applied sciences

MDPI

Article

Neuroplasticity-Based Pruning Method for Deep Convolutional Neural Networks

Jose David Camacho, Carlos Villaseñor, Carlos Lopez-Franco and Nancy Arana-Daniel *

Department of Computer Science, University of Guadalajara, 1421 Marcelino García Barragán, Guadalajara 44430, Jalisco, Mexico; camacho.castillo.jdavid@gmail.com (J.D.C.); carlos.villasenor@academicos.udg.mx (C.V.); carlos.lfranco@academicos.udg.mx (C.L.-F.)
* Correspondence: nancy.arana@academicos.udg.mx

Abstract: In this paper, a new pruning strategy based on the neuroplasticity of biological neural networks is presented. The novel pruning algorithm proposed is inspired by the knowledge remapping ability after injuries in the cerebral cortex. Thus, it is proposed to simulate induced injuries into the network by pruning full convolutional layers or entire blocks, assuming that the knowledge from the removed segments of the network may be remapped and compressed during the recovery (retraining) process. To reconnect the remaining segments of the network, a translator block is introduced. The translator is composed of a pooling layer and a convolutional layer. The pooling layer is optional and placed to ensure that the spatial dimension of the feature maps matches across the pruned segments. After that, a convolutional layer (simulating the intact cortex) is placed to ensure that the depth of the feature maps matches and is used to remap the removed knowledge. As a result, lightweight, efficient and accurate sub-networks are created from the base models. Comparison analysis shows that in our approach is not necessary to define a threshold or metric as the criterion to prune the network in contrast to other pruning methods. Instead, only the origin and destination of the prune and reconnection points must be determined for the translator connection.

Keywords: pruning; neuroplasticity; deep learning; convolutional layers; transfer learning

Citation: Camacho, J.D.; Villaseñor, C.; Lopez-Franco, C.; Arana-Daniel, N. Neuroplasticity-Based Pruning Method for Deep Convolutional Neural Networks. *Appl. Sci.* **2022**, *12*, 4945. https://doi.org/10.3390/app12104945

Academic Editors: Tan-Hsu Tan, Mohammad Alkhaleefah and Yang-Lang Chang

Received: 20 April 2022
Accepted: 11 May 2022
Published: 13 May 2022

1. Introduction

Deep convolutional neural networks have shown an excellent performance in computer vision tasks such as classification [1,2], object detection [3–6], semantic segmentation [7–9], reconstruction [10], and many others. However, it is well known that the performance of these models is strongly related to how deep and wide is the architecture [11]; consequently, these models usually require a considerable amount of memory and specialized hardware for their training and inference steps. Therefore, one of the key obstacles for convolutional neural networks applications is the overparameterization, which induces knowledge redundancy over the network layers.

In recent years, network compression techniques [12] have been developed to reduce the number of parameters and the complexity of networks. Some compression techniques such as low-rank approximation [13,14], parameter quantization [15,16], and parameter binarization [17] have shown efficient compression rates and desirable accuracy. On the other hand, network pruning [18–24] is another popular compression technique that has achieved excellent compression rates and accuracy. Furthermore, these techniques usually present a straightforward implementation, and they can be used alongside other compression techniques.

The network pruning methods are designed to remove those unimportant connections that generate redundancy and can be removed without drastically affecting the network performance [25]. Nevertheless, most of these compression techniques establish an iterative process of pruning and retraining as the strategy to compensate for the accuracy loss. Therefore, the pruning process often requires extensive retraining periods, and the compression has to be applied gradually to avoid a drastic impact on the performance.

Neuroplasticity (also known as neural plasticity) is the ability of biological neural networks to create, modify and reorganize their neural connections. This adaptability has been studied in different applications [26], such as neural synaptic plasticity, learning and memory, homeostasis, sensory training, and stroke recovery. In addition, one of the principal neuroplasticity research is the study of synaptic restructuring as part of the experience acquired over time, which has shown that through the experience obtained, neural structures are reorganized to create simple and more efficient connections.

In biological neural networks, knowledge compression is derived from the experience acquired; for example, the motor cortex produces reorganizations in the neural connections by dominating motor tasks [27]. Analogous to this, the iterative process of pruning and retraining may be regarded as the specialization phase for a pre-trained artificial neural network; thus, a restructuring phase for the neural connections takes place to compress the network.

After a stroke, a person may present motor functions deficit, visual disabilities, facial paralysis, or numbness in the extremities. However, there are studies [28,29], on stroke survivors which suggest that during the recovery process, there is a time interval where the peri-infarct cortex may play a crucial role in the recovery of motor functions. In this way, knowledge remapping [30] is possible due to redundancy and the fuzzy neural structures, which can be removed or relocated in other intact cortex zones.

In the computational neuroscience field, increasing attention has been paid to studying the role of neural plasticity on the learning performance for accomplishing a machine learning task. For instance, Chrol-Cannon et al. [31], shares a review of different and important neural plasticity models designed to understand memory and cognition, learning, and self-organizing structural mechanisms, aiming to replicate them in artificial neural networks.

On the other hand, in visual neuroscience, object recognition in humans and non-humans primates is commonly associated with the neural plasticity activated by experience acquired over time. In this process, it has been hypothesized that specific network regions tend to present a strong activation with familiar and known objects. Therefore, the visual experience produces dramatic local changes in the response of the visual cortex. However, Op de Beeck et al. [32] suggested a different neural plasticity function, where the visual experience produces moderate and relatively distributed changes in the visual cortex, which modulate a pre-existing, rich, and flexible set of neural object representations. Extrapolating the above assumption to the artificial counterpart, moderate and distributed changes to the network kernels in convolutional neural networks should be applied. Therefore, the pre-existing object representations (feature maps) will be modulated according to the new visual experience (recognition task) acquired.

In this paper, a new pruning strategy based on the neuroplasticity of biological neural networks is presented. The pruning algorithm is inspired by the knowledge remapping ability after injuries in the cerebral cortex [33]. Thus, it is proposed to simulate induced injuries into the network by pruning full convolutional layers or entire blocks, assuming that the knowledge from the removed segments of the network may be remapped and compressed during the recovery (retraining) process. To reconnect the remaining segments of the network, a translator block is introduced. The translator is composed of a pooling layer and a convolutional layer. The pooling layer is optional and placed to ensure that the spatial dimension of the feature maps matches across the pruned segments. After that, a convolutional layer (simulating the intact cortex) is placed to ensure that the depth of the feature maps matches and is used to remap the removed knowledge.

This paper is organized as follows: In Section 2, there is a review of the related state-of-the-art works from artificial neural network pruning techniques. Then, in Section 3, our bio-inspired pruning method is presented. After that, in Section 4, the proposed pruning method is tested over popular pre-trained convolutional architectures. Subsequently, in Section 5, the experimental results are reported.Finally, in Section 6, conclusions and future work are discussed.

2. Related Work

Two different classes of pruning methods can be identified in the state-of-the-art: unstructured and structured.

Unstructured methods are based on pruning weights within a given threshold, which successfully reduces the number of parameters. However, they generate sparse models and often require extensive retraining to regain accuracy. For instance, Han et al. [18,34] proposed pruning those weights that have small magnitudes on AlexNet [35] and VGG [36] architectures. It is assumed that weights with small magnitudes do not contribute to relevant transformations for the generated output. Therefore, after pruning, the network is lightweight but requires retraining to compensate for the accuracy loss.

On the other hand, structured methods are designed to remove substructures, for example, kernels, filters, and layers. By removing entire structures, the parameters are reduced and the computational complexity is decreased. Hu et al. [19], introduced the average percentage of zeros (APoZ) criterion, which suggests that if most of the activations from a calculated feature map are zero, then the neuron can be pruned without drastically affecting the accuracy. This pruning method is data-driven; therefore, the results may change according to the dataset used.

Li et al. [21] proposed using the L1-norm as the criterion to remove unimportant filters. Thus, a percentage of filters that can be removed based on their sensitivity is calculated for each layer. Then the pruning strategy is to remove *smaller* filters because it is probable that those filters will generate feature maps with weak activations; in other words, with no relevant abstractions from the input data.

Wu et al. [24] introduced a pruning method based on the feature extraction measurement ability (FEAM), a combination of both practical and theoretical analysis. First, the L1-norm is calculated for each kernel, which quantifies the kernels' abilities to extract features. This ability is acquired during training, and it is called kernel dispersion. After that, the feature dispersion is calculated, a data-driven quantifier of the kernel's ability to extract relevant features from images. Therefore, the pruning criterion is the FEAM score, calculated as the product of the measured abilities. Then, the pruning process is iteratively applied, removing kernels with lower FEAM scores followed by a retraining phase to compensate for the accuracy loss.

