and Yuille [20], whose approach consists of evolving a graph connecting different stages (each corresponding to a convolutional layer), and which allows for the optimization of complex nonsequential networks. Also relevant is CoDeepNEAT by Miikkulainen et al. [21], which consists of a modification of NEAT to support the evolution in CNNs, with support to nonsequential architectures and improved with a coevolutionary approach. In EXACT, proposed by Desell [22], another NEAT-approach is used to evolve the filter sizes of convolutional layers and their connectivity. Real et al. [23] from Google Brain also proposed an approach where a graph was evolved, with each node corresponding to a convolutional layer, allowing for complex topologies. Moreover, Sun et al. [24] described the application of a GA to the evolution of CNNs, innovating with the introduction of variable-length chromosomes. Also, Dufourq and Bassett [25] introduced EDEN, which relied on a genetic algorithms with two genes for encoding the network architecture and the learning rate.

In the works proposed by Suganuma et al. [26] and by Davison [27], genetic programming is used instead for evolving the architecture of the CNN. Meanwhile, Bochinski et al. [28] proposed IEA-CNN, an approach using an evolutionary strategy, innovating by sorting the evolved layers by descending complexity, effectively reducing the search space factorially on the number of layers. Additionally, they extend their contribution by building ensembles out of evolved models, using a fitness function that takes the global classification error of the population, and naming this alternative CEA-CNN.

State-of-the-art works have been also presented during 2018. For example, Baldominos et al. [3] conducted a research where two implementations of evolutionary algorithms were successfully used to evolve the topologies of CNNs for improving the performance of handwritten digit recognition. Liu et al. [29] suggested using a genetic algorithm for evolving individuals using mutation by adding, removing, or editing edges in a computation graph which can be translated into a convolutional neural network. Finally, first Kramer [30] and later Prellberg and Kramer [31] have presented an approach based on an evolutionary algorithm that relies only on the mutation operator and have introduced a mechanism to support parameters inheritance, so that descendants during the evolutionary process do not need to learn weights from scratch. Assuno et al. [32] have presented DENSER, a work where a multi-level encoding of candidate solutions allow for the optimization of the topology of the network and the activation functions, with authors claiming that it can be used also to evolve the hyperparameters of the learning process as well as of the data augmentation stage. In late 2018, Wang et al. [33] used differential evolution to optimize different hyperparameters of both convolutional and fully-connected layers. Sun et al. [34] have recently proposed the application of a genetic algorithm where two building blocks (ResNet and DenseNet) are used to evolve the CNN architecture.

In this promising field, there are still many research lines that are worth being explored. As we already described in the previous section, in this paper we expect to contribute to the field of knowledge by working in two different subjects of study that arise naturally from the advantages and limitations of neuroevolution. First, given that evolutionary algorithms evolve not only a single individual but rather a population of them, we hypothesize that the output of the optimization process can be combined altogether to build a committee of neural networks that perform better than any single model.

The idea of ensembling several neural network models into a committee is not new and has been extensively carried out in the literature. For example, in the field of image

classification, Cirean et al. [35] attained an improvement of 0.8 percentage points over the best result reported in the state of the art of the MNIST database by using committees of CNNs. In recent years, this idea has been applied to a variety of fields, such as facial expression analysis [36], astrophysics [37], pose estimation [38], or medical imaging [39]. However, the idea of building an ensemble out of a population of neuroevolved CNN topologies is less common and, to the best of our knowledge, has been only explored before by Real et al. [23] and by Bochinski et al. [28] in 2017. In the former work, the ensemble is built by choosing the top-2 models of the evolved population based on validation accuracy. When testing over the CIFAR10 dataset, the ensemble attains an accuracy of 95.6%, versus 94.6% of the best single model. Beyond neuroevolution, the process of ensembling automatically determined CNN topologies is also used in MetaQNN [40], where reinforcement learning is used to determine optimal topologies. As a result, ensembles attain a test accuracy over the MNIST dataset of 99.68%, compared to 99.56% when using a single model. In the latter text, ensembles are evolved using a fitness function that considers the global classification error of the population, and authors report a classification accuracy of 99.76% with an ensemble of 34 CNNs, compared to 99.66% with a single model, a very competitive performance among the state of the art.

Regarding the second subject of study, since neuroevolution is an expensive process, we believe that it is worth exploring whether the best topology evolved for a problem can be reused for a similar problem, even when the data are different. The deep learning literature has studied transfer learning before, but understood as the application of a trained machine learning model to a different domain with very few additional trainings [41]. In the case of neural networks, the learned parameters (weights) are reused for a different problem. In some cases, only the feature representations (i.e., the parameters of the convolutional layers) may be used, training the classifier from scratch. In this paper, however, we will be focusing on topology transfer learning. Thus, we are interested in transferring knowledge not of the CNN model weights but rather of its architecture and learning hyperparameters. In other words, once the expensive process of determining the best CNN topology for solving a certain problem is completed, we are interested in testing whether this evolved topology is useful for training a CNN model in a different machine learning problem. In the case of being useful, then plenty of computation time could be saved by avoiding the repetition of the evolutionary process. This approach contrasts with the one followed by Real et al. [23], where they run again the complete neuroevolutionary process for the CIFAR-100 dataset, using the same encoding and hyperparameters that they used for CIFAR-10.

In this paper, we will explore both alternatives separately and then combine both of them together to put the robustness of these improvements in the neuroevolutionary process to the test. To the best of our knowledge, the study of the use of ensembles along with topology transfer learning within the context of neuroevolution is novel and has not been addressed earlier in the literature.

3. Committees of Convolutional Neural Networks

A committee of machine learning models, commonly referred to as an "ensemble", is a set of models that operate together in order to provide a single response. In a classification problem, each model will return a class and then all responses will be considered to produce a single outcome.

In this paper, we will rely on the neuroevolutionary process described in the work by Baldominos et al. [3], which already implement specific mechanisms to preserve the diversity of the population of CNN topologies. This mechanism consists in a niching strategy where individuals more similar to others in the population are penalized. To do so, an adjusted fitness (f_a) for the i-th individual (i_i) is computed as follows, being $f(i_i)$ the non-adjusted fitness for such individual:

$$f_a(i_i) = f_n(i_i) \times \left(1 - \frac{\sum_{i_j \in P, j \neq i} sim(i_i, i_j)}{|P| - 1}\right) \quad (1)$$

In the previous equation, $sim(i_i, i_j)$ is a value that represents the similarity of two individuals and is computed as follows:

$$sim(i_i, i_j) = \begin{cases} \frac{|p(i_i) \cap p(i_j)|}{|p|}, & \text{if } i_i[n_c] = i_j[n_c] \text{ and } i_i[n_d] = i_j[n_d] \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

In the formula for computing similarity, n_c refers to the number of convolutional layers and n_d refers to the number of fully connected layers. We can see that individuals with a different number of layers are considered completely different. When the number of layers coincide, then the formula looks for the fraction of hyperparameters whose value is equal between both individuals. From this formula, it is easy to check that the image of the similarity function is in the range $[0, 1]$, where 0 means that two individuals are completely different and 1 means that they share the exact same setup.

By using this niching strategy, we expect the increase in diversity to result in better ensembles, since models are guaranteed to be very different from each other.

