EvoPruneDeepTL: An Evolutionary Pruning Model for Transfer Learning based Deep Neural Networks

Javier Poyatos[a], Daniel Molina[a,*], Aritz D. Martinez[b], Javier Del Ser[b,c], Francisco Herrera[a,d]

[a]Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, Granada, 18071, Spain
[b]TECNALIA, Basque Research & Technology Alliance (BRTA), Derio, 48160, Spain
[c]University of the Basque Country (UPV/EHU), Bilbao, 48013, Spain
[d]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia

Abstract

In recent years, Deep Learning models have shown a great performance in complex optimization problems. They generally require large training datasets, which is a limitation in most practical cases. Transfer learning allows importing the first layers of a pre-trained architecture and connecting them to fully-connected layers to adapt them to a new problem. Consequently, the configuration of the these layers becomes crucial for the performance of the model. Unfortunately, the optimization of these models is usually a computationally demanding task. One strategy to optimize Deep Learning models is the pruning scheme. Pruning methods are focused on reducing the complexity of the network, assuming an expected performance penalty of the model once pruned. However, the pruning could potentially be used to improve the performance, using an optimization algorithm to identify and eventually remove unnecessary connections among neurons. This work proposes EvoPruneDeepTL, an evolutionary pruning model for Transfer Learning based Deep Neural Networks which replaces the last fully-connected layers with sparse layers optimized by a genetic algorithm. Depending on its solution encoding strategy, our proposed model can either perform optimized pruning or feature selection over the densely connected part of the neural network. We carry out different experiments with several datasets to assess the benefits of our proposal. Results show the contribution of EvoPruneDeepTL and feature selection to the overall computational efficiency of the network as a result of the optimization process. In particular, the accuracy is improved reducing at the same time the number of active neurons in the final layers.

Keywords: Deep Learning, Evolutionary Algorithms, Pruning, Feature Selection, Transfer Learning

1. Introduction

Deep Learning (DL) (Goodfellow et al., 2016) is one of the most attractive research areas in machine learning in recent times, due to the great results offered by such models in a plethora of applications. DL architectures are successfully used in many different problems, like audio classification (Lee et al., 2009), audio recognition (Noda et al., 2015), object detection (Zhou et al., 2017), image classification for medical analysis (Muhammad et al., 2021) or vehicular perception (Muhammad et al., 2020).

Convolutional Neural Networks (CNNs) (Lecun et al., 1998) constitute the state-of-the art in image classification (Sultana et al., 2018). CNNs include two parts, the first part is actually a feature extractor based on convolution and pooling operations. The second part usually contains one or more fully connected

*Corresponding author

Email addresses: jpoyatosamador@ugr.es (Javier Poyatos), dmolina@decsai.ugr.es (Daniel Molina), aritz.martinez@tecnalia.com (Aritz D. Martinez), javier.delser@tecnalia.com (Javier Del Ser), herrera@decsai.ugr.es (Francisco Herrera)

layers. In these fully-connected layers, the neuron of each layer is connected to all the neurons of the previous layer, which generates a large number of weights to be trained. The design of an appropriate network for each problem is a requirement in order to obtain a good performance. The training process of a DL architecture is frequently time-consuming. Complexity reduction maintaining the performance is an important challenge in DL, currently attracting significant attention in the community. Transfer Learning (TL) (Weiss et al., 2016) is usually considered the alternative. It is very common to use DL model with fixed and pre-trained weights in the convolutional layers with a dataset (like ImageNet (Krizhevsky et al., 2012)) and then add and train several layers, named fully-connected layers, to adapt the network to a different classification problem (Shin et al., 2016; Khan et al., 2019; Gómez-Ríos et al., 2019).

The architecture of fully-connected layers used for the problem is a critical decision and its design is still an open issue in terms of the number of layers and neurons per layer (Liu et al., 2017b). There are general guidelines based on the experience working with these layers rather than rules to follow for the configuration of them. Therefore, any kind of optimization in them could provide a benefit in terms of model complexity and performance. The pruning approaches follow the key idea of reducing the complexity of the model, which creates new networks with less computational cost for training. This idea is followed in (Frankle and Carbin, 2019), which also shows that, in the end, the accuracy can also improve as a result of pruning.

Pruning is interpreted as removing unnecessary connections from the model, but learning which connections are the fittest to improve the performance of the model is the key point. In fact, the selection of the best *features* for the problem is known as Feature Selection (FS) (Iguyon and Elisseeff, 2003). In our case, TL allows the extraction of the features of the input data of the DL model. These features are the input of the fully-connected layers that will be trained and, as a result of that, will largely affect the performance of the network. Nonetheless, in many cases, the problem that is formulated to learn these features is usually different, sometimes more complex, than the one at hand and, therefore, not all the learned patterns would be required. For that reason, FS gives rise to an interesting option to select and retain the subset of all features that lead to an improved performance of the model (Yildirim et al., 2018).

In pruning scenarios, the main aim of most of the traditional pruning techniques mainly aim at reducing the number of trainable parameters of the network, at the cost of a lower performance. They seek to control the performance degradation resulting from the process, but it is not their priority. Furthermore, they locally optimize parts of the network rather than searching for globally optimal pruning policies, yielding usually subobtimal pruned subnetworks with a lower performance. Another disadvantage of these pruning proposals is the fact that, as the pruning affect all layers, the complete network must be trained again, hence obtaining no advantages from the TL process. It could be useful to have a pruning technique that prioritizes results over complexity reduction, targeting a global performance improvement of the network while reducing its complexity.

Transforming the fully-connected layers into a sparse representation, in which each connection could be active or inactive, could be used to prune neural networks. Following this approach, both pruning and FS can be seen as optimization problems, in which the target is to obtain the active set of connections that produce the best performance. This optimization problem can be globally tackled by optimization algorithms like Evolutionary Algorithms (Bäck et al., 1997) (EAs). They have been successfully applied to many complex optimization problems. Even though they cannot guarantee the achievement of the optimum for the problem at hand, they obtain good results with limited resources and reasonable processing time. Another advantage is their versatility: several of them, like genetic algorithms (Goldberg, 1989) (GAs) allow optimizing solutions with different representations (Chambers, 2000). The spectrum of problems in which EAs can be used is very wide. EAs have been traditionally applied to optimize neural networks (Iba, 2018), but their usage in DL networks to improve DL networks (Martinez et al., 2021), to train them (Mohapatra et al., 2021), and to create new DL networks from scratch (Elsken et al., 2019b) is more recent. The use of EA's is mainly oriented towards optimizing a complete network. However, in this paper, our aim is to adapt the fully-connected layers (the only trained for the problem to solve using TL) to improve the accuracy in the predictions, together with the complexity reduction. Our main hypothesis is the convenience of use of EAs to prune the fully-connected layers via a sparse representation.

We propose an evolutionary pruning model based on TL for deep neural networks, Evolutionary Pruning for Deep Transfer Learning (EvoPruneDeepTL). EvoPruneDeepTL can be applied to a DL model that resorts to TL to tackle a new task. EvoPruneDeepTL combines sparse layers and EA, consequently, neurons in such layers are pruned to adapt their sparsity pattern to the addressed problem. EvoPruneDeepTL is able to efficiently explore the neuron search space (to discover coarsely grained solutions) or, alternatively, in the connection search domain (fine-grained solutions). An important aspect to analyze in EvoPruneDeepTL is that one of its solution encoding schemes effectively leads to a feature selection mechanism, in which we deactivate the extracted features and the EA evolves these features to learn which ones fit best as predictors for the given problem.

To assess the performance of EvoPruneDeepTL, we have conducted an extensive experimentation that leads to several valuable insights. To begin with, experimental results showcase the behavior and effectiveness of EvoPruneDeepTL in terms of precision and in terms of reduction of the complexity of the network. Thanks to the flexibility of EvoPruneDeepTL, it is applied to perform either pruning or FS. Both cases improve the accuracy of the network when the comparison is made against reference models and CNN pruning methods from the literature. Moreover, in most cases, the FS scheme achieves a better performance than the pruning scheme in terms of the accuracy of the network. Furthermore, the network pruned by the FS scheme also achieves a significantly reduced number of connections in its fully connected part, contributing to the computational efficiency of the network. In short, this extensive experimentation is used to provide answer to the following four questions as the thread running through this experimental study:

(RQ1) which is the performance of EvoPruneDeepTL against fully-connected models?

(RQ2) which would be better, to remove neurons or connections?

(RQ3) which is the performance of EvoPruneDeepTL when compared to other efficient pruning methods?

(RQ4) which would be better, the use of pruning of fully-connected layers or Feature Selection?

The rest of the article is structured as follows: Section 2 exposes related work to our proposal present in the literature. Section 3 shows the details of the proposed EvoPruneDeepTL model. Section 4 presents our experimental framework. In Section 5, we show and discuss the EvoPruneDeepTL's results of the experiments of pruning, feature selection and against efficient CNN pruning methods of the literature. Finally, Section 6 draws the main conclusions stemming from our work, and outlines future work departing from our findings.

2. Related work

The purpose of this section is to make a brief review of contributions to the literature that link to the key elements of our study: Transfer Learning (Subsection 2.1), Neural Architecture Search (Subsection 2.2), CNN pruning (Subsection 2.3), Evolutionary Algorithms (Subsection 2.4) and Feature Selection with Deep Learning (Subsection 2.5).

2.1. Transfer Learning

TL (Pan and Yang, 2010) is a DL mechanism encompassing a broad family of techniques (Tan et al., 2018). Arguably, the most straightforward method when dealing with neural networks is *Network-based deep transfer learning*, in which a previous network structure with pre-trained parameters in a similar problem is used. It offers good results by the behavior of DL models, in which first layers detect useful features on the images, and later layers strongly depend on the chosen dataset and task. As finding these standard features on the first layers seems very common regardless of the natural image datasets, its trained values can be used for different problems (Yosinski et al., 2014). Training DL models from scratch is usually time-consuming due to the great amount of data in most cases. TL gives some benefits which make it a good option for DL: reduction of time needed for training (Sa et al., 2016), better performance of the model and less need of data.