3. Neuroplasticity-Based Pruning Method

The proposed method is designed to compress networks by pruning entire layers or blocks. Thus, layers/blocks are removed from the network, and the remaining substructures are reconnected by a translator block, where the knowledge will be remapped.

3.1. Pruning

A VGG16 architecture [36] is shown in Figure 1, displaying with blue rectangles the regular convolutional layers. Meanwhile, the yellow ones are convolutions followed by pooling layers. These blue and yellow rectangles illustrate those layers that are candidates to prune. On the other hand, the top of the network is summarized as one green block because, in this work, it is not considered to prune the fully connected layers but only the convolutional ones.

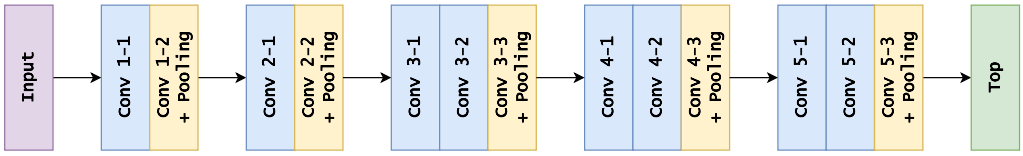

Figure 1. VGG16 architecture, illustrating the convolutional layers and blocks of the network.

To compress the network by pruning sub-structures, it is proposed to select an origin layer A and a forward destination layer B. All the layers between A and B will be pruned, and the remaining sub-structures must be reconnected. For instance, in Figure 2,

it can be observed all the possible connections addressed to the last convolutional layer of the network, where each blue dashed arrow stands for a new sub-model generated after pruning.

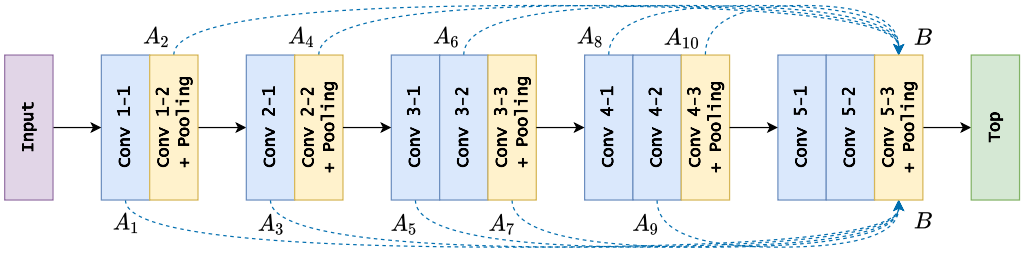

Figure 2. Example of some sub-models that can be generated using the proposed pruning method on a VGG16 architecture, considering different layers as the origin (A_i) and the last convolutional layer as the destination (B).

As it can be seen in Figure 2, the destination layer is fixed; however, by changing the destination layer, new sub-models can be generated. A heuristic for selecting the destination layer is deduced in the experiments section. The experimental results suggested using the last sub-structures from the network as the destination. Thus, the generated sub-models are lightweight, and the performance is similar to that achieved by base models.

Most of the time, the origin layer's output dimension will miss-match the destination layer's input dimension. Therefore, to successfully reconnect the origin and destination layers, the output from the origin layer needs a transformation before it passes to the destination layer. That is the reason for including a translator block presented in the following sub-section.

3.2. Translator Block

The pruned layers left sub-structures that must be reconnected; therefore, to reconnect the remaining layers a new block called translator is introduced, which can be observed in Figure 3.

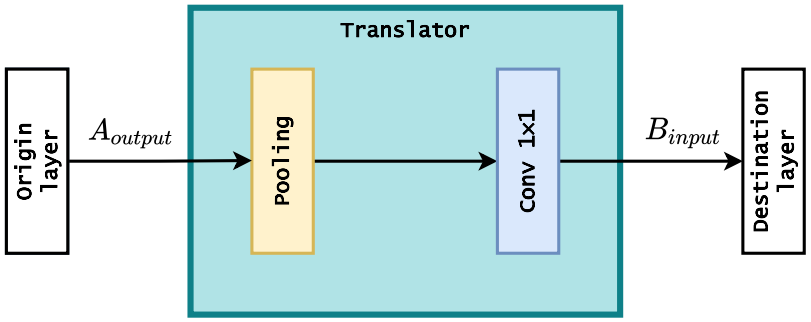

Figure 3. Translator block, designed to reconnect the pruned sub-structures.

First, it is proposed to use a pooling layer to ensure that the spatial dimension matches across the pruned segments. A pooling layer is preferred over a convolutional one since it allows projecting feature maps without creating new learnable parameters on the network. If the pruned sub-structures match the spatial dimensions, thus the pooling layer is not needed, and the computational complexity is decreased.

If the pooling layer is required, it must be created with an adequate pool size and stride to project the feature maps between the removed segments.

After the pooling layer, it is proposed to use a 1×1 convolutional layer, which is placed for two reasons. First, this convolutional layer is used to ensure that the number of features maps generated in layer A fits into the input of layer B. Moreover, even if the feature maps generated match in the pruned segments, the layer is placed to aggregate learnable parameters on the network.

The included parameters in the translator will be used to re-learn the removed knowledge from the pruned layers. Thus, analogous to biological neural networks, this layer can be regarded as the intact cortex that should compress the knowledge.

In Figure 4, there is an example of a VGG16 [36] pruned architecture, where it can be observed that blocks 3 and 4 are deleted, then the last layer from block 2 and the first layer from block 5 are used for the re-connection with the translator block.

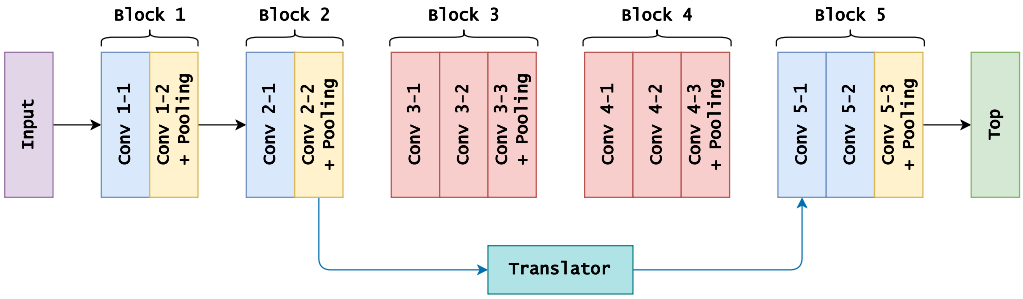

Figure 4. Example of VGG16 pruned architecture. Blocks 3 and 4 are pruned, thus, block 2 and 5 are reconnected with the translator block.

3.3. Retraining

After pruning sub-structures from the network and reconnecting the remaining ones with the translator, the network needs to be retrained to compensate for the accuracy loss.

Most state-of-the-art pruning methods apply an iterative pruning-retraining process, assuming the network connectivity may be harmed if relevant (important) connections are removed using a one-shot strategy, producing a drastic accuracy drop.

On the one hand, pruning-retraining iteratively reduces the parameters gradually and monitors the accuracy during the pruning process. On the other hand, the iterative process may require several pruning-retraining loops to achieve similar results as in the original models, which implies using more computational resources and time.

This work proposes adopting a one-shot pruning-retraining strategy because our method's key objective is not to reduce parameters gradually.

The one-shot process assumes that a group of P consecutive layers (pruned layers) can be removed and compressed into a new block T (translator) on the network. Thus, during retraining, the translator block learns a transformation for the feature maps that is equivalent to all the transformations produced by P, the pruned layers. This can be observed in Figure 5.

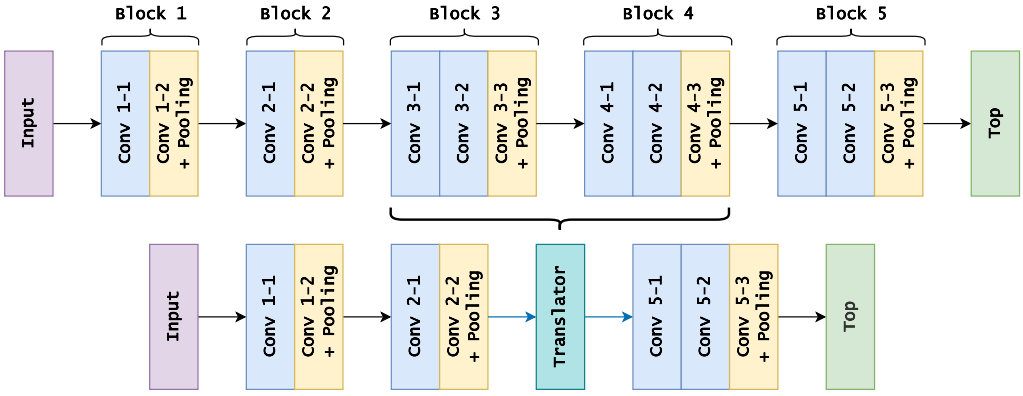

Figure 5. Example of replacing a group of layers P (blocks 3 and 4) for the translator block.