During the neuroevolutionary process, the top 20 topologies found during the evolution are stored in a hall-of-fame and then trained for a longer time to come up with competitive models. Then, we will build committees of CNNs from the best models found during the neuroevolutionary procedure. To do so, we will sort the models based on their performance and then build committees by adding models one at a time, up to a maximum of 20 models. The committee will work based on a majority voting policy; that is, its response will correspond to the class returned by the majority of the models comprising the committee. In case of a tie, it will be resolved by choosing the class decided by the most competitive model involved in the tie.

3.1. Evolution of CNN Models. Following the procedure described by Baldominos et al. [3], we will use both genetic algorithms (GA) and grammatical evolution (GE) to automatically determine the topology and hyperparameters of CNNs that maximize the classification performance.

The GA encoding consists in a 69-bit binary string using Gray encoding. The chromosome encodes the following parameters:

(i) Input configuration: batch size from 25 to 200.

(ii) Convolutional layers: number of convolutional layers, and, for each convolutional layer, number of kernels, kernel size, pooling size (or no pooling), and activation function.

(iii) Dense layers: number of dense layers, and, for each dense layer, type of the layer (feedforward or recurrent), number of neurons, activation function, weights regularization (L1 or L2), and dropout rate.

(iv) Learning process: gradient descent function (stochastic gradient descent, Adam, Adamax, etc.) and learning rate.

In GE, the approach is similar but the individuals' phenotype is defined by a language generated by a grammar specified in Backus-Naur form (BNF), which can be found in Algorithm 1. The genotype in GE consists in a sequence of integers which are used to choose production rules from the grammar until a valid string is generated (i.e., there are no remaining non-terminal symbols). The set of valid strings (the language) correspond to the search space of candidate solutions. This encoding provides better flexibility than the GA since it reduces redundancy.

The learning procedure takes place as follows: the networks represented by each chromosome have been trained only for 5 epochs and using a random 50% sample of the training set in each epoch. This allows generating fairly tight estimates of the performance of the networks, saving the time necessary for the evaluation of individuals without whom the evolutionary process would be unfeasible. When evaluating the fitness of an individual, the niching strategy described earlier is used to compute the adjusted fitness.

The complete procedure for evaluating each candidate solution is as follows:

(1) Translate the genotype into a phenotype by creating a CNN topology with the parameters specified by the genotype.

(2) Randomly initialize the network's weights.

(3) Train the network during 5 epochs, using a random 50% of the data in each epoch.

(4) Compute the classification error of the network on a set that is different from the training set and assign the obtained error as the individual fitness, which must be minimized.

(5) Compute the adjusted fitness provided the niching strategy.

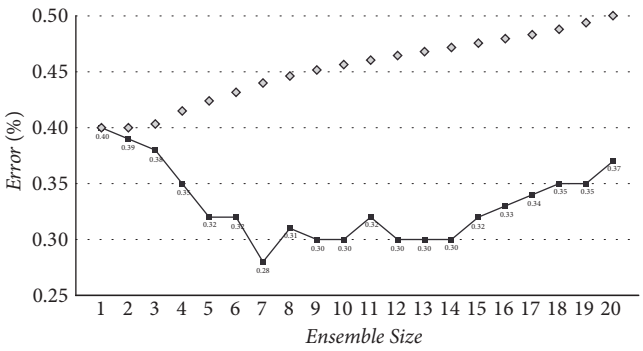

Figure 1: Error rate of CNNs committees with up to 20 models from GA with MNIST.

Throughout the execution of the evolutionary system, the best individuals found thus far are stored such that at the end of the process we will have a set called *hall-of-fame*, with the 20 best architectures found throughout the evolutionary process. The hyperparameters of each of the top 10 topologies for the GA are detailed in Table 1. Each of these architectures is trained for 30 epochs and without sampling. To avoid biases resulting from the stochastic nature of the process, each topology is trained 20 times. The results of this full training stage are summarized in Table 2 (each row corresponds to each topology in the hall of fame, whereas columns describe the distribution of performance of fully trained models).

3.2. Results and Discussion of Committees of CNNs. For the sake of clarity and economy, throughout this section, we will be more exhaustive when describing the results obtained using GA, which are slightly better, with a very small difference, than those attained by GE.

For the ensemble, we use the 20 models generated at the end of the evolutionary process (we only choose the best for each different topology) and will follow a majority-voting policy: for each instance in the validation set, the class predicted by the 20 models is calculated, and the one that obtains the largest consensus is generated.

The error rates of the 20 ensembles are shown in Figure 1 in the dark line. It is worth recalling that we are testing different ensembles size by adding models one at a time, adding first those with better performance. Light-colored diamonds in the figure show, just for reference, the average error rate of all the models included in the ensemble. Because classifiers are sorted by ascending error, this average increases as more classifiers are added.

As we expected, the error rate of a committee is lower than the average error rate of its components in all cases. It can be seen how the introduction of a few models decreases the error down to 0.28% (when 7 classifiers are used) and then stabilizes for a while, around 0.3%, until it starts to increase again when using more than 14 classifiers. In the case of GE, the best committee found led to an error rate of 0.29%.

It is worth noting that the best committee found in our research is the one involving the best 7 classifiers. As said before, this ensemble classifies the MNIST test set with just an error rate of 0.28%. To the best of our knowledge, when considering those works where no data augmentation is used

```
< dnn >          ::= <input> < conv_lys > < dense_lys > < opt_setup >
<input>          ::= < batch_size >
< batch_size >   ::= 25 | 50 | 100 | 150
< conv_lys >     ::= < conv > | < conv > < conv > | < conv > < conv > < conv >
< conv >         ::= < n_kernels > < k_size > < act_fn > <pooling>
< n_kernels >    ::= 8 | 16 | 32 | 64 | 128 | 256
< k_size >       ::= 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<pooling>        ::= null | < p_size >
< p_size >       ::= 2 | 3 | 4 | 5 | 6
< dense_lys >    ::= <dense> | <dense> <dense> | <dense> <dense> <dense>
<dense>          ::= < d_type > < n_units > < act_fn > < reg_fn > < dropout_r >
< d_type >       ::= rnn | lstm | gru | feedforward
< n_units >      ::= 32 | 64 | 128 | 256 | 512 | 1024
< act_fn >       ::= relu | linear
< reg_fn >       ::= null | l1 | l2 | l1l2
< dropout_r >    ::= 0 | 0.5
< opt_setup >    ::= < opt_type >  < learn_rate >  < batch_size >
< opt_type >     ::= sgd | nesterov | momentum | adagrad | adamax | adam | adadelta | rmsprop
< learn_rate >   ::= 5E-1 | 1E-1 | 5E-2 | 1E-2 | 5E-3 | 1E-3
```

Algorithm 1: Definition of the grammar in Backus-Naur Form for the MNIST dataset. *Source:* Baldominos et al. [3].