TL has been applied to several real-world applications, such as sound detection (Jung et al., 2019) or coral reef classification (Gómez-Ríos et al., 2019). Moreover, in (Tajbakhsh et al., 2016) two different approaches for TL are discussed: fine-tuning or full training. They demonstrated that, for medical reasons, a pre-trained CNN with adequate fine-tuning performed better in terms of accuracy than a CNN trained from scratch. Another approach of TL is presented in (Mehdipour Ghazi et al., 2017), in which an optimization of TL parameters for plant identification is proposed.

There are many different deep neural networks proposed in the literature. One of the most popular is ResNet, which uses residual learning to improve the training process, obtaining better performance than other models (He et al., 2016). ResNet models are characterized by the use of deeper neural networks without loss of information due to their architecture. Different ResNet models with TL have been used in several applications (Scott et al., 2017), such as medical classification like pulmonary nodule (Nibali et al., 2017) and diabetic retinopathy classification(Wan et al., 2018). Due to its applications and the benefits of its usage, ResNet is the neural network model used in this study.

2.2. Neural Architecture Search

The appropriate design of a neural network is a key point to solve DL problems. Nevertheless, finding the best architecture that optimally fits the data and, as a result of that, gives the best outcome for the problem is extremely difficult. Recently, the term Neural Architecture Search, NAS, has obtained a great importance in this field. The objective of NAS is the automatic search for the best design of a NN to solve the problem at hand.

The first work in this field is presented in (Stanley and Miikkulainen, 2002) in the beginning of this century, in which they demonstrate the effectiveness of a GA to evolve topologies of NN.

In (Zoph et al., 2018), the authors design the NASNet architecture, a new search space to look for the best topology for the tackled problem. Moreover, in (Liu et al., 2018a) the authors propose to search for structures in increasing order of their level of complexity, while learning a surrogate model to guide the search through structure space.

NAS methods usually rely on Reinforcement Learning, RL, and EA, like (Zoph et al., 2018) or (Liu et al., 2018b), in which the authors explore the search space using a hierarchical genetic representation. Another example of RL for NAS is shown in (Kokiopoulou et al., 2020). The authors propose a novel method that, by sharing information on multiple tasks, is able to efficiently search for architectures.

NAS can also be viewed as a multi-objective problem. Among these methods, one of them is presented in (Elsken et al., 2019a), in which the authors propose a multi-objective for NAS that allows approximating the entire Pareto-front of architectures. Another example is Neural Architecture Transfer (Lu et al., 2021), that allows to overcome a common limitation of NAS, that is requiring one complete search for each deployment specification of hardware or objective. They use an integrated online transfer learning and a many-objective evolutionary search procedure.

Recently, one of the most well-known multi-objective EA, NSGA-II, has been used for NAS (Lu et al., 2019), called NSGA-Net. This novel proposal looks for the best architecture through a three-step search: an initialization step from hand-crafted architectures, an exploration step that performs the EA operators to create new architectures, and finally an exploitation step that utilizes the knowledge stored in the history of all the evaluated architectures in the form of a Bayesian Network.

Lastly, there are more advanced techniques of NAS and EA given by (Real et al., 2019), in which a new model for evolving a classifier is presented, and by (Real et al., 2020), in which the authors propose AutoML-Zero, an evolutionary search to build a model from scratch (with low-level primitives for feature combination and neuron training) which is able to get a great performance over the addressed problem.

2.3. CNN Pruning

The main reason to optimize the architecture of a deep neural network is to reduce its complexity. That reduction can be done in different ways (Long et al., 2019a). One of them is by designing compact models from scratch instead of resorting to architectures comprising multiple layers. Another strategy is via weights-sharing (Ullrich et al., 2017). An alternative method to reduce the complexity of DL models is low-rank factorization (Long et al., 2019b), based on a matrix decomposition to convolutional layers to estimate parameters. However, one of the most popular is Network Pruning. The objective of pruning is to remove unnecessary parameters from a neural network, so that they do not participate during training and/or inference. It can be done in the convolutional phase on the channels, kernels and weights or even in the fully connected phase on the neurons. In (Masson et al., 2021) they show a classification of pruning methods for channels. They categorize the pruning methods for channel reduction, and they also specify the criteria used to select these channels: based on weights or based on feature maps.

We have seen that network pruning has achieved a great importance in the literature as many researchers have applied different techniques to simplify a CNN using a pruning scheme. In (Liu et al., 2019) they classify the pruning methods in unstructured and structured pruning, and make a review of all the state-of-art structured pruning methods. Unstructured pruning methods remove weights without following any order. For the structured methods, there are some rules or even constraints which define how the pruning is done (Anwar et al., 2017). Typically, the pruned layers appertain to the convolutional phase (Luo et al., 2017). In our proposal, we instead apply a structured pruning scheme to the fully-connected layers.

Among pruning methods, the value-based *weight pruning* (Han et al., 2015) and *neuron pruning* (Srinivas and Babu, 2015) have arisen as the most used, particularly due to their simplicity. The logic behind this pruning methods is straightforward: a certain amount (%) of the weights or neurons that contribute less to the final trained model are removed from the architecture. This makes the network quicker to perform inference and endows it with better generalization capabilities. However, multiple pruning and retraining steps demonstrated that it is possible to recover fully or partially the knowledge lost in the pruning phase. Further along the series of pruning approaches published to date, *Polynomial Decay* (Zhu and Gupta, 2018) is a scheduled pruning method that considers that a higher amount of weights can be pruned in early stages of pruning, while systematically less amount of weights should be pruned in late stages. Between pruning steps the network is retrained for some epochs. An implementation of the discussed methods can be found for Tensorflow. [1]

Pruning a CNN model reduces its complexity, but sometimes leads to a decrease of the performance of the model, although there are some proposals that reduce the complexity of the model with no loss of accuracy (Han et al., 2016).

Pruning a neural network can be conceived as an optimization method in which we start from the original vector, and connections/neurons are decision variables whose value is evolved towards optimizing a given objective. In this context of evolution of neural networks, evolutionary algorithms for evolving DL architectures have been applied (Iba, 2018). While this combination of EA and DL models seemed to be a great scheme for the optimization of DL models, especially for CNN network, the optimization of DL models is still an open problem (Liu et al., 2017b). Many proposals have been published about this problem like in (Martinez et al., 2021), where they make a review of proposals using EAs for optimizing DL models, prescribing challenges and future trends to effectively leverage the synergy between these two areas.

Researchers have presented a great variety of proposals about the optimization of DL models using EA, most commonly for CNN. In (Martín et al., 2018), the authors developed EvoDeep, an EA with specific mutation and crossover operators to automatically create DL models from scratch. Moreover, in (Real et al., 2017) a novel evolution approach to evolve CNN models using a GA was proposed. Another example of the optimization of CNN was developed in (Assunçao et al., 2019), in which a GA was presented for the optimization of the topology and parameters of the CNN.

In our proposal, we improve the performance of the models using a TL approach to extract the features of the images and apply a reduction of the fully-connected layer using a GA to optimize a sparse layer.

2.4. Evolutionary Algorithms for CNN Pruning

In the previous section, several works of CNN pruning have been presented, but none of them use an EA to prune. In this section, we mention some studies present in the literature which have used an EA in order to prune a CNN model. To begin with, in (Liu et al., 2017a), the authors propose a sparse approach to reduce CNN complexity. EAs are also a good way to prune CNN. In (Mantzaris et al., 2011), a first attempt of pruning and GA is proposed for a medical application. They use a GA to search for redundancy factors

[1] https://www.tensorflow.org/model_optimization/guide/pruning, Tensorflow Pruning. Last access: 28/01/2022

in a neural network. Moreover, in (Samala et al., 2018) another EA is presented to prune deep CNN for breast cancer diagnosis in digital breast tomosynthesis. A combined approach of EA and sparse is proposed by Wang et al. (Wang et al., 2020), in which a GA and sparse learning are applied to a scheme of network channel pruning in the convolutional scheme of the CNN. For pruning CNN, not only GAs but also other algorithms are used, like Differential Evolution (DE). In (Salehinejad and Valaee, 2021) the authors propose to use a Differential Evolution algorithm to prune the convolutional phase and the fully-connected phase of some Deep CNN, obtaining a reduction of the model but a small decrease of its performance.

However, all previous works are focused on reducing the complexity, using the EA to reduce the accuracy loss of the pruned network. Also, many of them try to reduce the whole model, changing the complete architecture and making the pre-trained values unusable. The re-training of the network may be a time-consuming task, so we assume that TL is useful in this context. We therefore maintain the original architecture with pre-trained values. Our model focuses on improving the performance of the model by pruning connections of the fully-connected layers using a GA to evolve the connections. In this environment, the search space of the GA is narrower and a faster convergence of the algorithm may be reached.

In addition to that, in the field of neural architecture search, more advanced techniques have been developed. Among them, in Section 2.2, either (Real et al., 2019) and (Real et al., 2020) have been commented. Nonetheless, they also have a great relevance in this section. The first one evolves a classifier, whereas in the second one, the authors develop an evolutionary search to build a model from scratch.

2.5. Feature Selection and Deep Learning

One of the advantages of using TL is reducing the required time to train a DL architecture. Nonetheless, the result of this process may lead to recognize patterns that are not useful to address the problem at hand. For that reason, once TL is applied, a FS process to obtain the best features might lead to a better performance of the neural network (Roy et al., 2015).