The knowledge from the removed sub-structures should be compressed using the learnable parameters introduced in the convolutional layer of the translator block. Thus, the layers in the network must be frozen while retraining (i.e., their parameters will not be

adapted during the retraining phrase), except for the convolutional layer in the translator. This can be observed in Figure 6.

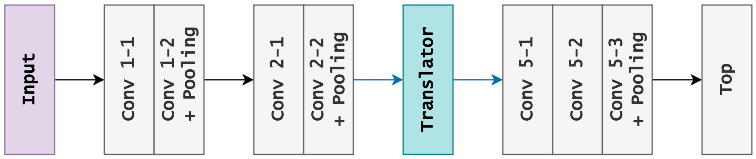

Figure 6. Example of VGG16 pruned architecture and reconnected with the translator block. The gray rectangles indicate that the layers are frozen, then the translator is the only block active for the retraining process.

Leaving the translator active during retraining allows recreating the knowledge in a specific network region. Most of the previously learned tasks will be remapped into a different (new in this case) region of the neural network. Similar to the biological counterpart, this process can be seen as the biological neural network recovery process after injury.

In our method, multiple origin and destination layers can be selected as candidates, which produce a combination of C possible sub-models where, for each sub-model, a group of layers P_i (with $i = 1, 2, ..., C$) will be pruned, and the model will be retrained once.

In contrast to other pruning methods, our pruning-retraining strategy generates different sub-models that may recover (or even improve) the base model's performance, using a single retraining step with a fixed number of epochs. Furthermore, each sub-model is different and non-dependent from the other generated models. Hence, the one-shot pruning-retraining process for each sub-model can be computed individually and even in a parallel model.

Using a one-shot pruning-retraining should be faster than other pruning methods, which apply the process iteratively until desired results are achieved or a maximum number of iterations are reached.

4. Experiments

An exhaustive creating-pruning-retraining process of the possible sub-models is carried out to explore the impact of pruning sub-structures in different parts of the networks. In addition, this exhaustive process allowed us to deduce a heuristic that let us know the best destination layer for the pruning process, as we will present in Section 4.4. This workflow is illustrated in Figure 7.

Figure 7. Workflow for the experiments.

The first three steps of the workflow are executed once because they generate the base models that will be compressed with the proposed pruning method. On the other hand, the last two steps are executed once for each sub-model that can be generated from the base models by selecting different layers as the origin and destination of the pruning process.

4.1. Pre-Trained Networks

In the experiments, first, a pre-trained network is selected. VGG16 [36], MobileNet [37], and MobileNetV2 [38] are used as the base architectures because they allow us to explore the impact of removing entire convolutional layers, including those with pooling, batch-norm, and residual connections.

The base networks were pre-trained on ImageNet [39], and they were downloaded using the TensorFlow [40] framework for Python.

4.2. Dataset Selection

After selecting the pre-trained networks, it is proposed to use different datasets to validate the replication of the results over the experiments. Therefore, the datasets used for this work are: road damage [41], flowers [42], and caltech-101 [43].

The datasets are partitioned into two subsets in the experiments, 70% of the image samples for training and 30% for testing.

4.3. Transfer Learning

It is proposed to apply transfer learning [44] to reuse the pre-trained networks with the previously mentioned datasets. The top of the networks are removed, and new ones are computed using the Hyperband search algorithm [45] included in the keras-tuner [46] python module.

The new top scheme for the architectures includes four layers: a depth-wise convolution with dropout (Conv dw + Dropout), a global average pooling (GAP), a dense layer with dropout, and a dense layer as the final output of the network. This proposed new top for the networks can be observed in Figure 8.

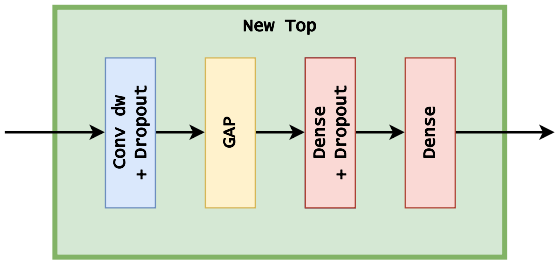

Figure 8. Proposed scheme for the new top of the architectures, included when transfer learning is applied.

The Hyperband's search space has four parameters: the dropout ratio (from 0 to 0.5) for the two layers, the number of neurons (from 4 to 64) for the first dense layer, and the learning rate (from 1×10^{-5} to 1×10^{-2}) for training. Thus, each base architecture has the same layers on top but with different configurations and learning rates.

Only the new top of the network is active during the hyperband execution. Thus, the network's backbone is frozen while the top learns to identify key abstractions from the computed feature maps. Furthermore, the search algorithm is executed with a limit of 30 epochs, looking for the model with the best testing accuracy.

In addition, the optimizer algorithm for training is sKAdam [47], a gradient-based optimization algorithm presented in a previous work by the authors.

4.4. Pruning

After applying the transfer learning process, three new models are generated for each selected dataset. These models are used as the base to create multiple sub-models, which are used to evaluate the compression ratio and performance obtained by skipping and pruning layers or blocks on the network.

To define the destination layer for re-connecting the remaining sub-structures, it is proposed to skip at least two layers forward from the origin pruning layer on the sub-models. Hence, the translator's efficiency is evaluated with the compression ability to recreate the removed knowledge from small sub-structures up to complex and deep sub-structures.

4.4.1. VGG16

The VGG16 [36] base architecture is presented in Figure 1. However, the top of the network is replaced with the previously computed according to the dataset.

Figure 2, illustrates an example of the sub-structures connections for the layers of the network, where each convolutional layer is a candidate to be an origin or destination layer.

Therefore, the possible re-connections between the layers generate 55 different sub-models for each dataset.

4.4.2. MobileNet

Figure 9 shows the base architecture for the MobileNet [37]. Each depth-wise separable convolution (DS-Conv) block is represented by an orange rectangle, which summarizes both convolutions with their respective batch-norm and activation function.

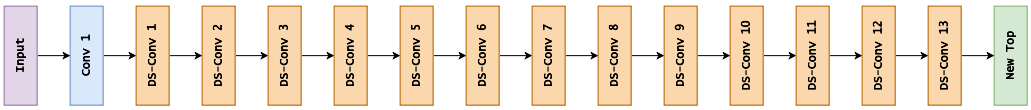

Figure 9. MobileNet architecture, illustrating the depth-wise separable convolutions.

The first convolutional layer and the depth-wise separable convolution are used as the candidates for the origin and destination layers. Therefore, the possible re-connections between the layers generate 78 different sub-models for each dataset.

4.4.3. MobileNetV2

Figure 10 shows the base architecture for the MobileNetV2 [38]. As it can be observed, the inverted bottlenecks are represented with orange rectangles as entire blocks. Furthermore, the adder layers are represented by yellow rectangles.

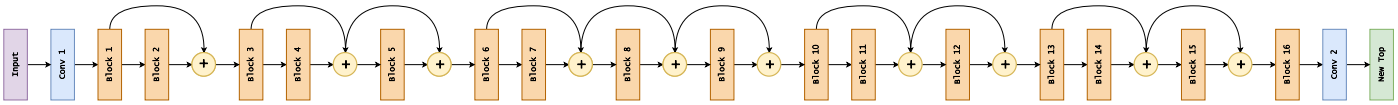

Figure 10. MobileNetV2 architecture, illustrating the inverted bottlenecks as blocks and their shortcut connections.

The translator block can be placed on different parts of the network, considering restrictions for the origin and destination layers due to the shortcut connections. First, the origin layer must be the first convolutional layer, an inverted bottleneck (if it does not have an adder layer next), or an adder layer. Then, only inverted bottlenecks and the last convolutional layer can be candidates for the destination layer.

Therefore, the possible re-connections between the layers generate 136 different sub-models for each dataset.

4.5. Retraining

This process is one-shot execution as the pruning phase. As mentioned before, during retraining, it is proposed that only the translator is active, then the other layers are frozen. Consequently, the acquired knowledge is encapsulated by the kernels of the translator.