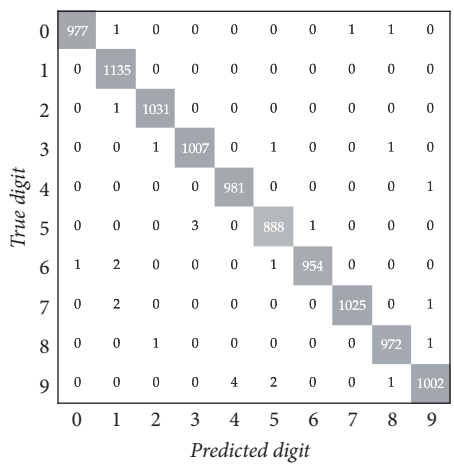

Figure 2: Confusion matrix of the best committee found for MNIST.

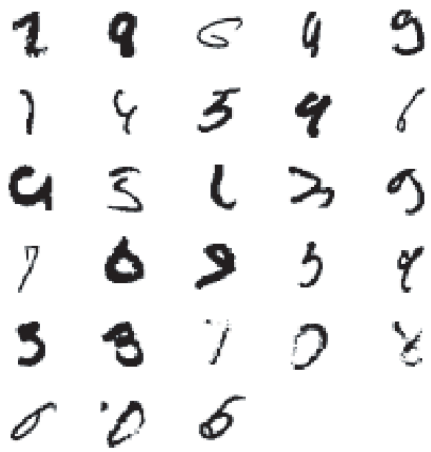

Figure 3: The 28 misclassified images in the MNIST test set with the best committee found.

at all, this result is only outperformed by the works by Chang and Chen [42] and by Bochinski et al. [28], whose proposals have both resulted in a test error rate of 0.24%.

An error rate of 0.28% over a test set of 10000 samples translates into 28 incorrectly classified samples. The confusion matrix of the best model is shown in Figure 2. The interpretation of this confusion matrix is very interesting: we can see how the most frequent error involves the number '9' being classified as a '4'. Some other common mistakes involve recognizing '3' instead of '5' or '1' instead of '7'. Mixing up these numbers may be acceptable if they are poorly written, as is the case of these samples. To be more specific, the 28 images that were misclassified are depicted in Figure 3. We can see how those manuscript digits are indeed very unclear or poorly written. For example, the fourth image in the first row could be either a '4' or a '9', the third image in the second row could be a '3' or a '5', etc. It can difficult even for humans to recognize these numbers properly.

It is remarkable that the best classifier obtained using a single model translates into a test error rate of 0.40% (see Table 2), whereas using ensembles of neuroevolved models makes it possible to reduce this error rate down to 0.28%. This makes the use of evolutionary systems even more interesting, since it allows not only the automatic generation of the architectures and parameters of CNNs, already a complex and sophisticated task, but also a considerable improvement of the performance of the CNNs through the use of an ensemble of the evolved models, which manually would be almost impossible to attain.

4. Topology Transfer Learning

The second objective of this work is to verify whether a CNN model or committee optimized for a certain domain can be transferred directly to a different (but similar) domain,

Table 1: Top 10 topologies in the hall-of-fame for the GA in the MNIST dataset. **c** (convolutional), **d** (dense), $\boldsymbol{ck_i}$ (kernels in the i-th conv. layer), $\boldsymbol{cs_i}$ (kernel size in the i-th conv. layer), $\boldsymbol{cp_i}$ (pooling size after the i-th conv. layer–1 means no pooling), $\boldsymbol{ca_i}$ (activation in the i-th conv. layer), $\boldsymbol{dt_i}$ (type of the i-th dense layer), $\boldsymbol{dn_i}$ (number of neurons in the i-th dense layer), $\boldsymbol{dd_i}$ (dropout in the i-th dense layer), $\boldsymbol{da_i}$ (activation in the i-th dense layer), $\boldsymbol{dr_i}$ (regularization in the i-th dense layer), $\boldsymbol{B}$ (batch size), $\boldsymbol{f}$ (optimizer), $\boldsymbol{\eta}$ (learning rate).

#	Architecture					
1		$B = 25$	f = AdaMax	$\eta = 0.001$		
	C	$ck_1 = 64$	$cs_1 = 6$	$cp_1 = 1$	ca_1 = linear	
		$ck_2 = 128$	$cs_2 = 5$	$cp_2 = 1$	ca_2 = ReLU	
		$ck_3 = 128$	$cs_3 = 8$	$cp_3 = 1$	ca_3 = ReLU	
	D	dt_1 = feed-forward	$dn_1 = 128$	$dd_1 = 0$	da_1 = ReLU	dr_1 = none
2		$B = 50$	f = Nesterov	$\eta = 0.001$		
	C	$ck_1 = 64$	$cs_1 = 4$	$cp_1 = 1$	ca_1 = ReLU	
		$ck_2 = 128$	$cs_2 = 4$	$cp_2 = 1$	ca_2 = ReLU	
		$ck_3 = 256$	$cs_3 = 4$	$cp_3 = 2$	ca_3 = ReLU	
	D	dt_1 = feed-forward	$dn_1 = 128$	$dd_1 = 0.5$	da_1 = linear	dr_1 = none
		dt_2 = feed-forward	$dn_2 = 512$	$dd_2 = 0$	da_2 = linear	dr_2 = none
3		$B = 150$	f = Nesterov	$\eta = 0.05$		
	C	$ck_1 = 128$	$cs_1 = 5$	$cp_1 = 2$	ca_1 = ReLU	
		$ck_2 = 256$	$cs_2 = 5$	$cp_2 = 4$	ca_2 = linear	
	D	dt_1 = feed-forward	$dn_1 = 32$	$dd_1 = 0$	da_1 = linear	dr_1 = none
4		$B = 50$	f = RMSProp	$\eta = 0.001$		
	C	$ck_1 = 32$	$cs_1 = 3$	$cp_1 = 1$	ca_1 = ReLU	
		$ck_2 = 64$	$cs_2 = 5$	$cp_2 = 1$	ca_2 = ReLU	
		$ck_3 = 32$	$cs_3 = 8$	$cp_3 = 1$	ca_3 = ReLU	
	D	dt_1 = feed-forward	$dn_1 = 256$	$dd_1 = 0.5$	da_1 = linear	dr_1 = none
5		$B = 25$	f = Momentum	$\eta = 0.01$		
	C	$ck_1 = 64$	$cs_1 = 4$	$cp_1 = 1$	ca_1 = ReLU	
		$ck_2 = 256$	$cs_2 = 8$	$cp_2 = 1$	ca_2 = ReLU	
		$ck_3 = 128$	$cs_3 = 7$	$cp_3 = 3$	ca_3 = linear	
	D	dt_1 = feed-forward	$dn_1 = 128$	$dd_1 = 0$	da_1 = linear	dr_1 = none
6		$B = 50$	f = AdaMax	$\eta = 0.005$		
	c	$ck_1 = 8$	$cs_1 = 4$	$cp_1 = 1$	ca_1 = ReLU	
		$ck_2 = 128$	$cs_2 = 6$	$cp_2 = 5$	ca_2 = linear	
	d	dt_1 = feed-forward	$dn_1 = 512$	$dd_1 = 0$	da_1 = ReLU	dr_1 = none
7		$B = 25$	f = AdaGrad	$\eta = 0.01$		
	c	$ck_1 = 32$	$cs_1 = 6$	$cp_1 = 2$	ca_1 = ReLU	
		$ck_2 = 128$	$cs_2 = 4$	$cp_2 = 2$	ca_2 = linear	
	d	dt_1 = rnn	$dn_1 = 32$	$dd_1 = 0$	da_1 = linear	dr_1 = l2
		dt_2 = feed-forward	$dn_2 = 1024$	$dd_2 = 0.5$	da_2 = ReLU	dr_2 = none
8		$B = 50$	f = AdaGrad	$\eta = 0.01$		
	c	$ck_1 = 128$	$cs_1 = 7$	$cp_1 = 1$	ca_1 = ReLU	
		$ck_2 = 256$	$cs_2 = 2$	$cp_2 = 2$	ca_2 = ReLU	
		$ck_3 = 128$	$cs_3 = 9$	$cp_3 = 2$	ca_3 = ReLU	
	d	dt_1 = feed-forward	$dn_1 = 32$	$dd_1 = 0$	da_1 = linear	dr_1 = none
		dt_2 = feed-forward	$dn_2 = 64$	$dd_2 = 0$	da_2 = linear	dr_2 = none
9		$B = 25$	f = AdaGrad	$\eta = 0.01$		
	c	$ck_1 = 64$	$cs_1 = 8$	$cp_1 = 2$	ca_1 = ReLU	
		$ck_2 = 256$	$cs_2 = 5$	$cp_2 = 3$	ca_2 = ReLU	
	d	dt_1 = feed-forward	$dn_1 = 256$	$dd_1 = 0.5$	da_1 = ReLU	dr_1 = none