An example of this process is presented in an arrhythmia detection task addressed in (Yildirim et al., 2018), in which the authors propose a mechanism based on feature extraction and selection to improve and ultimately obtain one of the best results for this problem. In relation to medical problems, the combination of FS and DL is also used in cancer diagnosis and digital breast tomosynthesis. In (Samala et al., 2018) they use a TL approach and then a FS process followed by an evolution through a GA that leads to a reduced network with the same performance. Another example is described for remote sensing scene classification, in which the FS makes an impact to improve the performance of the neural network models (Zou et al., 2015), as the authors formulate the FS problem as a feature reconstruction problem. Their iterative method selects the best features to solve this problem as the discriminative features.

In our proposal, if we assume that TL is applied and we only have one fully-connected layer, then the pruning is made in relation to the extracted features of the network and, therefore, we are making a selection of the features that adjust at best to the tackled problem.

3. Evolutionary Pruning for Deep Transfer Learning

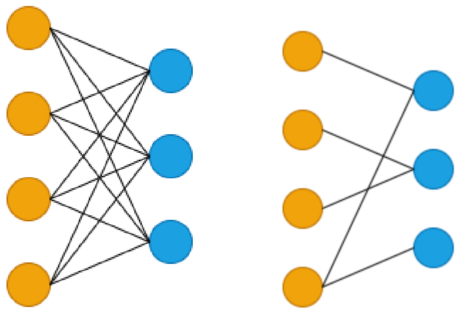
This section describes EvoPruneDeepTL, which is a model that replaces fully-connected layers with sparse layers optimized using a genetic algorithm in a TL approach. Subsection 3.1 gives a notion of the concept of sparse layer and the description of EvoPruneDeepTL. In Subsection 3.2, we define the evolutionary components of EvoPruneDeepTL. The description of the process of creating the network and how the pruning is made is shown in Subsection 3.3.

3.1. Global scheme of Evolutionary Pruning for Deep Transfer Learning

In a fully-connected layer, all neurons of each side are connected. Sometimes, all these connections may not be necessary, and the learning process can be reduced. For that reason, the fully-connected layer can be replaced by a sparse layer, in which some connections are eliminated.

In this work, our goal is to improve the performance of the neural network and, at the same time, to decrease the maximum number of connections or neurons. To this end, we use a sparse layer, which is composed of a subset of all connections of a fully-connected layer.

Fig. 1a shows the fully-connected network architecture, while Figure 1b represents the sparsely connected architecture with a connection matrix of 4×3 because we have 3 classes (blue circles) and 4 neurons of the previous layer.



(a) Fully-Connected Layer

(b) Sparse Layer

Figure 1: Representation of both architectures

In this study, we propose a novel method to prune the neurons, that considers the removal of both single connections and groups of connections of the input connections of a specific neuron, as can be observed in Fig. 2. Fig 1b shows a sparse layer that leads to the encoding strategy used in this work. This encoding, which is represented by the chromosome of the GA, is required is required to know exactly which connections are removed.

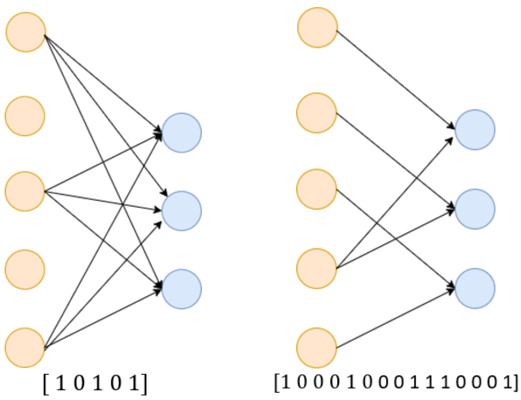
EvoPruneDeepTL model utilizes a GA designed to optimize the connections of a sparse layer. The GA takes each individual as a mask for the neural network and creates a sparse layer activating from the mask. This optimized mask gives rise to a pruned neural network suitable for the problem under consideration.

The optimization is performed using both methods, either by groups of connections or by single connections. The genome representation of each chromosome of the GA is binary-coded and represents the active *neurons* or the active *connections*. The GA evolves the configuration of the network towards its best pruned variant in terms of accuracy. Next, we describe both encoding strategies:

- **Neurons**: each gene of the chromosome represents the number of active neurons. A value 1 in position i means that the neuron i is active, and a 0 that is inactive. A non-active neuron implies that all the input connections are removed both in training and inference times. The length of the chromosome in this case is the number of neurons of the sparse model.

- **Connections**: each gene represents the connection between the layers. The interpretation of the binary values is the following: if a gene is 1, the connection between the corresponding layers exists, otherwise, that connection does not exist. Therefore, the length of the chromosome is the maximum number of connections, $D = D_1 \times D_2$, where D_1 is the number of neurons in the previous layer, and D_2 is the number of neurons in that layer.

An example of both encoding strategies is shown in Fig. 2. In both cases the pruned connections are from the input on the layer, i.e. the right layer. The left image shows a representation of neuron-wise encoding, in which a group of neurons is selected to be active and the rest are pruned. The right image depicts how single connections are pruned.



(a) Neuron Encoding

(b) Single Connection Encoding

Figure 2: Representation of encoding strategies

3.2. Evolutionary components of EvoPruneDeepTL

In this subsection we introduce the evolutionary components of EvoPruneDeepTL. It is a steady-state genetic algorithm for the previous mentioned encoding strategies (neuron encoding vs single connection encoding, Fig 2): in each iteration two individuals are selected and crossed, producing two offsprings, that could be mutated by a p_{mut} probability. The offspring candidates are introduced in the population only if they improve the worse candidates in the population, replacing them.

As previously mentioned, in EvoPruneDeepTL each chromosome is a binary array and each gene represents a connection between two layers. Each generation follows the classical scheme of selection, crossover, mutation and replacement. The best solutions found during the evolutionary search are kept in a population of individuals. Next, we describe the different components:

Selection: the implemented selection operator is Negative Assorting Mating (Fernandes and Rosa, 2001). The first parent is picked uniformly at random, while the second parent is selected between three possible candidates. The candidate with more Hamming distance from the first parent is chosen as the second parent, thereby ensuring that the recombined parents are diverse. This selection method allows for a higher degree of exploration of the search space.

Crossover: EvoPruneDeepTL uses the uniform crossover operator shown in Expression 1. Given two parents **P** and **Q**, where $\mathbf{P} = \{p_i\}_{i=1}^{D}$ and $\mathbf{Q} = \{q_i\}_{i=1}^{D}$. Then two offsprings $\mathbf{P'} = \{p_i'\}_{i=1}^{D}$ and $\mathbf{Q'} = \{q_i'\}_{i=1}^{D}$ are created following the equations:

$$\begin{aligned} pi' &= \begin{cases} p_i & \text{if } r \leq 0.5 \\ q_i & \text{otherwise} \end{cases} \\ q_i' &= \begin{cases} q_i & \text{if } r \leq 0.5 \\ p_i & \text{otherwise} \end{cases} \end{aligned} \quad (1)$$

where r is the realization of a continuous random variable with support over the range $[0.0, 1.0]$.

Mutation: EvoPruneDeepTL adopts the so-called single point mutation. A mutation probability for each individual is defined by p_m. Then, a gene of that individual is uniformly randomly selected and its bit is flipped.

Replacement Strategy: at the end of every generation, the two individuals resulting from the crossover and mutation operators compete against the worst two elements. As a result, the population is updated with the best two individuals among them, i.e. those whose fitness value is better.

Initialization: the genes composing the individuals are initialized to 0 or 1 as per the following probabilistic condition with a p_{one} probability:

$$I_i = \begin{cases} 1 & \text{if } r \leq p_{one} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where r is the realization of a uniform continuous random variable with support $[0.0, 1.0]$.

Evaluation of individuals: the fitness value of every individual is given by the accuracy over a test dataset of the neural network pruned as per the decoded individual, and trained over the training dataset of the task at hand.

In Algorithm 1, we show a brief pseudocode of EvoPruneDeepTL. The skeleton of EvoPruneDeepTL is based on a GA in which the evaluation of an individual is reformulated. In our case, each of them is decoded into a mask that leads to a sparse layer. Then, the network is created and trained over the train set. Finally, the fitness of this individual is given by the accuracy of this network evaluated over the test set.

Algorithm 1: EvoPruneDeepTL

```
   Result: Pruned network
1  Initialization of configurations;
2  Evaluation of population (see lines 10-12);
3  while evaluations < max_evals do
4      Parent selection;
5      Generate offsprings;
6      if mutation_condition then
7          Perform mutation;
8      end
9      for each child in children population do
10         SparseNet ← Create sparse network using individual;
11         TrainedNet ← Train SparseNet using train set;
12         Child fitness ← Accuracy of TrainedNet evaluated in test set;
13     end
14     Replacement Strategy;
15 end
```

3.3. EvoPruneDeepTL Network

This proposal is designed assuming that part of the network to be pruned is imported from another pretrained model, i.e., TL. In this work, we have selected one of the most influential CNNs, ResNet-50, although other network models such as Inception, VGG, etc, can also be used. In particular, we used the feature extractor of ResNet-50 pretrained on Imagenet and we removed the fully connected layers. Moreover, two different compositions of these last layers are used in this work. The first one is only composed of one layer of 512 neurons, which will be the sparse layer, followed by the output layer. The second architecture of these layers presents two fully-connected layers of also 512 neurons and finally the output layer. For this case, single and both layer optimization will be performed by EvoPruneDeepTL.