The retraining can be interpreted as an extension of the first training, which began in the parameters search. Thus, the extension marks a transition in the network, from learning a new task to compressing the knowledge acquired. Therefore, it is proposed to retrain the sub-models for 50 epochs and use learning rates computed with the hyperband search.

5. Experimental Results

The pruning overall experimental results are reported in this section, illustrating the compression rates, performance, and best sub-models generated with the proposed pruning method. Furthermore, comparisons with other popular state-of-the-art pruning methods are presented.

5.1. VGG16

This section presents the experimental results obtained using our neuroplasticity pruning method with the base architecture of a VGG16 that has been trained with several datasets.

In Table 1, the performance and information of the architectures computed with the Hyperband [45] algorithm are presented. For each architecture, we report their number of parameters, floating-point operations per second (FLOPs), testing accuracy, and training accuracy.

The FLOPs are calculated using the keras-flops [48] python module.

Table 1. VGG16 architectures, computed using the Hyperband algorithm.

Dataset	Params	FLOPs	Testing Acc.	Train Acc.
Road Damage	14.7343 M	30.71 G	0.7929	0.8421
Flowers	14.7302 M	30.71 G	0.8571	0.8964
Caltech 101	14.7518 M	30.71 G	0.8821	0.9513

As it can be observed, the architectures have a similar number of parameters and FLOPs, and only the accuracy for training and testing varies because of the dataset used. New sub-models are created from these computed architectures, applying the proposed pruning method by removing entire network sub-structures.

5.1.1. Road Damage

In this subsection, experimental results for recognition of road damages using the images of the dataset [41], classified by VGG16 sub-models generated using the neuroplasticity-based pruning method, are presented.

In Figure 11, a total of 55 scatter points are plotted, where each point on the figure represents a generated sub-model after the one-shot pruning and retraining process.

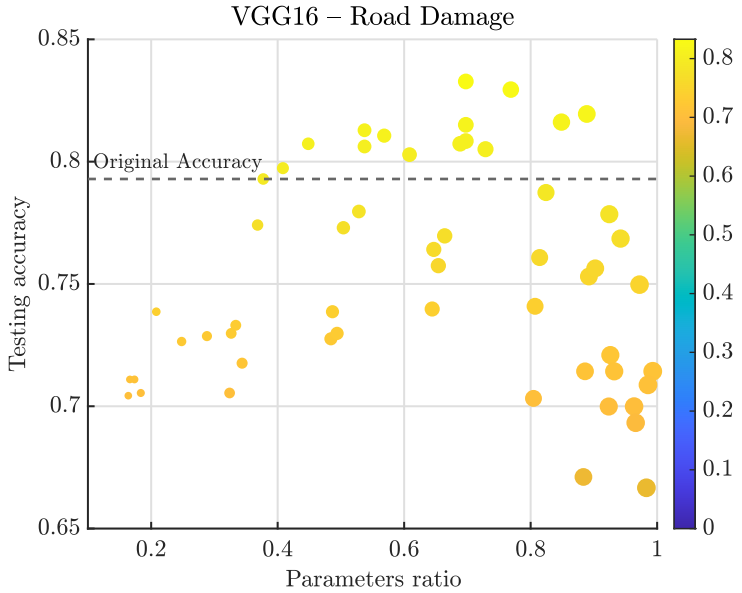

Figure 11. VGG16 sub-models trained on the road damage dataset.

In the above figure, the horizontal-axis represents the params ratio, which is calculated as the relation between the number of params from a generated sub-model and the number of params from the base model. On the other hand, the vertical-axis represents the testing accuracy of the model after retraining. It is important to note that a dotted line has been included to show the accuracy reached using the original model of the VGG16 network (labeled as original accuracy). In addition, the size of the scatter points changes according to the parameters ratio (smaller points for smaller parameters ratio, bigger points for higher parameters ratio), and their color is assigned by the testing accuracy reached.

In Figure 11, it can be observed that different sub-models reached a higher accuracy than the original network. Most of these sub-models have a params ratio between 0.6 and

0.4; thus, our proposed pruning method achieves a network compression of 40% up to 60% without losing accuracy and outperforming the accuracy reached by the base model.

Table 2 shows the performance and information of the top five generated sub-models, selected according to their accuracy reached and params ratio. In the first column of Table 2, the names of the models are shown. The first part of the name indicates the origin layer of the pruning, and the second part of the name stands for the destination layer of the pruning.

Table 2. VGG16 Top-5 Road Damage sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
conv4-1 to conv5-3	**0.3772**	21.57 G (70.24%)	0.7929	0.8113
conv3-2 to conv5-2	0.4083	**16.89 G (55.00%)**	0.7973	0.7819
conv3-3 to conv5-2	0.4484	20.59 G (67.05%)	0.8073	0.8359
conv4-2 to conv5-3	0.5374	25.27 G (82.29%)	0.8062	0.8502
conv4-1 to conv5-2	0.5374	22.49 G (73.23%)	**0.8128**	**0.8905**

As it can be seen in Figure 11 and Table 2, the Top five selected sub-models are those that presents excellent testing accuracy levels and lower params ratio. However, the experimental results are achieved by pruning and retraining (only the translator) once. Thus, if fine-tuning is applied is likely to increase the accuracy reached.

For a better illustration of the experimental results produced by pruning and reconnecting different sub-structures, in Figure 12 two heatmaps are presented. On each heatmap, there is a grid where the performance of the sub-models is mapped over an indicator.

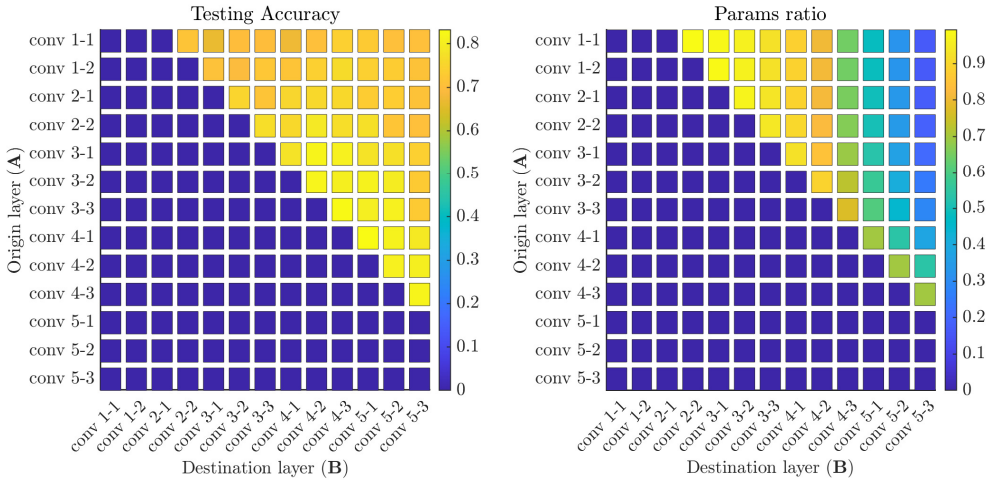

Figure 12. VGG16 Road Damage sub-models heatmaps.

Figures 11 and 12 shows that most sub-models achieved competitive accuracy values, which means that even those models with lower accuracy present a suitable performance. For instance, there is a sub-model with a translator connection between the layers Conv 3-1 and Conv 5-3. This sub-model has an accuracy of 0.7386 and a params ratio of 0.2081. Hence, when fine-tuning is applied is probable that this model will outperform the original accuracy, and only using close to 20% of the original params from the base model.

5.1.2. Flowers

In this subsection, experimental results for recognition of flowers using the images of the dataset [42], classified by VGG16 sub-models generated using the neuroplasticity-based pruning method, are presented.

In Figure 13, the sub-models generated from the VGG16 trained on the flowers dataset are displayed.

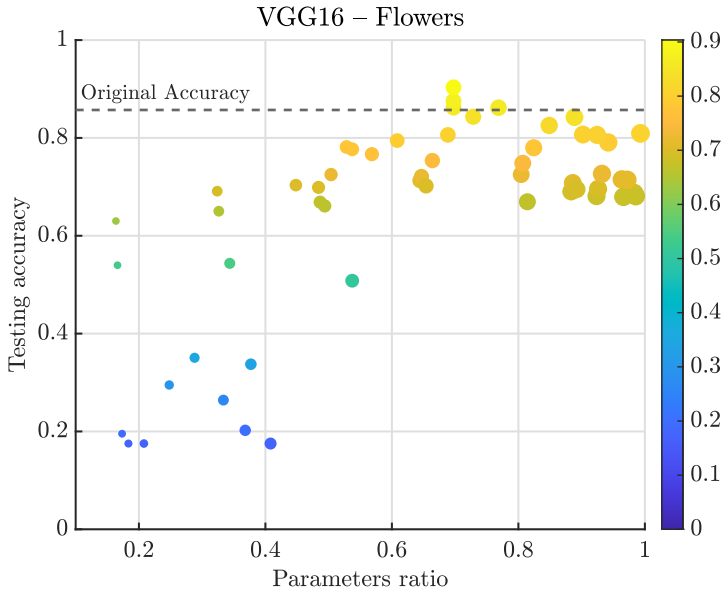

Figure 13. VGG16 sub-models trained on the flowers dataset.