Table 1: Continued.

#		Architecture				
10		$B = 50$	$f =$ SGD	$\eta = 0.05$		
	c	$ck_1 = 64$	$cs_1 = 7$	$cp_1 = 1$	$ca_1 =$ ReLU	
		$ck_2 = 128$	$cs_2 = 7$	$cp_2 = 4$	$ca_2 =$ linear	
	d	$dt_1 =$ feed-forward	$dn_1 = 256$	$dd_1 = 0$	$da_1 =$ linear	$dr_1 =$ none
		$dt_2 =$ feed-forward	$dn_2 = 512$	$dd_2 = 0.5$	$da_2 =$ linear	$dr_2 =$ none

Table 2: Summary of errors (in %) of the best 20 GA and GE individuals after full training in MNIST. *Source:* Baldominos et al. [3].

	GA		GE	
#	Mean	Best	Mean	Best
1	0.5130	0.41	0.4680	0.41
2	0.5380	0.49	0.4765	0.42
3	0.5485	0.49	0.5310	0.47
4	0.5875	0.50	0.4300	0.37
5	0.6115	0.46	0.5545	0.47
6	0.5010	0.45	0.5265	0.42
7	0.5760	0.50	0.4615	0.40
8	0.6085	0.53	0.5830	0.50
9	0.4795	0.40	0.5530	0.48
10	0.6090	0.57	0.5330	0.48
11	0.5600	0.49	0.4985	0.44
12	0.5850	0.54	0.4815	0.42
13	0.7095	0.60	0.5090	0.42
14	0.6045	0.54	0.5680	0.50
15	0.5780	0.47	0.5095	0.48
16	0.6265	0.51	0.5245	0.45
17	0.6010	0.51	0.5525	0.48
18	0.6815	0.62	0.6015	0.52
19	0.4805	0.40	0.5100	0.46
20	0.5930	0.52	0.5525	0.48

resulting in a performance that is good enough. This would prove the robustness of the models generated by evolution and would allow extending them to new data, without being forced to repeat the whole process again. In order to validate those transfer capabilities, we decide to use the EMNIST database, which shares structure with MNIST, the one used for evolving the models, and comprises a similar problem: handwriting recognition. We hypothesize that topologies that performed well with MNIST should also be able to attain reasonably good results with EMNIST. In this section, we will explore the problem of transferring a CNN topology learned using neuroevolution.

4.1. The EMNIST Database. EMNIST (Extended MNIST) database was introduced in 2017 by Cohen et al. [43] and consists of a set of handwritten characters (both digits and letters).

Figure 4 shows ten samples for each letter in the EMNIST dataset, including both uppercase and lowercase variants, and two samples for each digit (in the last two columns).

EMNIST database is derived from NIST Special Database 19 [44], which contains NIST's (National Institute of Standards and Technology of the US) entire corpus of training materials for handprinted document and character recognition. It contains over 810,000 isolated characters from 3,699 writers [45] who filled a form. These characters have been labelled after manually checking.

Authors releasing EMNIST admit that, in the past years, deep learning and convolutional neural networks have allowed scientists to achieve accuracies over 99.7% in the MNIST dataset, stating that at that point "*the dataset labeling can be called into question*" [43]. For this reason, they suggest that MNIST has become a non-challenging benchmark.

Even though NIST Special Database 19, from which EMNIST was extracted, was available since 1995, it has remained mostly unused. The main reason was that this dataset was difficult to access and challenging to use in modern computers. Recently, in 2016, NIST has released a second edition of this database [45] which is much easier to access.

Figure 4: Samples of all letters and digits in the EMNIST dataset.

In order to make both compliants in terms of structure, authors have performed a processing similar to the one done with the MNIST database. The result is a dataset that contains more instances than MNIST, includes letters apart from digits, and, in consequence, is a more challenging benchmark for evaluating the performance of character recognition systems. This processing comprises the next steps:

(1) Original images in the NIST Special Database 19 are stored as 128x128-pixel BW images.

(2) A Gaussian blur filter with $\sigma = 1$ is applied to soften the edges.

(3) Blank padding is removed, reducing the image to the region of interest (the actual digit).

(4) The image is then centered in a square image while preserving the aspect ratio, padding it with a 2-pixel border.

(5) The image is downsampled to 28x28 pixel using bicubic interpolation.

As a result, each instance in the EMNIST database is a 28x28-pixel grayscale image, where each pixel is a number between 0 and 255.

In this paper we will use two different taxonomies provided by the EMNIST dataset:

(i) *Digits*: similar to MNIST, but with four times more instances (280,000 instead of 70,000).

(ii) *Letters*: this dataset contains only letters, and a distinction between uppercase and lowercase is not made. As a result, the dataset contains 26 classes and a total of 145,600 samples.

To the best of our knowledge, the EMNIST dataset is so new that there are not published works using it as a benchmark. The original EMNIST paper by Cohen et al. [43] includes a baseline using a linear classifier and OPIUM (Online Pseudo-Inverse Update Method), a classifier introduced by van Schaik and Tapson [46]. It is worth noting that the performance in the original EMNIST paper is reported in terms of accuracy instead of error rate. For this reason, in this section, we will use this metric for reporting the performance of each work.

More recently, Peng and Yin [47] have used Markov random field-based CNN achieving an accuracy of 95.44% in the letters dataset and of 99.75% in the digits dataset. Also, Singh et al. [48] reported an accuracy of 99.62% in EMNIST Digits, using a CNN with three convolutional layers and two fully connected layers. In EDEN [25], authors also tested their neuroevolution approach against the EMNIST Digits test set, obtaining an accuracy of 99.3%. The dataset has also been used by Netftci et al. [49] for testing the performance of event-driven random backpropagation when training neuromorphic deep neural networks, although they have combined letters and digits data for classification, and therefore its performance cannot be compared with the results obtained in this work; and by Shu et al. [50] albeit with the purpose of pairwise classification.