The sparse layer is created using an adjacency matrix that defines the connections between the fully-connected layers. EvoPruneDeepTL performs the pruning in relation to the connections that compose the input of the neuron. Consequently, there may be some neurons of the second layer which have no connection from the previous layer. Attending to the number of layers and the layer that is going to be pruned, we describe the different models at hand. EvoPruneDeepTL is able to prune either architectures with one or two fully-connected layers. The case of its application to one layer is shown in Fig. 3. Moreover, when EvoPruneDeepTL is applied to architectures of two layers, three cases are possible: pruning the first layer (Fig. 4), pruning the second layer (Fig. 5) or both layers. The Feature Selection approach is applied to one layer models and is shown in Fig. 6.

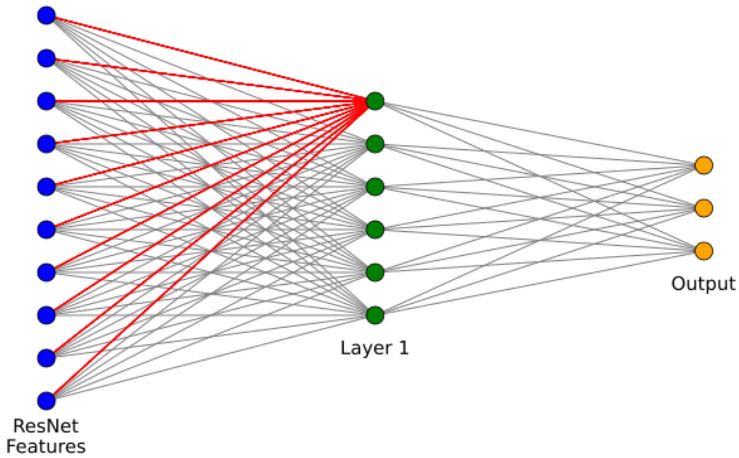


Figure 3: Pruning architectures with one layer

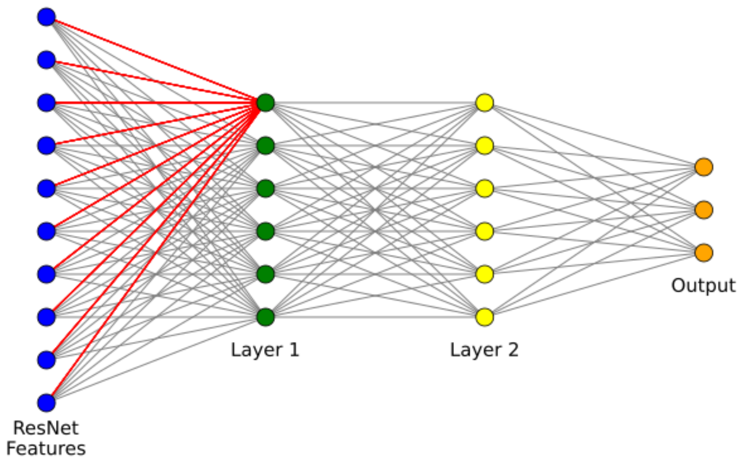


Figure 4: Pruning first layer of architectures with two layers

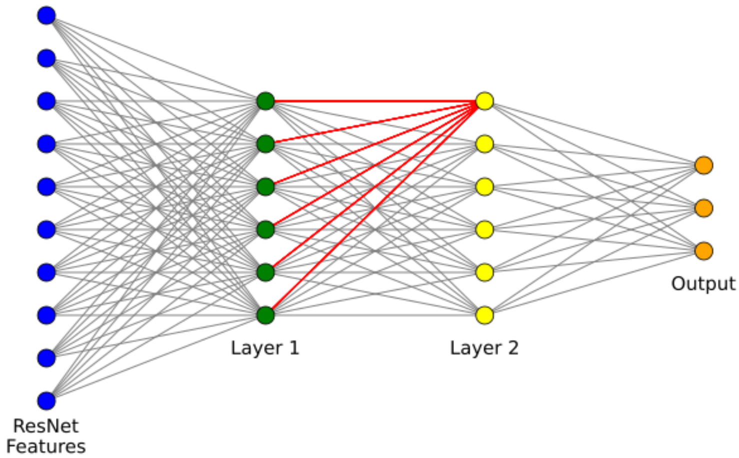


Figure 5: Pruning second layer of architectures with two layers

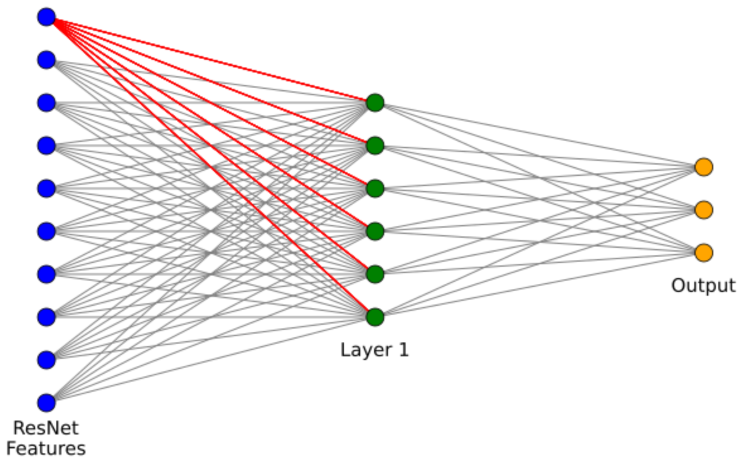


Figure 6: Feature Selection of architectures with one layer

4. Experimental Framework

In this section, we describe the experimental framework followed in our study. First, we give a brief description of the analyzed datasets. Then, the training setup is presented, emphasizing the parameters of EvoPruneDeepTL and the experimental conditions.

4.1. Datasets

In our study, we have chosen several diverse and representative datasets that are suitable for TL due to their size, as they require less training and inference time. Therefore, these datasets are suitable for population metaheuristics, as many individuals are evaluated. The selected data sets are shown in Table 1, which portrays their main characteristics for our experiments.

These datasets are diverse and taken from the literature:

Table 1: Datasets used in the experiments.

Dataset	**Image Size**	**# classes**	**# Instances (train / test)**
SRSMAS	$(299, 299)$	14	333 / 76
RPS	$(300, 300)$	3	2520 / 372
LEAVES	$(256, 256)$	4	476 / 120
PAINTING	$(256, 256)$	5	7721 / 856
CATARACT	$(256, 256)$	4	480 / 121
PLANTS	$(100, 100)$	27	2340 / 236

- SRSMAS (Gómez-Ríos et al., 2019) is a dataset to classify coral reef types with different classes and high distinction difficulty.
- RPS (Moroney, 2019) is a dataset to identify the gesture of the hands in the popular Rock Paper Scissors game from images that have different positions and different skin colors.
- LEAVES is composed of images of healthy and unhealthy citrus leaves, with different shades of green (Rauf et al., 2019).
- PLANTS is another dataset from the natural environment (Singh et al., 2020), in which the task is to differentiate between leaves of different plants such as tomato, apple or corn, among others.
- CATARACT comes from the medical domain (Choi, 2020), whose purpose is to classify different types of eye diseases.
- PAINTING is related to the painting world (Museum, 2018). The images in this dataset have been taken from a museum and the task is to recognize different types of paintings.

Examples for several of the above datasets are shown in Fig. 7.

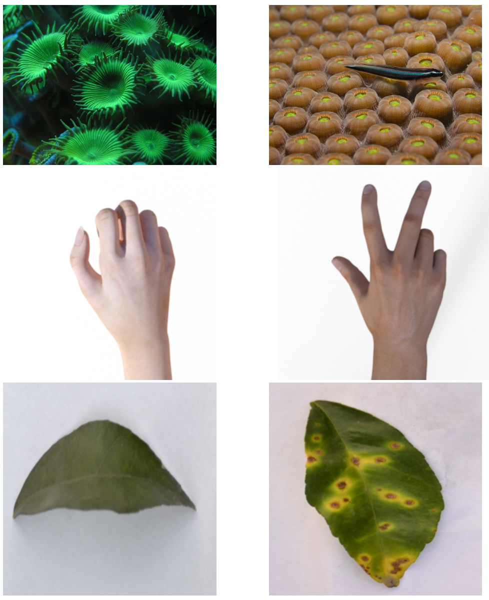


Figure 7: Images of datasets. Top: SRSMAS examples. Middle: RPS examples. Bottom: LEAVES examples.

4.2. Training setup

The evaluation of EvoPruneDeepTL requires splitting the images of the datasets in train and test subsets. As the results could strongly depend on the train and test sets, we have applied in SRSMAS and LEAVES a 5 fold cross-validation [2]

For the remaining datasets, the train and test had already been defined beforehand, so we have used them for the sake of replicability.

The training is done using SGD as optimizer, a batch size of 32 images, and maximum 600 epochs, but the training stops when no improvement of loss is obtained in ten consecutive epochs. The model with the greater accuracy on the training set is saved. As we apply TL, only the last layers are trained, whereas the remaining ones are frozen with the parameter values imported from the pre-trained ResNet-50 network.

Table 2: Parameters of EvoPruneDeepTL.

Parameter	**Value**
Maximum Evals	200 (one layer) 300 (both layer)
# Runs	5
Population size	30
NAM	3
p_{mut}	0.07
Batch Size	32

The parameters of EvoPruneDeepTL are indicated in Table 2. We have set the maximum evaluations to two different values, 200 and 300, because there are some experiments we carry out to analyze the behavior of EvoPruneDeepTL that need an adaptation of this value because the search space in these experiments is wider. The best solution found in terms of accuracy is returned. We note that in case of several solutions with the same accuracy, the returned solution is the configuration with the lowest percentage of active neurons. Note that the number of runs and total function evaluations is kept low to meet a computationally affordable balance between performance and the high execution times required for simulation. This is shown in Table 3, in which the average time per execution of the models with two layers is indicated. Unfortunately, this limited number of runs per experiment impedes the application of statistical tests to assess the significance of the reported differences, as tests conventionally used for this purpose require larger sample sizes to reach meaningful conclusions.