Once again, it can be observed that some sub-models achieved higher accuracy than the original network, which is illustrated by the dotted line in Figure 13.

Table 3 shows the performance and information of the top five generated sub-models, selected according to their accuracy reached and params ratio. Once more, the models are named using the first part of the name as the origin layer, and the second part of the name refers to the destination layer of the pruning.

Table 3. VGG16 Top-5 Flowers sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
conv2-2 to conv5-1	**0.5038**	**12.24 G (39.86%)**	0.7251	0.6863
conv3-1 to conv5-1	0.5283	14.11 G (45.95%)	0.7815	0.7502
conv3-3 to conv5-1	0.6084	21.52 G (70.07%)	0.7946	0.8120
conv3-1 to conv4-3	0.6885	17.97 G (58.52%)	0.8062	0.8590
conv4-3 to conv5-3	0.6974	28.97 G (94.33%)	**0.9035**	**0.9428**

From the top-accuracy generated sub-models, it can be observed that connections between the last layers of the network are relevant. In fact, from the heatmaps presented in Figure 14 for this experiment, it is evident that not only the models from the last two columns are those with low params ratio but also with low accuracy.

In the testing accuracy heatmap, we can observe an accuracy drop from models with connections to the convolutional layer 5-2, which may suggest that up to the convolutional layer 5-1, there are substantial feature maps that were pruned from the network. In other words, relevant transformations to the input volume are lost by pruning specific sub-structures of the network, which may cause an undesired performance. Furthermore, the sub-models with connections to the convolutional layer 5-3 present a similar behavior, then multiple sub-models have an accuracy drop by skipping and pruning some sub-structures of the network.

In Figures 12 and 14, there is consistent that sub-models with desired performances (i.e., accuracy values close to or above 0.8) and non-significant accuracy loss are those with params ratio close to 0.6, as the top-accuracy sub-models reported in Table 3.

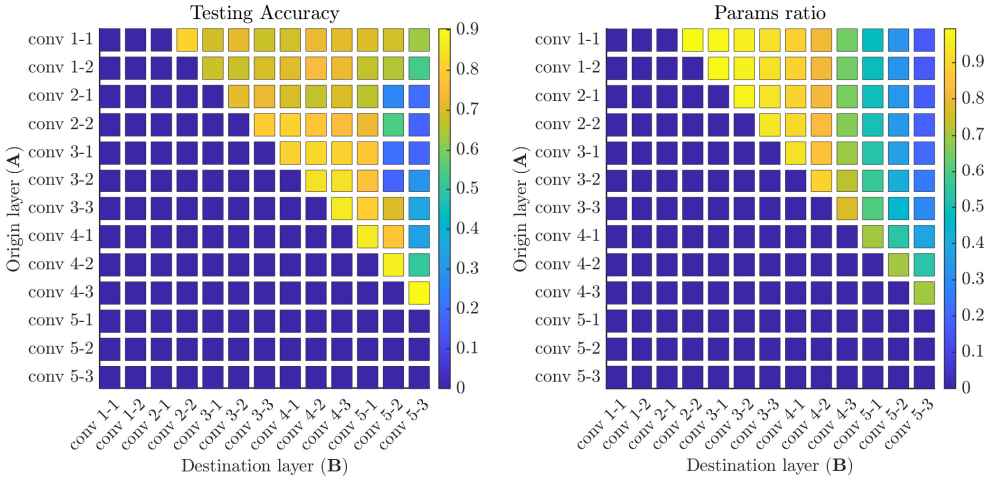

Figure 14. VGG16 Flowers sub-models heatmaps.

The above may suggest that experimentally and for this particular architecture, the compressed models with params ratio closer to 0.6 may lead to the balance between the accuracy levels and the compression rates. Furthermore, the models with parameters ratios lower than 0.6 are those that create connections between the last layers of the network. Therefore, an experimental criterion to prune networks while using our proposed method is to generate connections between the latter layers/block of the network, to create lightweight and efficient networks.

5.1.3. Caltech 101

In this subsection, experimental results for recognition of 101 different objects using the images of the dataset [43], classified by VGG16 sub-models generated using the neuroplasticity-based pruning method, are presented.

In Figure 15, the sub-models generated from the VGG16 trained on the caltech-101 dataset are shown.

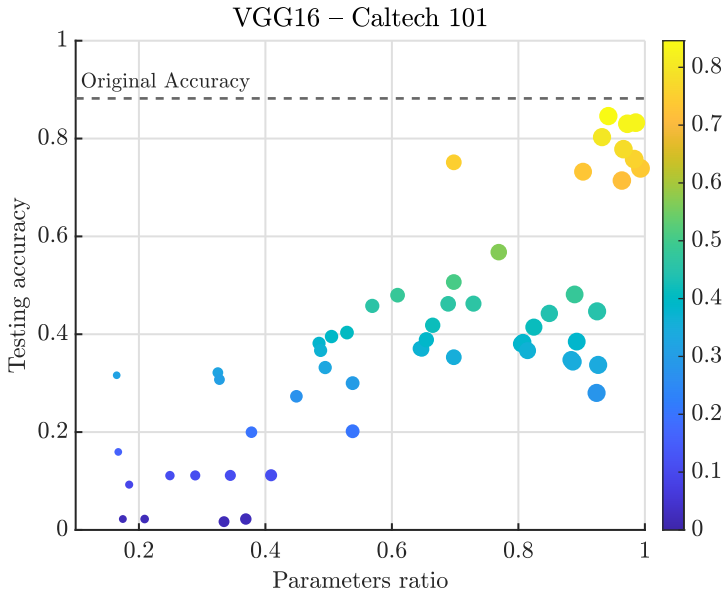

Figure 15. VGG16 sub-models trained on the caltech-101 dataset.

This time, just a few models achieved accuracy levels close to the accuracy reached by the original network (represented by the dotted line). Most of these models do not present the hoped params ratio, and the compression is marginal, which may happen because the

Caltech-101 dataset is unbalanced [49]. Then, the necessary knowledge to identify a specific class might be sparse all over the network.

Therefore, by removing sub-structures, it is possible that instead of reducing redundancy, we were deleting sparse significant knowledge. In other words, relevant feature maps transformations probably were pruned from the sub-structures. Thus, in such a case, during retraining is complicated to recover the accuracy since only the translator is active and learning.

Table 4 shows the performance and information on the top five generated sub-models, selected according to their accuracy reached and params ratio. Once more, the models are named using the first part of the name as the origin layer, and the second part of the name refers to the destination layer of the pruning.

Table 4. VGG16 Top-5 Caltech-101 sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
conv4-3 to conv5-3	**0.6979**	28.97 G (94.33%)	0.7514	**0.9682**
conv3-3 to conv4-3	0.7690	25.37 G (82.61%)	0.5678	0.8042
conv2-2 to conv4-1	0.9022	**21.51 G (70.04%)**	0.7322	0.7835
conv2-1 to conv3-3	0.9322	21.67 G (70.56%)	0.8029	0.8918
conv2-2 to conv3-3	0.9422	25.37 G (82.61%)	**0.8459**	0.9672

Even when most of the generated sub-models have an undesired performance, the translator connection between the convolutional layers 2-2 and 5-1 presents a decent performance and a parameters compression close to 30%. Thus, that model is a candidate to recover the accuracy using fine-tuning.

Figure 16 shows the heatmaps from the sub-models trained on the Caltech-101 dataset. In the heatmaps, it can be observed that some sub-models with connections between the first layers of the network are those which present the hoped accuracy values. The above may support that, for this particular dataset and architecture, the key transformations for the inputs lie in the first layer of the network. Therefore, if these relevant sub-structures are removed, the accuracy is dropped, and it is difficult for the translator to recover the performance.