Despite the fact of EMNIST not being used so far in other published works, some researchers have used NIST Special Database 19 in the past. It should be noted that these works are not directly comparable, because the database is not identical; however, results can be extrapolated. For example, Milgram et al. [51] reported an accuracy of 98.75% using SVMs with sigmoid function, Granger et al. [52] attained an accuracy of 96.49% using particle swarm optimization to evolve the topology of neural networks, and Oliveira et al. [53] reported an accuracy of 98.39% with a multi-layer perceptron.

Other authors have used also letter from NIST Special Database 19. For example, Radtke et al. [54] used record-to-record travel (accuracy of 96.53% for digits and 93.78% for letters), Koerich and Kalva [55] tested a multi-layer perceptron only with the letters dataset (accuracy of 87.79%.), and Cavalin et al. [56] used hidden Markov models (accuracy of 98% for digits and up to 90% for letters, though this result is only using uppercase letters, and the accuracy decreases to 87% when lowercase letters are also considered).

Finally, to the best of our knowledge, Cireşan et al. [35] are the only authors to have used this database for testing the performance of committees of CNNs, attaining accuracies of 88.12% for the whole database, 92.42% for letters, and 99.19% for digits.

A summary of the reviewed works is shown in Table 3. The upper side of the table shows the performance of classical machine learning models and non-convolutional neural networks, whereas the lower side shows those works involving CNNs. Best results are boldfaced.

4.2. Transfer of Evolved CNN Topologies. In order to evaluate the performance of the neural models obtained in the MNIST

Table 3: Side-by-side comparison of the results for the EMNIST dataset along with the reported accuracy, including works using similar datasets from NIST Special Database 19.

Technique	Letters	Digits
Linear classifier [43]	55.78%	84.70%
OPIUM [43]	85.15%	95.90%
SVMs (one against all + sigmoid) [51]	–	98.75%
Multi-layer perceptron [53]	–	98.39%
Hidden Markov model [56]	90.00%	98.00%
Record-to-record travel [54]	93.78%	96.53%
PSO + fuzzy ARTMAP NNs [52]	–	96.49%
Multi-layer perceptron [55]	87.79%	–
Markov random field CNN [47]	**95.44%**	**99.75%**
Parallelized CNN [48]	–	99.62%
EDEN [25]	–	99.30%
Committee of 7 CNNs [35]	92.42%	99.19%

domain, when directly transferred to the EMNIST domain, the 20 best topologies, obtained by neuroevolution, are selected and then transferred into the new domain where they will follow a standard learning procedure to check their performance. We have performed this task for both evolutionary algorithms (GA and GE) and for both the letters and digit domains. However, in this paper, we will report only the results obtained by GE, being similar and slightly better than those using GA, for clarity and efficiency reasons.

After full training of the neuroevolved topologies, a statistical summary of the accuracies for each architecture is shown in Table 4 for the Letters dataset and Table 5 for the Digits dataset, showing the mean, median, standard deviation, and maximum and minimum values. It should be recalled that the results shown correspond to the accuracies obtained, over the test set, by the networks transferred from the MNIST domain to the Letters and Digits domains. We have reported the performance in terms of accuracy instead of error for consistency with most works in the state of the art.

It can be seen that first individuals do not behave better than the rest, pointing out that these topologies are not explicitly optimized for the EMNIST dataset. However, results are very good: both in the Digits and in the Letters datasets; the maximum accuracies obtained are over 99.7% and 95%, respectively. If we compare the results obtained with our evolutionary system (refer to Tables 5 and 4) with those of the state of the art (shown in Table 3), the enormous efficacy of the evolved models can be appreciated, even when the neuroevolution was carried out for a different (yet similar) domain. In the Letters domain, the accuracy of our approach (95.19% for Letters and 99.73% for Digits) is only outperformed by the work by Peng and Yin [47] (95.44% and 99.75%, respectively).

Moreover, the distributions of accuracies for each topology after full training with the EMNIST training set are depicted in Figure 5 for the Letters domain and Figure 6 for the Digits domain. It is interesting to realize that results are very homogeneous for most individuals, and variance is very small in all cases. This points out the robustness of the method: it is easy to obtain competitive models even in one or few executions of the neuroevolution and full training phases. The variance is higher in the Letters dataset, since the domain is more complicated and the number of classes is larger (26 against 10), although the same conclusions apply to a lesser extent.

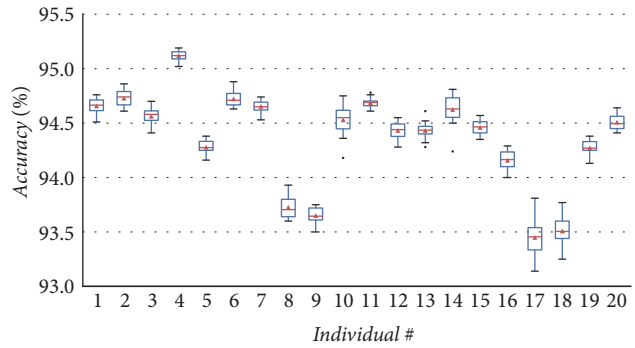

Figure 5: Boxplot showing the distribution of accuracies of the best 20 GE individuals after full training in the EMNIST Letters dataset.

It is also worth realizing that performance is consistent across both datasets. When a topology behaves especially well in one domain, it has a particular good behavior in the other as well. For example, in the Letters domain, the three best models are 4, 6, and 2, and so are in the Digits domain, too.

4.3. Topology Transfer with Committees of Evolved CNNs. Additionally, we find it interesting to explore how both mechanisms introduced in this paper perform when combined, i.e., whether committees for the EMNIST database can be built and obtain successful results given models whose topologies were designed for the MNIST dataset.

Figure 7 shows the accuracy evolution for the Letters dataset as new models are included. The best result in EMNIST Letters is an accuracy of 95.35% (error rate of 4.65%), with 10 CNNs, with an ensemble involving 20 CNNs.

Table 4: Summary of accuracies of the best 20 GE individuals after full training in EMNIST Letters.

#	Mean	Std. Dev.	Min.	Median	Max.
1	94.6585	0.074004	94.665	94.51	94.76
2	94.7300	0.070934	94.740	94.61	94.86
3	94.5635	0.084870	94.580	94.41	94.70
4	95.1215	0.049553	95.120	95.02	95.19
5	94.2790	0.064880	94.275	94.16	94.38
6	94.7230	0.068832	94.710	94.63	94.88
7	94.6540	0.053646	94.650	94.53	94.74
8	93.7270	0.094429	93.705	93.60	93.93
9	93.6515	0.070058	93.645	93.50	93.75
10	94.5305	0.139075	94.550	94.18	94.75
11	94.6890	0.045410	94.690	94.61	94.78
12	94.4335	0.071545	94.440	94.28	94.55
13	94.4330	0.075888	94.435	94.28	94.61
14	94.6260	0.132045	94.630	94.24	94.81
15	94.4605	0.062616	94.465	94.35	94.57
16	94.1590	0.093578	94.165	94.00	94.29
17	93.4505	0.156423	93.455	93.14	93.81
18	93.5095	0.149929	93.505	93.25	93.77
19	94.2725	0.074684	94.270	94.13	94.38
20	94.5085	0.070208	94.495	94.41	94.64

Table 5: Summary of accuracies of the best 20 GE individuals after full training in EMNIST Digits.