Table 3: Average time per run of EvoPruneDeepTL.

Dataset	First Layer	Second Layer	Both Layers
SRSMAS	9h 41min	10h 45min	15h 27min
RPS	4h 23min	5h 13min	8h 00min
LEAVES	10h 26min	16h 01min	17h 45min
PAINTING	13h 1min	15h 24min	22h 45min
CATARACT	2h 3min	2h 22min	3h 26min
PLANTS	6h 3min	6h 11min	10h 00min

All the following experiments have been carried out using Python 3.6 and a Keras/Tensorflow implementation deployed and running on a Tesla V100-SXM2 GPU. The code is published in a open repository in GitHub.[3]

[2]Sets for 5 fold CV for SRSMAS and LEAVES:
https://drive.google.com/drive/folders/1Xf7OeZyWDDG-_Y4VX_nnAdfz3Kwhy8LU?usp=sharing. Last Access: 28/01/2022

[3]Repository with the code of EvoPruneDeepTL: https://github.com/ari-dasci/S-EvoDeepTLPruning. Last access: 28/01/2022

5. Results and Discussion

In this section, we analyze the behavior of EvoPruneDeepTL. In order to show the benefits of using EvoPruneDeepTL, we propose four research questions (RQ) that they are going to be answered with different and diverse experiments over several datasets are carried out. We will show tables with the results of these experiments and we will analyze them to ensure the benefits of EvoPruneDeepTL. These RQ are the following ones:

(RQ1) which is the performance of EvoPruneDeepTL against fully-connected models?

We compare EvoPruneDeepTL against non-pruned models comprising fully-connected layers to study which model obtains a better performance in the experiments. Moreover, we remark the flexibility of EvoPruneDeepTL applying it with one and several layers.

(RQ2) which would be better, to remove neurons or connections?

We compare EvoPruneDeepTL using the two alternatives explained in the previous section: 1) pruning the neurons or 2) each individual represents exact connections between neurons, allowing for a more finely grained optimization. The goal of this section is to check which representation obtains the best results. On the one hand, the neuron representation of the length of chromosomes is shorter, so the domain search is smaller. On the other hand, the connections representation is a more fine-detail representation, so it could potentially allow the algorithm to obtain better results.

(RQ3) which is the performance of EvoPruneDeepTL when compared to efficient pruning methods?

We compare the performance of EvoPruneDeepTL against several efficient pruning methods published in the literature for compressing CNN networks: Polynomial Decay (Zhu and Gupta, 2018) , Weight Pruning (Han et al., 2015) and Neuron Pruning (Srinivas and Babu, 2015). This comparison of EvoPruneDeepTL and the CNN models is made in terms of accuracy and model compression.

(RQ4) which would be better, the use of pruning of fully-connected layers or Feature Selection?

A particular case of EvoPruneDeepTL stands when the optimization of the network is done with one fully-connected layer and features are evolved towards the fittest for the problem at hand. Our aim is to check if this scheme improves the overall performance of the model over a dataset in terms of the accuracy of the models.

This section is divided in Section 5.1, where the comparison of the diverse representations of pruning that EvoPruneDeepTL makes against the reference models is presented to answer RQ1. Next, Section 5.2 discusses whether EvoPruneDeepTL should operate over neurons or connections to analyze RQ2. Section 5.3 provides a complete comparison among EvoPruneDeepTL and other efficient pruning methods in order to solve RQ3. Section 5.4 explains the approximation of Feature Selection. A whole comparison against all the previous models is made to assess the importance of the Feature Selection to answer RQ4.

5.1. Answering RQ1: Pruning

In this section, we assess the performance gaps between the proposed EvoPruneDeepTL against other reference models to answer RQ1. In each subsection, several and diverse experiments are carried out to present results that assure the quality of EvoPruneDeepTL when it is compared to other models. This pruning section is composed of Section 5.1.1, in which we compare EvoPruneDeepTL against reference models with only one layer; of Section 5.1.2 we make the same experiments but with two layers and the optimization of one of them, and of Section 5.1.3, that shows the optimization of two consecutive layers at the same time.

In the following, we describe the different reference models:

- The first reference is composed of fully-connected layers of 512 units and the output layer. That is equivalent to the model with all neurons in active mode (all gens to 1). This model is the one with all active neurons, we call it *Not Pruned*.

- A grid search scheme is compared to EvoPruneDeepTL to check whether the improvement made by EvoPruneDeepTL could be obtained with a simple search over the percentage of neurons of the fully-connected layer. We have tested the fully-connected model with different number of neurons: 10% to 90% of its total units increasing this percentage by 10% (including both), and for each dataset we have identified the number of neurons which gives the best accuracy.

- The best result of the above models is also noted and it is called *Best Fixed*. When implemented over both layers, pruning is referred to as *Best Fixed Both*.

5.1.1. Pruning neurons of one fully-connected layer

This section introduces the results of pruning models with only one fully-connected layer. Table 4 shows the comparison of EvoPruneDeepTL against the reference models. In this case, the reference model only has one fully-connected layer composed by 512 units and the output layer.

For each dataset, the first row shows the obtained average accuracy by the models over the test set, whereas the second row informs about the average percentage of active neurons.

Table 4: Average results of EvoPruneDeepTL against not pruned models with one fully-connected layer.

Dataset	Measure	Not Pruned	Best Fixed	EvoPrune DeepTL
SRSMAS	Accuracy	0.832	0.866	**0.885**
	% Active neur.	100	20	25
RPS	Accuracy	0.938	0.938	**0.954**
	% Active neur.	100	40	46
LEAVES	Accuracy	0.923	0.927	**0.935**
	% Active neur.	100	80	38
PAINTING	Accuracy	0.939	0.945	**0.951**
	% Active neur.	100	60	46
CATARACT	Accuracy	0.703	0.719	**0.732**
	% Active neur.	100	70	39
PLANTS	Accuracy	0.432	0.432	**0.480**
	% Active neur.	100	10	49

These results show how EvoPruneDeepTL is capable of distinguishing the pruning configurations that lead towards an improvement of performance of the models, as it obtains a greater accuracy in all the datasets for every reference model. Moreover, in most datasets a higher compression ratio than the best fully-connected model is also achieved.

5.1.2. Pruning neurons of two fully-connected layers

In this section, our challenge is to improve the performance of a two fully-connected layer network. For that reason, EvoPruneDeepTL is applied to each layer individually.

The results of applying EvoPruneDeepTL to each layer individually are shown in Table 5, where **First Layer** indicates the case of the optimization of the first layer, and **Second Layer** describes the other case. In this case, Both *Not Pruned* and *Best Fixed* are the reference models with two fully-connected layer.

In this case, results follow the same path as the previous one: in all the datasets EvoPruneDeepTL achieves an improvement of the accuracy over the reference models. Moreover, the Second Layer case obtains more compressed networks than the First Layer option.

Comparing the results of the scheme of one and two layers, both have similar results, only in RPS and CATARACT the difference in terms of accuracy is higher. Thus, these experiments have shown the ability of EvoPruneDeepTL of improving the overall performance of networks and, at the same time, reducing their complexity.

Table 5: Average results of EvoPruneDeepTL against not pruned models with two fully-connected layers.

		First Layer			**Second Layer**		
Dataset	Measure	Not Pruned	Best Fixed	EvoPrune DeepTL	Not Pruned	Best Fixed	EvoPrune DeepTL
SRSMAS	Accuracy	0.858	0.858	**0.883**	0.858	0.860	**0.884**
	% Active neur.	100	100	46	100	80	47
RPS	Accuracy	0.922	0.938	**0.959**	0.922	0.949	**0.969**
	% Active neur.	100	30	37	100	30	16
LEAVES	Accuracy	0.919	0.926	**0.937**	0.919	0.929	**0.935**
	% Active neur.	100	40	28	100	60	12
PAINTING	Accuracy	0.939	0.944	**0.950**	0.939	0.941	**0.951**
	% Active neur.	100	60	53	100	90	53
CATARACT	Accuracy	0.703	0.711	**0.740**	0.703	0.703	**0.735**
	% Active neur.	100	70	63	100	100	59
PLANTS	Accuracy	0.402	0.448	**0.479**	0.402	0.441	**0.483**
	% Active neur.	100	10	45	100	50	37

5.1.3. Pruning neurons of both layers

In the previous sections we have tested EvoPruneDeepTL to *single-layer* optimization problems. In this section we increase the difficulty of the problem: the optimization of two consecutive fully-connected layers.

From the previous experiments, we have run EvoPruneDeepTL with 200 evaluations, but we have noticed that this number of evaluations might not be enough. This is due to the fact that we have now individuals of size 1024, 512 for each layer, and the search space is larger than in the rest of experiments. We have therefore also carried out the experiments with 300 function evaluations.

In Table 6, we show the results for reference models and EvoPruneDeepTL with 300 evaluations. The reference models stand the same as in the previous cases, but as they are implemented over both layers, the pruning is now referred to as *Best Fixed Both*.

In some cases, the percentage of remaining active neurons is higher than in the first and second layer models, but that is due to the complexity of this new problem. However, the performance of the network in these experiments indicates that the best option for pruning is achieved when the optimization is done to two consecutive layers.

Table 6: Average results of EvoPruneDeepTL against not pruned methods optimizing two consecutive layers.

Dataset	Measure	Not Pruned	Best Fixed Both	EvoPrune DeepTL
SRSMAS	Accuracy	0.858	0.863	**0.885**
	% Active neur.	100	50	64
RPS	Accuracy	0.922	0.946	**0.978**
	% Active neur.	100	90	12
LEAVES	Accuracy	0.919	0.934	**0.936**
	% Active neur.	100	15	34
PAINTING	Accuracy	0.939	0.949	**0.953**
	% Active neur.	100	40	51
CATARACT	Accuracy	0.703	0.735	**0.746**
	% Active neur.	100	85	63
PLANTS	Accuracy	0.402	0.466	**0.491**
	% Active neur.	100	55	41

These results prove the attainment of a sequential process to make pruning of DL models by adding layers and then, evolving their neurons to achieve a reduced configuration of the network. This process rises the performance of the models in terms of accuracy.