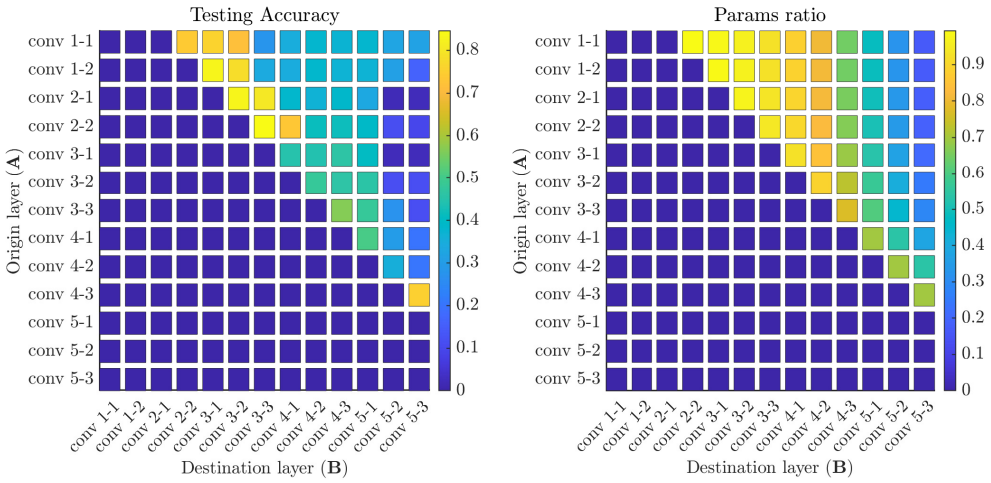

Figure 16. VGG16 Caltech-101 sub-models heatmaps.

From the VGG16 experiments, we observed that excellent params ratios are achieved most of the time. As well as outperform the accuracy of the original models. In other words, the sub-models generated by pruning and retraining present efficient performance as in the original model but using up to 60% to 20% of the number of params from the base model.

On the one hand, the experimental results are generated by pruning and retraining in a one-shot strategy. Thus, fine-tuning is likely to increase the accuracy levels reached. On the other hand, only the translator is active and acquiring knowledge during retraining, which means that only the translator is trying to re-learn the transformations produced by the pruned sub-structures. Therefore, our pruning-retraining method is as simple as retraining when it is used over a sub-structure already identified as redundant.

Furthermore, the undesired performance may be caused by the layered approach used to prune sub-structures. Therefore, in the following experiments, more complex sub-structures will be selected as candidates to be pruned.

5.2. MobileNet

This section presents the experimental results obtained using our neuroplasticity pruning method with the base architecture of a MobileNet that has been trained with several datasets.

In Table 5, the performance and information of the architectures computed with the Hyperband [45] algorithm are presented. For each architecture we report on their number of parameters, floating-point operations per second (FLOPs), testing accuracy, and training accuracy.

The FLOPs are calculated using the keras-flops [48] python module.

Table 5. MobileNet architectures, computed using the Hyperband algorithm.

Dataset	Params	FLOPs	Testing Acc.	Train Acc.
Road Damage	3.2679 M	1.14 G	0.8328	0.8601
Flowers	3.2597 M	1.14 G	0.8973	0.9530
Caltech 101	3.2707 M	1.14 G	0.9266	0.9993

As it can be observed, the architectures have a similar number of parameters and FLOPs, and only the accuracy for training and testing varies because of the dataset used. New sub-models are created from these computed architectures, applying the proposed pruning method by removing entire sub-structures of the network.

5.2.1. Road Damage

In this subsection, experimental results for recognition of road damages using the images of the dataset [41], classified by MobileNet sub-models generated using the neuroplasticity-based pruning method, are presented.

For this experiment, instead of using a layered approach as the criterion to select candidates for pruning, it is proposed to use the depth-wise separable convolution blocks as the origin and destination sub-structures.

In Figure 17, a total of 78 scatter points are plotted, where each point on the figure represents a generated sub-model after the one-shot pruning and retraining process.

In the above figure, the horizontal-axis represents the params ratio, which is calculated as the relation between the number of params from a generated sub-model and the number of params from the base model. On the other hand, the vertical-axis represents the testing accuracy of the model after retraining. It is important to note that a dotted line has been included to show the accuracy reached using the original model of the MobileNet network (labeled as original accuracy). In addition, the size of the scatter points changes according to the parameters ratio (smaller points for smaller parameters ratio, bigger points for higher parameters ratio), and their color is assigned by the testing accuracy reached.

Similar to the VGG16 Road Damage experiment, most sub-models achieved an excellent performance. Therefore, even using different base models, our proposed pruning method shows an efficient way to obtain competitive accuracy levels and compression rates.

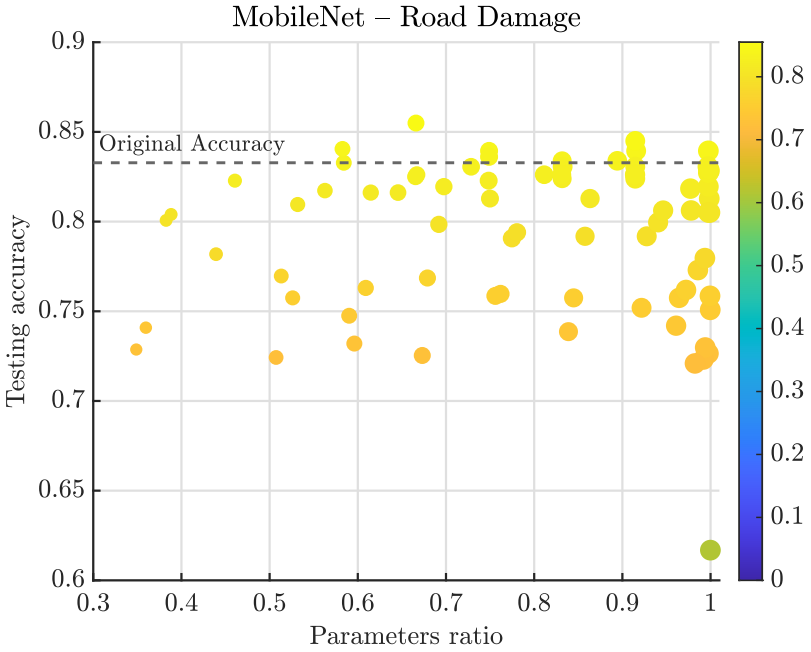

Figure 17. MobileNet sub-models trained on the road damage dataset.

Table 6 shows the performance and information on the top five generated sub-models, selected according to their accuracy reached and params ratio. In the first column of Table 6, the names of the models are shown. The first part of the name indicates the origin layer of the pruning, and the second part of the name stands for the destination layer of the pruning.

Table 6. MobileNet Top-5 Road Damage sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
DS-Conv 2 to DS-Conv 13	**0.3826**	**0.2547 G (22.34%)**	0.8007	0.8028
DS-Conv 3 to DS-Conv 13	0.3883	0.3690 G (32.36%)	0.8040	0.8208
DS-Conv 5 to DS-Conv 13	0.4606	0.5426 G (47.59%)	0.8228	0.8786
DS-Conv 6 to DS-Conv 13	0.5826	0.6208 G (54.45%)	0.8405	**0.9061**
DS-Conv 6 to DS-Conv 12	0.6660	0.7241 G (63.51%)	**0.8549**	0.8938

As in the previous experiments, the top-accuracy generated sub-models have connections to the last sub-structures of the network. Thus, once again, the hypothesis of pruning networks by connecting the layers/blocks to the final sub-structures is supported. Then, this hypothesis can be used as a heuristic for determining the destination layers of our pruning method.

For a better illustration of the experimental results produced by pruning and reconnecting different sub-structures, in Figure 18 two heatmaps are presented. On each heatmap, there is a grid where the performance of the sub-models is mapped over an indicator.

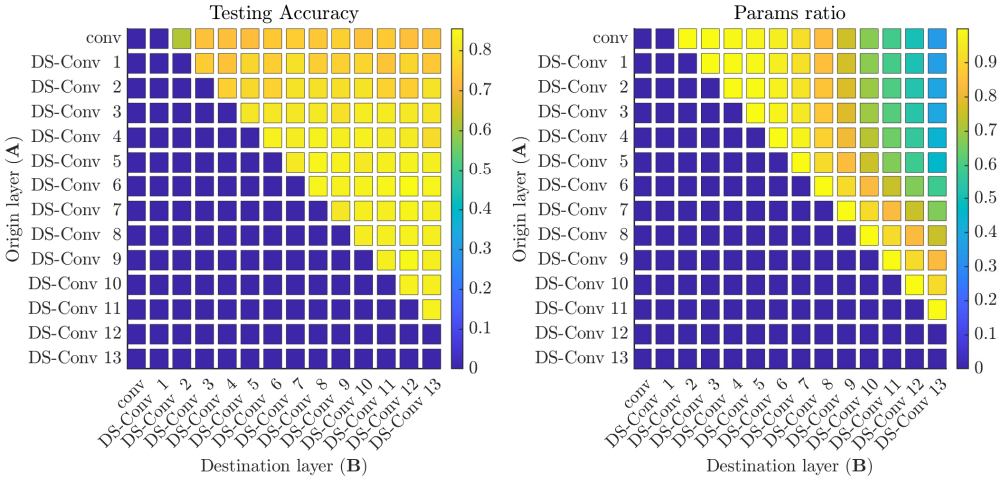

Figure 18. MobileNet Road Damage sub-models heatmaps.