#	Mean	Std. Dev.	Min.	Median	Max.
1	99.6835	0.009881	99.680	99.66	99.70
2	99.6880	0.015079	99.685	99.66	99.72
3	99.6155	0.018489	99.610	99.58	99.65
4	99.7145	0.009987	99.720	99.70	99.73
5	99.5420	0.029487	99.540	99.48	99.60
6	99.6780	0.019628	99.680	99.64	99.72
7	99.6765	0.015652	99.680	99.64	99.70
8	99.5410	0.024900	99.540	99.49	99.59
9	99.4655	0.039533	99.475	99.39	99.52
10	99.6570	0.016575	99.650	99.63	99.69
11	99.6785	0.013485	99.675	99.66	99.72
12	99.6115	0.016944	99.610	99.58	99.64
13	99.6665	0.016944	99.665	99.63	99.70
14	99.6205	0.022589	99.620	99.58	99.66
15	99.6560	0.011425	99.660	99.64	99.68
16	99.6305	0.013945	99.630	99.60	99.65
17	99.3050	0.070599	99.320	99.16	99.39
18	99.5220	0.031722	99.525	99.44	99.59
19	99.6480	0.015079	99.650	99.62	99.68
20	99.6015	0.015313	99.600	99.58	99.63

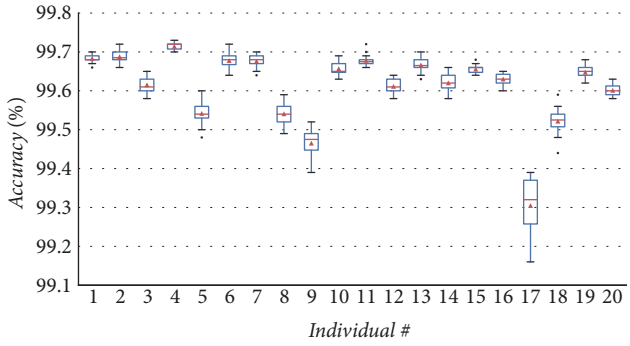

Figure 6: Boxplot showing the distribution of accuracies of the best 20 GE individuals after full training in the EMNIST Digits dataset.

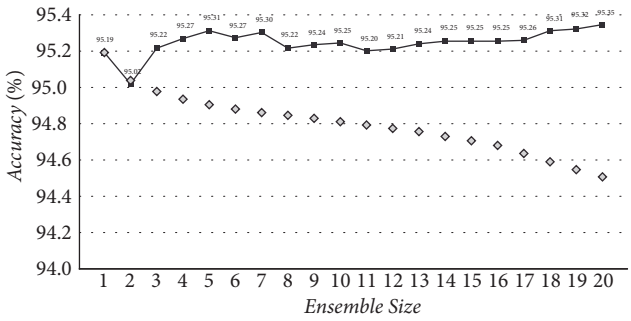

Figure 7: Accuracy of CNNs committees with up to 20 models from GE with EMNIST Letters.

The accuracy looks very steady when more than three CNNs are involved, yet results improve at the end, attaining better accuracy with a larger number of models.

The plot referred to the Digits dataset is shown in Figure 8. It is noticeable that, beyond 8 individuals, the accuracy stabilizes around 99.75% (error rate of 0.25%). The best accuracy is 99.7725% (error rate of 0.2275%), with a committee of six CNNs. The accuracy is very steady as new models are added to the ensemble, with the worst result of an ensemble of at least three CNNs obtaining only 0.03 percentage points less than the aforementioned result.

Given these results, we can conclude that once again the use of committees improves the results over those attained with single models. In particular, the using of committees in the Letters dataset increases the accuracy from 95.19% (using a single model) to 95.35%, reducing the gap with the result reported by Peng and Yin [47] (95.44%). Regarding the Digits dataset, the accuracy raises from 99.73% to 99.7725%, a result that would head the ranking with this dataset.

Regarding the interpretation of the results, Figure 9 shows the confusion matrix for the EMNIST Letters dataset using the best ensemble found, which was obtained using the topologies optimized with GE. It can be seen that accuracy is almost perfect. The most common mistakes involve mixing up the letters 'I' and 'L', the letters 'G' and 'Q', and to a much lesser extent the letters 'V' and 'U'. These seem like acceptable mistakes given the high similarity of these characters. Figure 10 shows a random sample of 100 misclassified images in the EMNIST Letters dataset. As we already knew from the confusion matrix, most misclassified samples are vertical bars which could be either an L' or an 'I' (notice that both are even more similar when comparing a lowercase 'l' with an uppercase 'I'). Also, it can be seen how some characters are hardly recognizable even by a human.

As for the confusion matrix for the Digits dataset using the best ensemble found, it is depicted in Figure 11. Again, most values are in the main diagonal, representing an almost perfect accuracy. Most remarkable mistakes involve misclassifying digits '9' and '4', '3', and '5', and '2' and '3'. To a lesser extent, the ensemble also mixes up the digits '6' and '0'. From a test set of 40,000 samples, only 97 were incorrectly classified. In fact, only 91 instances from a total of 40,000 in the test set have been incorrectly classified, and these can be seen in Figure 12. Most of these digits are hardly recognizable. Actually, one instance seems to involve two digits in one (seventh row, eighth column). Others seem to be incomplete, and it is hard to tell whether they are a '5' or a '3' (e.g., first row, third column). Finally, the confusion between '4's and '9's seems to arise because either a digit '4' is very rounded on the top or the digit '9' seems to be slightly open, maybe due to the image being incomplete.

5. Conclusions

Convolutional neural networks are a very effective tool in numerous complex problems and in particular in classification tasks. The only drawback of those systems is the dependence between network models, understood as the union of architecture and parameters, and the results they are able to accomplish. This dependence makes it difficult and expensive to find the optimal model for each problem. One way to avoid this difficulty is by the use of evolutionary systems to automatically find appropriate architectures in each case, an approach called neuroevolution which has been used with success since the late 1980s but only in recent years have been applied to deep learning models.

In this work, we have focused on analyzing the potential of a neuroevolutionary system regarding two different lines of study: on one hand to improve the performance of the generated models, by means of the use of committees, and on the other hand to validate the robustness of such models to be transferred to new, similar, domains.

The use of committees exploits the property of evolutionary systems to generate a population of individuals, instead of working with a single solution. The use of multiple models allows the outcomes to be modulated, in such a way that possible errors of individual systems can be corrected, provided that the models that conform to the ensemble are different and effective enough, as not to distort the result of the best model. In this work, this effect is achieved through the inclusion of niching strategies in the evolutionary system, as well as the use of a historical set, hall-of-fame, of the best models.