5.2. Answering RQ2: which would be better, to remove neurons or connections?

This section is devised to formally answer the RQ2, which is to decide if it is better perform pruning of whole neurons or either single connections, by comparing different EvoPruneDeepTL chromosome representations: neurons and connections, as we described in Section 3. Two representations are shown in this section: the neuron representation, in which a gen represents the connections of a neuron, and the connections representation, in which a gene represents a specific connection in the sparse layer. Neuron representation obtains shorter chromosomes than the connections one. Meanwhile, the connection representation leads to a more detailed representation and a larger domain search.

Table 7 shows for each dataset and representation the mean accuracy and % of active connections for both pruning methods. The connection strategy is named *Edges*. The results show that even though there are some cases in which the edges optimization achieves a similar performance of the network, the neuron optimization presents more robust results. The models working at the neuron level are even able to further reduce the number of active neurons in some of the datasets.

As a conclusion of this experiment, we can confirm that using the neuron approach is the best representation and that the second layer model gives us more consistent results than the first layer pruning model, both in accuracy and in reduction of the model.

5.3. Answering RQ3: Comparing EvoPruneDeepTL with efficient methods for CNN pruning

This section is devised to analyze the RQ3 comparing EvoPruneDeepTL to other well known network pruning methods to present results that measure the performance of our model against these methods. This comparison is conducted in terms of quality and computational complexity, aimed to prove the potential of EvoPruneDeepTL with respect to other pruning counterparts. To this end, we implement two different pruning methods, namely, weight pruning and neuron pruning. These methods have a parameter in common, $S_f \in \mathbb{R}(0, 1)$, which denotes the target pruning percentage. It is set to the same percentage of reduction that EvoPruneDeepTL has obtained in the experiments discussed previously. Next, we briefly describe each of such methods:

Table 7: Average results of EvoPruneDeepTL against edges models.

		One Layer		Two Layers		
Dataset	Measure	Edges	EvoPruneDeepTL	Edges	EvoPruneDeepTL Layer 1	EvoPruneDeepTL Layer 2
SRSMAS	Accuracy	0.875	**0.885**	0.875	0.883	**0.884**
	% Active neur.	43	25	46	46	47
RPS	Accuracy	0.952	**0.954**	0.952	0.959	**0.969**
	% Active neur.	29	46	37	37	16
LEAVES	Accuracy	0.932	**0.935**	0.933	**0.937**	0.935
	% Active neur.	45	38	45	28	12
PAINTING	Accuracy	0.949	**0.951**	0.950	0.950	**0.951**
	% Active neur.	48	46	53	48	53
CATARACT	Accuracy	0.729	**0.732**	0.737	**0.740**	0.735
	% Active neur.	69	49	66	63	59
PLANTS	Accuracy	0.457	**0.480**	0.463	0.479	**0.483**
	% Active neur.	64	49	45	45	37

- `weight` (Han et al., 2015): Parameters with lower values are pruned at once. This method operates over the whole parameter set in the layer to be optimized. (Parameters: S_f)
- `polynomial decay` (Zhu and Gupta, 2018): Parameters are pruned guided by a Polynomial Decay schedule to the specified sparsity value. Between pruning steps, the network is allowed to fine tune for 5 epochs. This model is also applied over the whole parameter set in the layer to be optimized. Parameters used in the experimentation are listed in Table 8.
- `neuron` (Srinivas and Babu, 2015): Neurons with lower mean input connection values are pruned. (Parameters: S_f) as in Figs. 3, 4 and 5. (Parameters: S_f)

Table 8 summarizes the value of the parameters of Polynomial Decay algorithm, which have been adapted to our experiments. Then, given a desired sparsity value of S, the sparsity is updated over a span of k pruning steps following the next equation

$$S_k = S_f + (S_i - S_f) \cdot \left(1 - \frac{K_k - K_i}{K_f - K_i}\right)^{\alpha} \text{ if } K_k \bmod F = 0 \tag{3}$$

wherein parameters are described as follows:

- $S_{i,f} \in \mathbb{R}(0, 1)$ are the initial and final sparsity percentages.
- S_f depends on the experiment. It is the percentage of pruning that EvoPruneDeepTL has achieved and the end of the generations.
- $K_{i,f} \in \mathbb{N}$ configures at what training step the pruning algorithm starts and ends.
- $K_k \in \mathbb{N}(K_i, K_f)$ is the current step.
- nb is the number of batches. It is calculated as the length of the training set divided by the batch size.
- F configures the frequency at which Equation 3 is computed.

Parameters of the Polynomial Decay model are chosen to achieve a tradeoff between network recovery and the number of training epochs. Given the nature of this model, Polynomial Decay implies more training epochs than the implemented neuron and weight pruning methods. This fact could make the comparison

Table 8: Parameter values of Polynomial Decay.

Parameter	Value
S_i	0.1
K_i	0
K_f	$nb \cdot 25$
F	$nb \cdot 5$
α	3.0

between such methods unfair if the additional training epochs introduced by the Polynomial Decay model are high compared to the initial training epochs (i.e 600). To avoid this situation, Polynomial Decay is configured so that it sufficiently guarantees network recovery for all datasets while a minimal amount of extra training epochs are carried out, just an extra 4% from the initial 600 epochs (i.e 25 extra epochs).

Our analysis aims to verify whether the performance of the above efficient pruning methods are comparable to EvoPruneDeepTL in terms of solution quality (accuracy) when they are configured to prune the same amount of parameters. Thus, the experimentation is carried out for the previously four cases discussed, selecting the average outcomes from the experimentation conducted in this point.

First, we show the results of this comparison when only a fully-connected layer is used for the optimization. Table 9 shows the results for this case. EvoPruneDeepTL outperforms the CNN models in five out of the six cases, but only in PAINTING these results are better for the Polynomial Decay or Weight models.

Table 9: Average results of EvoPruneDeepTL against efficient CNN for one layer models.

		One Layer			
Dataset	Measure	Weight	Poly. Decay	Neuron	EvoPrune DeepTL
SRSMAS	Accuracy	0.805	0.823	0.745	**0.885**
	% Active neur.	25	25	25	25
RPS	Accuracy	0.917	0.927	0.869	**0.954**
	% Active neur.	46	46	46	46
LEAVES	Accuracy	0.918	0.920	0.886	**0.935**
	% Active neur.	38	38	38	38
PAINTING	Accuracy	0.993	**0.994**	0.874	0.951
	% Active neur.	46	46	46	46
CATARACT	Accuracy	0.678	0.679	0.658	**0.732**
	% Active neur.	39	39	39	39
PLANTS	Accuracy	0.406	0.411	0.365	**0.480**
	% Active neur.	49	49	49	49

Second, Table 10 shows the results for the optimization of models with two layers, where **First Layer** indicates the cases of the optimization of the first layer and **Second Layer** describes the cases of the second layer. Results point out that EvoPruneDeepTL outperforms most of the methods in all the models and datasets. This case presents similar results as the one layer case because only in the PAINTING dataset EvoPruneDeepTL has a lower performance in relation to the literature methods. As a result of that, EvoPruneDeepTL's robustness in performance over the literature methods has been shown in one-layer and two-layer networks.

Lastly, we compare the execution times for all the models. Evolutionary approaches are known to converge slowly in highly-dimensional search spaces as the one tackled in this paper. For that reason, in this

Table 10: Average results of EvoPruneDeepTL against efficient CNN pruning methods for two layers models.

		First Layer				**Second Layer**				**Both Layers**			
Dataset	Measure	Weight	Poly. Decay	Neuron	EvoPrune DeepTL	Weight	Poly. Decay	Neuron	EvoPrune DeepTL	Weight	Poly. Decay	Neuron	EvoPrune DeepTL
SRSMAS	Accuracy	0.795	0.815	0.775	**0.883**	0.834	0.837	0.779	**0.884**	0.845	0.847	0.647	**0.885**
	% Active neur.	46	46	46	46	47	47	47	47	64	64	64	64
RPS	Accuracy	0.886	0.911	0.803	**0.959**	0.845	0.911	0.696	**0.969**	0.694	0.899	0.490	**0.978**
	% Active neur.	37	37	37	37	16	16	16	16	12	12	12	12
LEAVES	Accuracy	0.913	0.918	0.812	**0.937**	0.904	0.919	0.712	**0.935**	0.911	0.925	0.747	**0.936**
	% Active neur.	28	28	28	28	12	12	12	12	34	34	34	34
PAINTING	Accuracy	**0.995**	0.993	0.850	0.950	0.937	0.938	0.920	**0.951**	0.934	0.940	0.853	**0.953**
	% Active neur.	53	53	53	53	53	53	53	53	51	51	51	51
CATARACT	Accuracy	0.668	0.684	0.673	**0.740**	0.694	0.689	0.648	**0.737**	0.686	0.696	0.611	**0.746**
	% Active neur.	63	63	63	63	59	59	59	59	63	63	63	63
PLANTS	Accuracy	0.408	0.403	0.343	**0.479**	0.392	0.420	0.313	**0.482**	0.393	0.411	0.278	**0.491**
	% Active neur.	45	45	45	45	37	37	37	37	41	41	41	41

section we also want to compare the required time of EvoPruneDeepTL and the other traditional approaches. Table 11 shows the time in seconds for each models. From these results, in terms of computational efficiency, our method suffers from the convergence slowness derived from the exploration of large search spaces.