To emphasize the compression reported in Figures 17 and 18, one of the top-accuracy sub-models (see Table 6) from this experiment is presented in Figure 19.

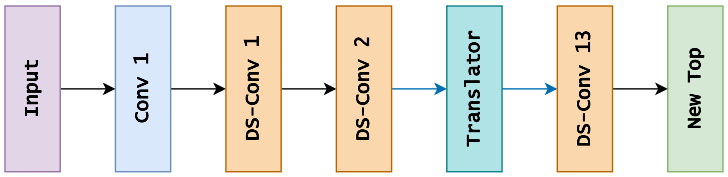

Figure 19. Top sub-model generated from the MobileNet architecture. DS-Conv 3 to DS-Conv 12 are pruned, thus, DS-Conv 2 and DS-Conv 13 are reconnected by the translator block.

The network presented in Figure 19 looks quite compact in comparison to the base model illustrated in Figure 9. Even when most of the depth separable convolutions are removed from the network, the translator block can recover an accuracy level close to the originally achieved by the base model. Furthermore, this compact model has close to 38% of the original number of parameters and requires just about 20% of the original FLOPs.

5.2.2. Flowers

In this subsection, experimental results for recognition of flowers using the images of the dataset [42], classified by MobileNet sub-models generated using the neuroplasticity-based pruning method, are presented.

In Figure 20, the sub-models generated from the MobileNet trained on the flowers dataset are displayed.

In contrast to the VGG16 experiment with the flowers dataset, in this experiment, different sub-models achieved testing accuracy levels equal to or above 0.8 and params compression up to 2×. The above may suggest that the MobileNet architecture is a robust feature extractor even when composed of a few blocks. Figure 19 illustrates this behavior.

On the other hand, there are horizontal and vertical patterns on the scatter points of the figure. These patterns appear over the figures from the other experiments, but they are slightly visible.

A group of sub-models produces the vertical pattern with a translator connection to the same destination layer (*B* fixed) but with a different origin. On the contrary, the horizontal pattern is produced by a group of sub-models with a translator connection from the same origin layer (*A* fixed) but with a different destination.

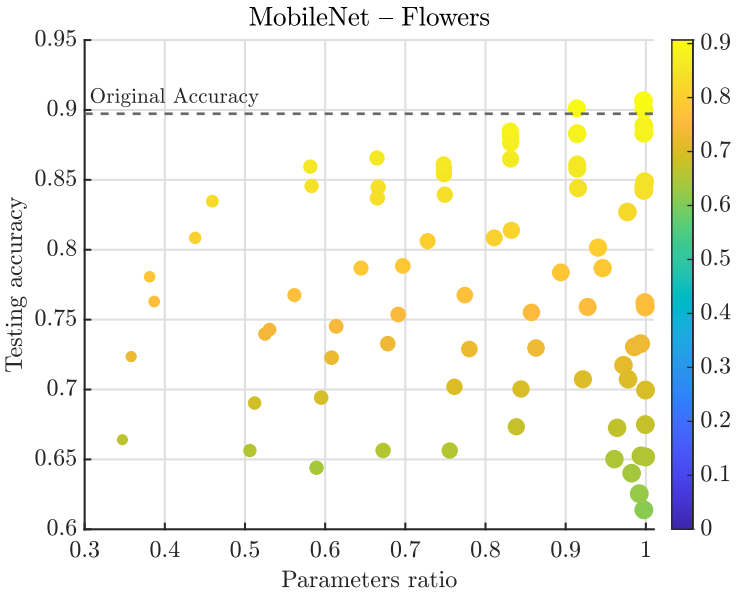

Figure 20. MobileNet sub-models trained on the flowers dataset.

To visually identify the horizontal and vertical patterns, Figure 21 illustrates different groups of sub-models within these conditions. All the scatter points from Figure 21 are identical to the scatter points from Figure 20 but in gray tone, except for those plotted with colors to emphasize the relevant patterns.

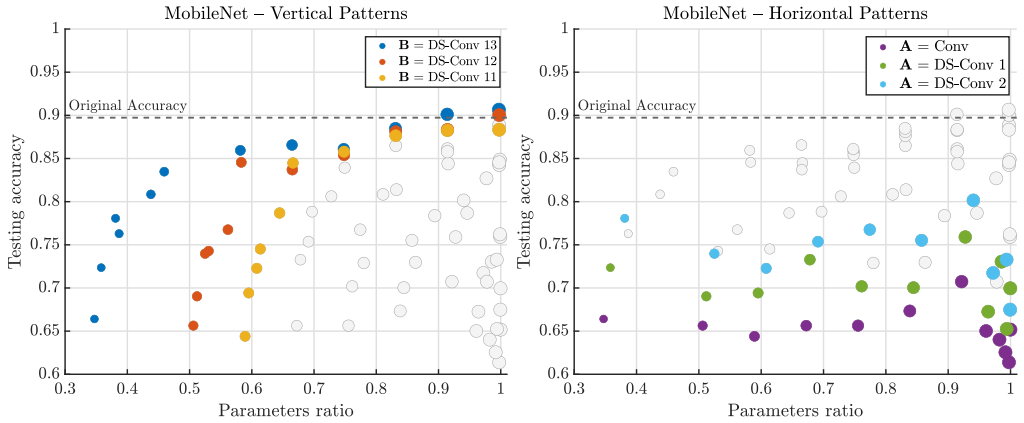

Figure 21. MobileNet sub-models patterns.

From the above figure, it can be observed that sub-models with connections to the last blocks of the network are attached to the left side of the horizontal-axis at the beginning, but then a slope appears as far as the params ratio increases. This dynamic supports that connections to the last layers/blocks of the network are desired because this creates sub-models with low params ratio. Furthermore, for this particular architecture, the efficiency of the DS-Conv blocks as feature extractors allows pruning the network by almost a 30% without critically affecting the network performance.

Table 7 shows the performance and information of the top five generated sub-models, selected according to their accuracy reached and params ratio. Once more, the models are named using the first part of the name as the origin layer, and the second part of the name refers to the destination layer of the pruning.

Table 7. MobileNet Top-5 Flowers sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
DS-Conv 4 to DS-Conv 13	**0.4378**	**0.4354 G (38.19%)**	0.8085	0.8782
DS-Conv 5 to DS-Conv 13	0.4593	0.5426 G (47.60%)	0.8347	0.9560
DS-Conv 5 to DS-Conv 12	0.5831	0.6204 G (54.42%)	0.8456	0.9418
DS-Conv 6 to DS-Conv 13	0.5816	0.6208 G (54.46%)	0.8595	0.9639
DS-Conv 7 to DS-Conv 12	0.6647	0.8291 G (72.73%)	**0.8656**	**0.9649**

Figure 22 shows the heatmaps from the sub-models trained on the Flowers dataset.

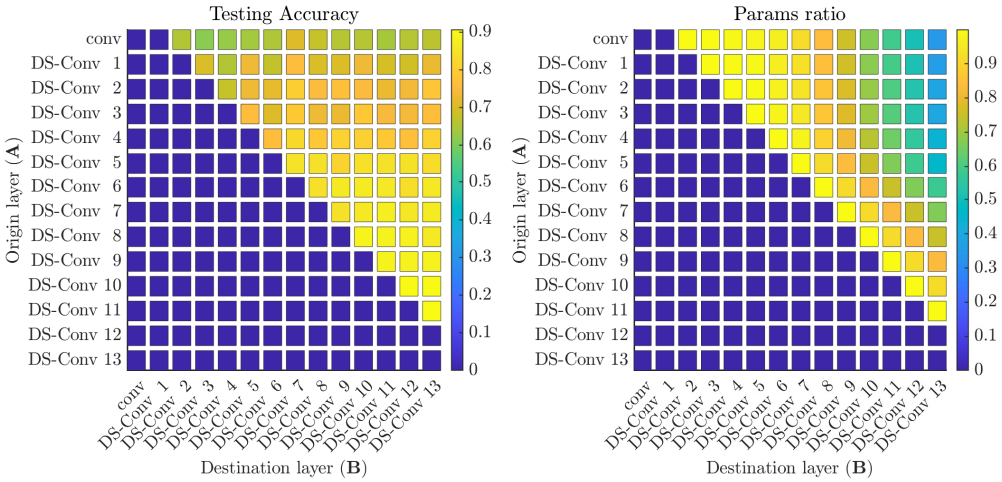

Figure 22. MobileNet Flowers sub-models heatmaps.