Experiments have been carried out for different sizes of ensembles, from 2 up to a maximum of 20, for the handwritten classification task. Results prove an improvement with respect to the isolated models, in all cases. In

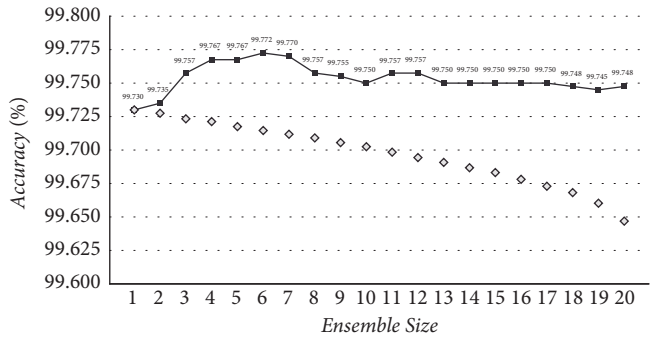

Figure 8: Accuracy of CNNs committees with up to 20 models from GE with EMNIST Digits.

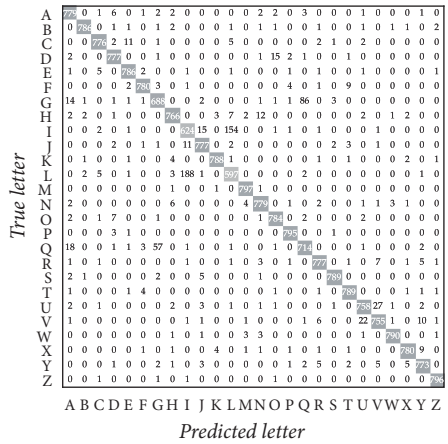

Figure 9: Confusion matrix of the best committee found for EMNIST Letters.

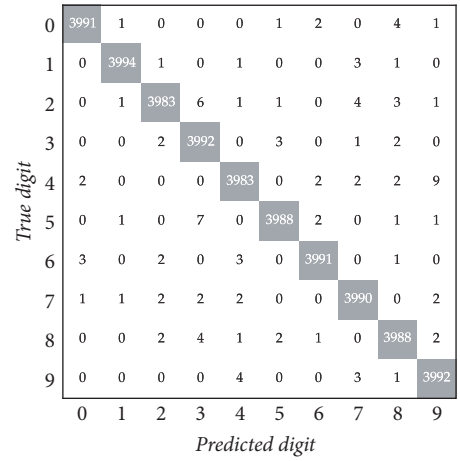

Figure 11: Confusion matrix of the best committee found for EMNIST Digits.

Figure 10: Sample of 100 misclassified images in the EMNIST Letters test set with the best committee found.

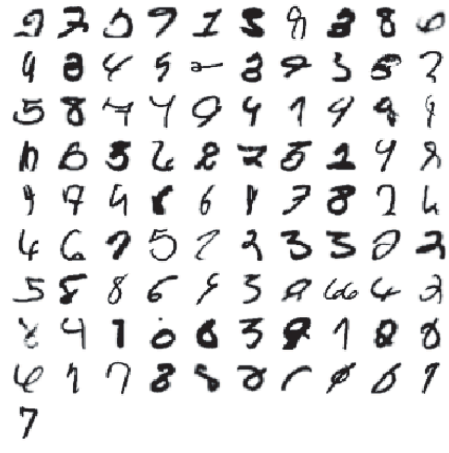

Figure 12: The 91 misclassified images in the EMNIST Digits test set with the best committee found.

addition, a more detailed analysis shows higher performance for small ensemble sizes, around seven models, from which the improvements are less important. The results also show an insignificant dependency on the size of the ensembles, from a size of four, which allows us to conclude that it is not necessary the use of large ensembles, above a critical minimum size.

Regarding topology transfer learning, our hypothesis was that once suitable topologies were found using neuroevolution, these topologies should behave reasonably well over a different domain or problem which is similar to the one used for evolving the population. If the hypothesis is correct, then a lot of time could be saved, since neuroevolution

is an computational expensive process. In our work, the neuroevolutionary process searched for optimal topologies for the MNIST database, and then we have transferred those topologies to EMNIST. EMNIST has been released recently and provides several databases: we have focused on EMNIST Digits, which is the same problem as MNIST but with data obtained from a different source, and EMNIST Letters, which is a similar yet different problem involving recognition of handwritten letters. Results have shown that transferred topologies are able to obtain a very high performance, attaining an accuracy of 99.73% in EMNIST Digits and 95.19% in EMNIST Letters.

These results show the great robustness of the models generated by our evolutionary system. Not only they achieve the best results reported so far in these domains, but also 18 out of the 20 models obtained outperform the best state-of-the-art results in the Digits domain, and so do the whole 20 in the Letters domain. These results are even more significant if we take into account that one of the referred works also uses CNNs, in an ensemble of 7 models. This confirms not only the difficulty of finding effective models by hand, but also that even models learned by our system in some domains, when applied directly to different, but similar, domains, obtain better results than those designed by experts focused on the latter. Of course, this happens without disregarding that, in the future, new works can unveil models better adapted to that domains, thus resulting in a better performance.

Additionally, we have realized that these two improvements are not exclusive, and when combined, the results can be improved, outperforming single models, reaching accuracies of 99.7725% for EMNIST Digits and 95.35% for EMNIST Letters. This translates into an improvement of 0.0425 percentage points for EMNIST Digits and 0.16 percentage points for EMNIST Letters when compared to the best accuracy obtained with individual models.

Based on all the tests and analyses, the proposed approaches are recommended even if the process is time-consuming, since it is fully automated and the output can be used for building committees or to be applied to different, yet similar problems. Results empirically prove how successful both approaches are and support some of the benefits of using neuroevolution for determining the best topologies and hyperparameters of CNNs.

Data Availability

The databases used in this paper are publicly available for download and, in particular, can be accessed from the following website: http://yann.lecun.com/exdb/mnist/, whereas EMNIST can be downloaded from the following site: https://www.nist.gov/itl/iad/image-group/emnist-dataset.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is partially supported by the Spanish Ministry of Education, Culture and Sport under FPU fellowship with identifier FPU13/03917.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16)*, pp. 770–778, Las Vegas, Nev, USA, June 2016.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 3431–3440, IEEE, Boston, Mass, USA, June 2015.

[3] A. Baldominos, Y. Saez, and P. Isasi, "Evolutionary convolutional neural networks: an application to handwriting recognition," *Neurocomputing*, vol. 283, pp. 38–52, 2018.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[5] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Network*, pp. 255–258, MIT Press, 1998.

[6] J. Bergstra, O. Breuleux, F. Bastien et al., "Theano: a CPU and GPU math compiler in Python," in *Proceedings of the 9th Python in Science Conference*, 2010.

[7] Y. Jia, E. Shelhamer, J. Donahue et al., "Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the ACM Conference on Multimedia (MM '14)*, pp. 675–678, ACM, Orlando, Fla, USA, November 2014.

[8] M. Abadi, P. Barham, J. Chen et al., "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, pp. 265–283, 2016.

[9] Lasagne, "Welcome to lasagne last visited on," 2017, http://lasagne.readthedocs.io.

[10] Keras, "Keras: the python deep learning library," 2017, https://keras.io.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[12] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[13] J. Koutník, J. Schmidhuber, and F. Gomez, "Evolving deep unsupervised convolutional networks for vision-based reinforcement learning," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 541–548, Canada, July 2014.

[14] P. Verbancsics and J. Harguess, "Image classification using generative neuroevolution for deep learning," in *Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 488–493, USA, January 2015.