To summarise, in this section we have fairly compared EvoPruneDeepTL to other well-known pruning methods, such as weight pruning and neuron pruning, guided by different pruning techniques. EvoPruneDeepTL is distinguished from other pruning methods due to the fact that they are advocate for shrinking the through their pruning process, but with an admissible decrease of the accuracy. Although our model is slower in terms of execution time, it scores higher accuracy levels than those of traditional pruning counterparts. Therefore, we conclude that EvoPruneDeepTL excels at determining which parameters to tune in neural networks with imported knowledge from other related tasks.

Table 11: Times in seconds per run of EvoPruneDeepTL against efficient CNN pruning methods with one and two layers models.

	One Layer				**Two Layers**					
Dataset	Weight	Poly. Decay	Neuron	EvoPruneDeepTL	Weight	Poly. Decay	Neuron	EvoPruneDeepTL Layer 1	EvoPruneDeepTL Layer 2	EvoPruneDeepTL Both Layers
SRSMAS	1,995	2125	1,995	34,510	2,395	2,545	2,398	34,856	38,731	55,596
RPS	1,674	1,893	1,674	19,851	1,229	1,379	1,229	15,758	18,790	28,774
LEAVES	2,425	2,560	2,425	35,243	2,430	2,565	2,430	37,561	57,695	63,897
PAINTING	1,386	1,508	1,386	61,734	2,903	3,243	2,903	46,856	55,414	81,913
CATARACT	594	627	584	6,768	449	473	449	7,392	8,529	12,350
PLANTS	298	407	298	28,456	270	370	270	21,788	22,235	35,998

5.4. Answering RQ4: Feature Selection

The RQ4 establishes the dichotomy of choosing pruning or feature selection for the given problem. For that reason, in this section we analyze the FS model by conducting the same group of experiments of the previous sections, to compare it against EvoPruneDeepTL to decide which one scores best among them. The FS scheme is a particular case of EvoPruneDeepTL if only one fully-connected layer composes the configuration of the network and the pruning and GA are focused on the extracted features of the ResNet-50 model.

Table 12 shows the results for this model against the reference methods. This case follows the same similarities of the previous ones, as FS obtains the best average results for all the datasets.

Table 12: Average results for Fetaure Selection against non pruning methods.

Dataset	Measure	Not Pruned	Best Fixed	Feature Selection
SRSMAS	Accuracy % Active neur.	0.832 100	0.866 20	**0.884** 60
RPS	Accuracy % Active neur.	0.938 100	0.938 40	**0.985** 45
LEAVES	Accuracy % Active neur.	0.923 100	0.927 80	**0.943** 59
PAINTING	Accuracy % Active neur.	0.939 100	0.945 60	**0.958** 55
CATARACT	Accuracy % Active neur.	0.703 100	0.719 70	**0.747** 55
PLANTS	Accuracy % Active neur.	0.432 100	0.432 10	**0.472** 68

Similarly to the previous sections, we have also compared this model with the CNN pruning methods with only one layer, as shown in Table 13. In this case, in four out of six datasets the Feature Selection outperforms these methods, but in LEAVES and PAINTING Weight and Polynomial Decay perform better than our model.

Table 13: Average results of Feature Selection against efficient CNN pruning methods.

		One Layer			
Dataset	Measure	Weight	Poly. Decay	Neuron	Feature Selection
SRSMAS	Accuracy % Active neur.	0.841 60	0.878 60	0.802 60	**0.884** 60
RPS	Accuracy % Active neur.	0.913 45	0.926 45	0.869 45	**0.985** 45
LEAVES	Accuracy % Active neur.	**0.947** 59	0.940 59	0.946 59	0.943 59
PAINTING	Accuracy % Active neur.	0.962 55	**0.968** 55	0.883 55	0.958 55
CATARACT	Accuracy % Active neur.	0.696 55	0.689 55	0.687 55	**0.747** 55
PLANTS	Accuracy % Active neur.	0.421 68	0.317 68	0.402 68	**0.472** 68

In this section we have compared our FS scheme against reference models and efficient pruning methods published in the literature. The results shed light over the benefits of this model as it is also able to achieve a great performance over the reference models and also, in most cases, against the CNN pruning methods.

The global results for EvoPruneDeepTL and its different versions are presented in Table 14. The rows show the achieved accuracy and the percentage of improvement in relation to the best reference models for each model.

Table 14: Results and percentage of improvement for each version of EvoPruneDeepTL in relation to each reference model.

Dataset	Measure	Pruning Model One Layer	Pruning Model Both Layers	Feature Selection
SRSMAS	Accuracy	**0.885**	**0.885**	0.884
	% Improvement	1.9	2.2	1.8
RPS	Accuracy	0.954	0.978	**0.985**
	% Improvement	1.6	3.2	4.7
LEAVES	Accuracy	0.935	0.936	**0.943**
	% Improvement	0.8	0.2	1.6
PAINTING	Accuracy	0.951	0.953	**0.958**
	% Improvement	0.6	0.4	1.3
CATARACT	Accuracy	0.732	0.746	**0.747**
	% Improvement	1.3	1.1	2.8
PLANTS	Accuracy	0.480	**0.491**	0.472
	% Improvement	4.8	2.5	4.0

Reviewing the results of EvoPruneDeepTL and FS, we confirm that FS is the best model as it obtains the best accuracy levels in four out of six datasets. Furthermore, the pruning of both layers carried out by EvoPruneDeepTL also attains very notable performance levels.

Moreover, if we consider the optimization using the pruning model, the optimization of both layers yields the best results in terms of mean accuracy for each dataset. However, when comparing pruning and FS, the latter has more robust models: it achieves the best performance in four datasets and it is also shown in the improvement percentage for each dataset.

In conclusion, it is shown with empirical evidence that pruning can be done by optimizing the fully-connected layers, specifically, by evolving their neurons to get the fittest configuration that reports an improvement of the network performance. An evolutionary feature selection based on the extracted features also achieves a great network performance, both in improving the accuracy and in reducing its complexity.

6. Conclusions

This paper has introduced EvoPruneDeepTL, a novel model that sparsifies the architecture of the last layers of a DL model initialized using TL. EvoPruneDeepTL is a combination of sparse layers and EA, so that the neurons of these layers are pruned using the EA, in order to adapt them to the problem to tackle and deciding which neurons/connections to leave active or inactive.

EvoPruneDeepTL is a flexible model that optimizes models with one and two layers and even two layers at the same time. Our results show that the pruning over complete neurons is better than pruning connections individually, establishing the last one as the best encoding strategy. The evolution of the sparse layer improves these models in terms of accuracy and also in terms of complexity of the network. In comparison with compared reference models and pruning methods from the literature, EvoPruneDeepTL achieves a better performance than all of them. Finally, the decision of choosing pruning or the Feature Selection has been answered, as the FS scheme derived from EvoPruneDeepTL has shown a better performance, in most cases, than the pruning methods.

From an overarching perspective, this work aligns with a growing strand of contributions where evolutionary computation and DL have synergized together to yield optimized models that attain better levels of performance and/or an increased computational efficiency. Indeed, this fusion of concepts (forged as Evolutionary Deep Learning) has been used for other optimization processes, including hyperparameter or structural tuning. Another recent case of the symbiosis of EA's and DL are represented in AutoML-Zero that use an evolutionary search to automatically search the best DL structure. AutoML-Zero and Evo-PruneDeepTL are two great examples of the benefits of combining EA's and DL that outline the potential and promising path of successes envisioned for this research area.

Future research work stemming from the results reported in this study is planned from a two-fold perspective. To begin with, we plan to achieve larger gains from the combination of DL and EA by extending the evolutionary search over higher layers of the neural hierarchy, increasing the number of optimized layers and neurons per layer. To this end, we envision that exploiting the layered arrangement in which neurons are deployed along the neural architecture will be essential to ensure an efficient search. The second research line relates to this last thought, aiming to improve the search algorithm itself by resorting advanced concepts in evolutionary computation (e.g. niching methods or co-evolutionary algorithms).

Acknowledgments

F. Herrera, D. Molina and J. Poyatos are supported by the Andalusian Excellence project P18-FR-4961, the infrastructure project with reference EQC2018-005084-P and the R&D and Innovation project with reference PID2020-119478GB-I00 granted by the Spain's Ministry of Science and Innovation and European Regional Development Fund (ERDF). Aritz D. Martinez and Javier Del Ser would like to thank the Basque Government for the funding support received through the EMAITEK and ELKARTEK programs, as well as the Consolidated Research Group MATHMODE (IT1294-19) granted by the Department of Education of this institution.

References

Anwar, S., Hwang, K., Sung, W., 2017. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems 13, 1–18.

Assunçao, F., Lourenço, N., Machado, P., Ribeiro, B., 2019. Denser: deep evolutionary network structured representation. Genetic Programming and Evolvable Machines 20, 5–35.

Bäck, T., Fogel, D.B., Michalewicz, Z. (Eds.), 1997. Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol, UK.

Chambers, L., 2000. The Practical Handbook of Genetic Algorithms: Applications, Second Edition. The Practical Handbook of Genetic Algorithms: Applications, Chapman and Hall/CRC.

Choi, S., 2020. Cataract dataset. Available from https://www.kaggle.com/jr2ngb/cataractdataset.

Elsken, T., Metzen, J.H., Hutter, F., 2019a. Efficient multi-objective neural architecture search via lamarckian evolution, in: International Conference on Learning Representations, pp. 1–23.

Elsken, T., Metzen, J.H., Hutter, F., 2019b. Neural architecture search: A survey. The Journal of Machine Learning Research 20, 1997–2017.

Fernandes, C., Rosa, A., 2001. A study on non-random mating and varying population size in genetic algorithms using a royal road function, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), IEEE. pp. 60–66.

Frankle, J., Carbin, M., 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9.