[15] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15, no. 2, pp. 185–212, 2009.

[16] S. R. Young, D. C. Rose, T. P. Karnowski, S. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through

an evolutionary algorithm," in *Workshop on Machine Learning in High-Performance Computing Environments*, 2015.

[17] S. R. Young, D. C. Rose, T. Johnston et al., "Evolving deep networks using HPC," in *Proceedings of the Machine Learning on HPC Environments*, pp. 3924–3928, Denver, CO, USA, November 2017.

[18] I. Loshchilov and F. Hutter, "CMA-ES for hyperparameter optimization of deep neural networks," 2016, https://arxiv.org/abs/1604.07269.

[19] C. Fernando, D. Banarse, M. Reynolds et al., "Convolution by evolution: differentiable pattern producing networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 109–116, ACM, New York, NY, USA, 2016.

[20] L. Xie and A. Yuille, "Genetic CNN," in *Proceedings of the 16th IEEE International Conference on Computer Vision, (ICCV '17)*, October 2017.

[21] R. Miikkulainen, J. Liang, E. Meyerson et al., "Evolving deep neural networks," 2017, https://arxiv.org/abs/1703.00548.

[22] T. Desell, "Large scale evolution of convolutional neural networks using volunteer computing," in *Proceedings of the the Genetic and Evolutionary Computation Conference Companion*, pp. 127-128, Berlin, Germany, July 2017.

[23] E. Real, S. Moore, A. Selle et al., "Large-scale evolution of image classifiers," in *Proceedings of Machine Learning Research*, vol. 70, pp. 2902–2911, 2017.

[24] Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks for image classification," 2017, https://arxiv.org/abs/1710.10741.

[25] E. Dufourq and B. A. Bassett, "EDEN: evolutionary deep networks for efficient machine learning," in *Proceedings of the Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, pp. 110–115, Bloemfontein, South Africa, November 2017.

[26] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO '17)*, pp. 497–504, Germany, July 2017.

[27] J. Davison, "DEvol: Automated deep neural network design via genetic programming," 2017, https://github.com/joeddav/devol.

[28] E. Bochinski, T. Senst, and T. Sikora, "Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms," in *Proceedings of the 24th IEEE International Conference on Image Processing, (ICIP '17)*, pp. 3924–3928, Beijing, China, September 2017.

[29] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, British Columbia, Canada, 2018.

[30] O. Kramer, "Evolution of convolutional highway networks," in *Applications of Evolutionary Computation*, K. Sim and P. Kaufmann, Eds., vol. 10784 of *Lecture Notes in Computer Science*, pp. 395–404, Springer International Publishing, Cham, Switzerland, 2018.

[31] J. Prellberg and O. Kramer, "Lamarckian evolution of convolutional neural networks," 2018, https://arxiv.org/abs/1806.08099.

[32] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "DENSER: deep evolutionary network structured representation," *Genetic Programming and Evolvable Machines*, 2018.

[33] B. Wang, Y. Sun, B. Xue, and M. Zhang, "A hybrid DE approach to designing CNN for image classification," in *Proceedings of the 31st Australasian Joint Conference on Artificial Intelligence*, 2018.

[34] Y. Sun, B. Xue, and M. Zhang, "Automatically evolving cnn architectures based on blocks," 2018, https://arxiv.org/abs/1810.11875.

[35] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification," in *Proceedings of the 11th International Conference on Document Analysis and Recognition*, pp. 1135–1139, September 2011.

[36] G. Pons and D. Masip, "Supervised committee of convolutional neural networks in automated facial expression analysis," *IEEE Transactions on Affective Computing*, vol. 9, no. 3, pp. 343–350, 2018.

[37] C. Schaefer, M. Geiger, T. Kuntzer, and J.-P. Kneib, "Deep convolutional neural networks as strong gravitational lens detectors," *Astronomy & Astrophysics* , vol. 611, 2018.

[38] Y. Kawana, N. Ukita, J.-B. Huang, and M.-H. Yang, "Ensemble convolutional neural networks for pose estimation," *Computer Vision and Image Understanding*, vol. 169, pp. 62–74, 2018.

[39] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, "An ensemble of fine-tuned convolutional neural networks for medical image classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 31–40, 2017.

[40] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[41] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[42] J. R. Chang and Y. S. Chen, "Batch-normalized maxout network in network," *Journal of Machine Learning Research*, vol. 48, 2015.

[43] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," 2017, https://arxiv.org/abs/1702.05373.

[44] NIST, "NIST Special Database 19," 2017, https://www.nist.gov/srd/nist-special-database-19.

[45] P. J. Grother and K. K. Hanaoka, "NIST special database 19 handprinted forms and characters database," Tech. Rep., National Institute of Standards and Technology, 2016.

[46] A. van Schaik and J. Tapson, "Online and adaptive pseudoinverse solutions for ELM weights," *Neurocomputing*, vol. 149, pp. 233–238, 2015.

[47] Y. Peng and H. Yin, "Markov random field based convolutional neuralx networks for image classification," in *IDEAL 2017: Intelligent Data Engineering and Automated Learning*, H. Yin, Y. Gao, S. Chen et al., Eds., vol. 10585 of *Lecture Notes in Computer Science*, pp. 387–396, Springer, Guilin, China, 2017.

[48] S. Singh, A. Paul, and M. Arun, "Parallelization of digit recognition system using deep convolutional neural network on CUDA," in *Proceedings of the Third International Conference on Sensing, Signal Processing and Security (ICSSS '17)*, pp. 379–383, Chennai, India, May 2017.

[49] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: enabling neuromorphic deep learning machines," *Frontiers in Neuroscience*, vol. 11, article no 324, 2017.

[50] L. Shu, H. Xu, and B. Liu, "Unseen class discovery in open-world classification," 2018, https://arxiv.org/abs/1801.05609.

[51] J. Milgram, M. Cheriet, and R. Sabourin, "Estimating accurate multi-class probabilities with support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, (IJCNN '05)*, pp. 1906–1911, Canada, August 2005.

[52] E. Granger, P. Henniges, R. Sabourin, and L. Oliveira, "Supervised learning of fuzzy ARTMAP neural networks through particle swarm optimisation," *Journal of Pattern Recognition Research*, vol. 2, no. 1, pp. 27–60, 2007.

[53] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Automatic recognition of handwritten numerical strings: a recognition and verification strategy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1438–1454, 2002.

[54] P. V. W. Radtke, R. Sabourin, and T. Wong, "Using the RRT algorithm to optimize classification systems for handwritten digits and letters," in *Proceedings of the 23rd Annual ACM Symposium on Applied Computing, (SAC '08)*, pp. 1748–1752, Brazil, March 2008.

[55] A. L. Koerich and P. R. Kalva, "Unconstrained handwritten character recognition using metaclasses of characters," in *Proceedings of the IEEE International Conference on Image Processing, (ICIP '05)*, pp. 542–545, Italy, September 2005.

[56] P. R. Cavalin, A. De Souza Britto Jr., F. Bortolozzi, R. Sabourin, and L. E. S. Oliveira, "An implicit segmentation-based method for recognition of handwritten strings of characters," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, pp. 836–840, France, April 2006.