Goldberg, D., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA .

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A., Herrera, F., 2019. Coral species identification with texture or structure images using a two-level classifier based on convolutional neural networks. Knowledge-Based Systems , 104891.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A., Krawczyk, B., Herrera, F., 2019. Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation. Expert Systems with Applications 118, 315 – 328.

Han, S., Mao, H., Dally, W., 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, in: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, pp. 1–14.

Han, S., Pool, J., Tran, J., Dally, W., 2015. Learning both weights and connections for efficient neural network, in: Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 1135 – 1143.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

Iba, H., 2018. Evolutionary Approach to Machine Learning and Deep Neural Networks: Neuro-Evolution and Gene Regulatory Networks. Springer.

Iguyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182.

Jung, S., Park, J., Lee, S., 2019. Polyphonic sound event detection using convolutional bidirectional lstm and synthetic data-based transfer learning, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 885–889.

Khan, S., Islam, N., Jan, Z., Din, I.U., Rodrigues, J.J.C., 2019. A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. Pattern Recognition Letters 125, 1–6.

Kokiopoulou, E., Hauth, A., Sbaiz, L., Gesmundo, A., Bartók, G., Berent, J., 2020. Task-aware performance prediction for efficient architecture search, in: ECAI 2020. IOS Press, pp. 1238–1245.

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

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

Lee, H., Pham, P., Largman, Y., Ng, A.Y., 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks, in: Advances in Neural Information Processing Systems 22, Curran Associates, Inc.. pp. 1096–1104.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K., 2018a. Progressive neural architecture search, in: ECCV.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K., 2018b. Hierarchical representations for efficient architecture search.

Liu, J., Wang, Y., Qiao, Y., 2017a. Sparse deep transfer learning for convolutional neural network, in: 31st AAAI Conference on Artificial Intelligence, AAAI 2017, pp. 2245–2251.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F., 2017b. A survey of deep neural network architectures and their applications. Neurocomputing 234, 11–26.

Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T., 2019. Rethinking the value of network pruning, in: 7th International Conference on Learning Representations, ICLR 2019.

Long, X., Ben, Z., Liu, Y., 2019a. A survey of related research on compression and acceleration of deep neural networks. Journal of Physics: Conference Series 1213, 052003.

Long, X., Ben, Z., Zeng, X., Liu, Y., Zhang, M., Zhou, D., 2019b. Learning sparse convolutional neural network via quantization with low rank regularization. IEEE Access 7, 51866–51876.

Lu, Z., Sreekumar, G., Goodman, E., Banzhaf, W., Deb, K., Boddeti, V., 2021. Neural architecture transfer. IEEE Transactions on Pattern Analysis & Machine Intelligence 43, 2971–2989.

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W., 2019. Nsga-net: Neural architecture search using multi-objective genetic algorithm, pp. 419–427.

Luo, J.H., Wu, J., Lin, W., 2017. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression, in: IEEE International Conference on Computer Vision, pp. 5068–5076.

Mantzaris, D., Anastassopoulos, G., Adamopoulos, A., 2011. Genetic algorithm pruning of probabilistic neural networks in medical disease estimation. Neural Networks 24, 831 – 835.

Martín, A., Lara-Cabrera, R., Fuentes-Hurtado, F., Naranjo, V., Camacho, D., 2018. Evodeep: a new evolutionary approach for automatic deep neural networks parametrisation. Journal of Parallel and Distributed Computing 117, 180–191.

Martinez, A.D., Del Ser, J., Villar-Rodriguez, E., Osaba, E., Poyatos, J., Tabik, S., Molina, D., Herrera, F., 2021. Lights and shadows in evolutionary deep learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. Information Fusion 67, 161–194.

Masson, H., Bhuiyan, A., Nguyen-Meidine, L.T., Javan, M., Siva, P., Ayed, I.B., Granger, E., 2021. Exploiting prunability for person re-identification. EURASIP Journal on Image and Video Processing 2021, 22.

Mehdipour Ghazi, M., Yanikoglu, B., Aptoula, E., 2017. Plant identification using deep neural networks via optimization of transfer learning parameters. Neurocomputing 235, 228 – 235.

Mohapatra, R., Saha, S., Coello, C.A.C., Bhattacharya, A., Dhavala, S.S., Saha, S., 2021. Adaswarm: Augmenting gradient-based optimizers in deep learning with swarm intelligence. IEEE Transactions on Emerging Topics in Computational Intelligence , 1–12.

Moroney, L., 2019. Rock, paper, scissors dataset. URL: http://www.laurencemoroney.com/rock-paper-scissors-dataset/.

Muhammad, K., Khan, S., Ser, J.D., Albuquerque, V.H.C.d., 2021. Deep learning for multigrade brain tumor classification in smart healthcare systems: A prospective survey. IEEE Transactions on Neural Networks and Learning Systems 32, 507–522.

Muhammad, K., Ullah, A., Lloret, J., Ser, J.D., de Albuquerque, V.H.C., 2020. Deep learning for safe autonomous driving: Current challenges and future directions. IEEE Transactions on Intelligent Transportation Systems , 1–21.

Museum, V.R., 2018. Art images: Drawing/painting/sculptures/engravings. Available from https://www.kaggle.com/thedownhill/art-images-drawings-painting-sculpture-engraving.

Nibali, A., He, Z., Wollersheim, D., 2017. Pulmonary nodule classification with deep residual networks. International journal of computer assisted radiology and surgery 12, 1799–1808.

Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H.G., Ogata, T., 2015. Audio-visual speech recognition using deep learning. Applied Intelligence 42, 722–737.

Pan, S., Yang, Q., 2010. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359.

Rauf, H.T., Saleem, B.A., Lali, M.I.U., Khan, M.A., Sharif, M., Bukhari, S.A.C., 2019. A citrus fruits and leaves dataset for detection and classification of citrus diseases through machine learning. Data in Brief 26, 104340.

Real, E., Aggarwal, A., Huang, Y., Le, Q.V., 2019. Regularized evolution for image classifier architecture search, in: Proceedings of the AAAI conference on artificial intelligence, pp. 4780–4789.

Real, E., Liang, C., So, D., Le, Q., 2020. Automl-zero: evolving machine learning algorithms from scratch, in: International

Conference on Machine Learning, PMLR. pp. 8007–8019.
Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V., Kurakin, A., 2017. Large-scale evolution of image classifiers, in: Proceedings of the 34th International Conference on Machine Learning, pp. 2902–2911.
Roy, D., Murty, K.S.R., Mohan, C.K., 2015. Feature selection using deep neural networks, in: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–6.
Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., McCool, C., 2016. Deepfruits: A fruit detection system using deep neural networks. Sensors 16, 1222.
Salehinejad, H., Valaee, S., 2021. Edropout: Energy-based dropout and pruning of deep neural networks. IEEE Transactions on Neural Networks and Learning Systems , 1–14.
Samala, R., Chan, H.P., Hadjiiski, L., Helvie, M., Richter, C., Cha, K., 2018. Evolutionary pruning of transfer learned deep convolutional neural network for breast cancer diagnosis in digital breast tomosynthesis. Physics in Medicine and Biology 63, 095005.
Scott, G.J., England, M.R., Starms, W.A., Marcum, R.A., Davis, C.H., 2017. Training deep convolutional neural networks for land–cover classification of high-resolution imagery. IEEE Geoscience and Remote Sensing Letters 14, 549–553.
Shin, H.C., Roth, H., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., Summers, R., 2016. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. IEEE Transactions on Medical Imaging 35, 1285–1298.
Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., Batra, N., 2020. Plantdoc: A dataset for visual plant disease detection, in: 7th ACM IKDD CoDS and 25th COMAD, p. 249–253.
Srinivas, S., Babu, R.V., 2015. Data-free parameter pruning for deep neural networks, in: Proceedings of the British Machine Vision Conference (BMVC), BMVA Press. pp. 31.1–31.12.
Stanley, K., Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evolutionary Computation 10, 99–127.
Sultana, F., Sufian, A., Dutta, P., 2018. Advancements in image classification using convolutional neural network, in: 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), pp. 122–129.
Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J., 2016. Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE Transactions on Medical Imaging 35, 1299–1312.
Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A survey on deep transfer learning, in: International conference on artificial neural networks, Springer. pp. 270–279.
Ullrich, K., Welling, M., Meeds, E., 2017. Soft weight-sharing for neural network compression, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, pp. 1–16.
Wan, S., Liang, Y., Zhang, Y., 2018. Deep convolutional neural networks for diabetic retinopathy detection by image classification. Computers and Electrical Engineering 72, 274–282.
Wang, Z., Li, F., Shi, G., Xie, X., Wang, F., 2020. Network pruning using sparse learning and genetic algorithm. Neurocomputing 404, 247 – 256.
Weiss, K., Khoshgoftaar, T.M., Wang, D., 2016. A survey of transfer learning. Journal of Big data 3, 1–40.
Yildirim, O., Pławiak, P., Tan, R.S., Acharya, U., 2018. Arrhythmia detection using deep convolutional neural network with long duration ecg signals. Computers in Biology and Medicine 102, 411–420.
Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks?, in: Advances in Neural Information Processing Systems, pp. 3320–3328.
Zhou, X., Gong, W., Fu, W., Du, F., 2017. Application of deep learning in object detection, in: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), IEEE. pp. 631–634.
Zhu, M., Gupta, S., 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression, in: 6th International Conference on Learning Representations (ICLR).
Zoph, B., Vasudevan, V., Shlens, J., Le, Q., 2018. Learning transferable architectures for scalable image recognition, pp. 8697–8710.
Zou, Q., Ni, L., Zhang, T., Wang, Q., 2015. Deep learning based feature selection for remote sensing scene classification. IEEE Geoscience and Remote Sensing Letters 12, 2321–2325.