For those models with regular performance but a high compression rate, it should be easy to increase the accuracy levels because retraining is applied once and only over the translator block. Therefore, fine-tuning or another training round for all the layers may be enough to improve the network performance.

5.2.3. Caltech 101

In this subsection, experimental results for recognition of 101 different objects using the images of the dataset [43], classified by MobileNet sub-models generated using the neuroplasticity-based pruning method, are presented.

In Figure 23, the sub-models generated from the MobileNet trained on the caltech-101 dataset are shown.

Once again, as in the VGG16 experiment with this architecture, most sub-models do not present the hoped params ratio and the compression reached is marginal. Therefore, even when the MobileNet architecture has shown an excellent performance as a feature extractor to compress the network by pruning most of the blocks, it is difficult to replicate the result using the unbalanced Caltech-101 dataset.

Table 8 shows the performance and information of the top five generated sub-models, selected according to their accuracy reached and params ratio. Once more, the models are named using the first part of the name as the origin layer, and the second part of the name refers to the destination layer of the pruning.

From the MobileNet experiments, we observed a better compression and performance compared to the experimental results obtained from the VGG16 sub-models. Thus, using our proposed pruning method, the base model is compressed into different efficient and lightweight networks.

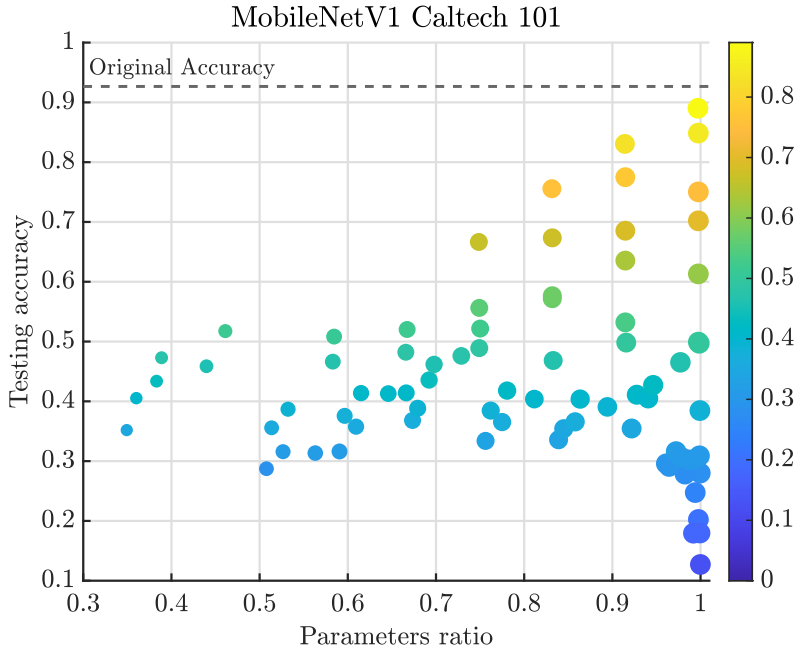

Figure 23. MobileNet sub-models trained on the caltech-101 dataset.

Table 8. MobileNet Top-5 Caltech-101 sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
DS-Conv 8 to DS-Conv 13	**0.7486**	**0.8308 G (72.88%)**	0.6665	0.7264
DS-Conv 8 to DS-Conv 12	0.8319	0.9341 G (81.94%)	0.6735	0.7272
DS-Conv 9 to DS-Conv 13	0.8314	0.9357 G (82.08%)	0.7557	0.7530
DS-Conv 9 to DS-Conv 12	0.9147	1.0390 G (91.14%)	0.7749	0.7906
DS-Conv 10 to DS-Conv 13	0.9142	1.0410 G (91.32%)	**0.8306**	**0.8197**

Figure 24 shows the heatmaps from the sub-models trained on the Caltech-101 dataset.

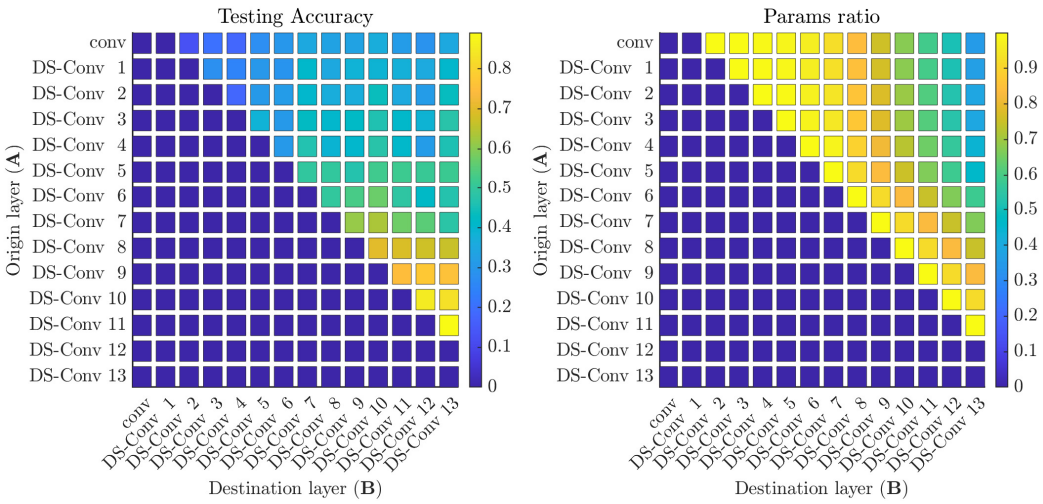

Figure 24. MobileNet Caltech-101 sub-models heatmaps.

In addition, the hypothesis of pruning networks by connecting layers/blocks to the later sub-structures of the network seems solid according to the experimental results produced. Once again, our pruning-retraining method is as simple as retraining if used over a sub-structure already identified as redundant. For instance, we assume that most of the middle layers/blocks of the network have knowledge redundancy, which is removed by testing connections up to the last sub-structures of the network.

5.3. MobileNetV2

This section presents the experimental results obtained using our neuroplasticity pruning method with the base architecture of a MobileNetV2 that has been trained with several datasets.

In Table 9, the performance and information of the architectures computed with the Hyperband [45] algorithm are presented. For each architecture we report on their number of parameters, floating-point operations per second (FLOPs), testing accuracy, and training accuracy.

The FLOPs are calculated using the keras-flops [48] python module.

Table 9. MobileNetV2 architectures, computed using the Hyperband algorithm.

Dataset	Params	FLOPs	Testing Acc.	Train Acc.
Road Damage	2.3067 M	0.6134 G	0.8361	0.8876
Flowers	2.2965 M	0.6134 G	0.8927	0.9646
Caltech 101	2.3096 M	0.6134 G	0.9209	0.9974

As it can be observed, the architectures have a similar number of parameters and FLOPs, and only the accuracy for training and testing varies because of the dataset used. New sub-models are created from these computed architectures, applying the proposed pruning method by removing entire sub-structures of the network.

5.3.1. Road Damage

In this subsection, experimental results for recognition of road damages using the images of the dataset [41], classified by MobileNetV2 sub-models generated using the neuroplasticity-based pruning method, are presented.

For this experiment, it is proposed to use the inverted bottlenecks as the origin and destination sub-structures. Therefore, in Figure 25, a total of 136 scatter points are plotted, where each point on the figure represents a generated sub-model after the one-shot pruning and retraining process.

In the above figure, the horizontal-axis represents the params ratio, which is calculated as the relation between the number of params from a generated sub-model and the number of params from the base model. On the other hand, the vertical-axis represents the testing accuracy of the model after retraining. It is important to note that a dotted line has been included to show the accuracy reached using the original model of the MobileNetV2 network (labeled as original accuracy). In addition, the size of the scatter points changes according to the parameters ratio (smaller points for smaller parameters ratio, bigger points for higher parameters ratio), and their color is assigned by the testing accuracy reached.

From the experimental results produced, it can be observed that this time the sub-models did not outperform the accuracy reached by the base model. However, there are different models with parameters reduction of almost 80% and accuracy levels nearby to the original from the base model.

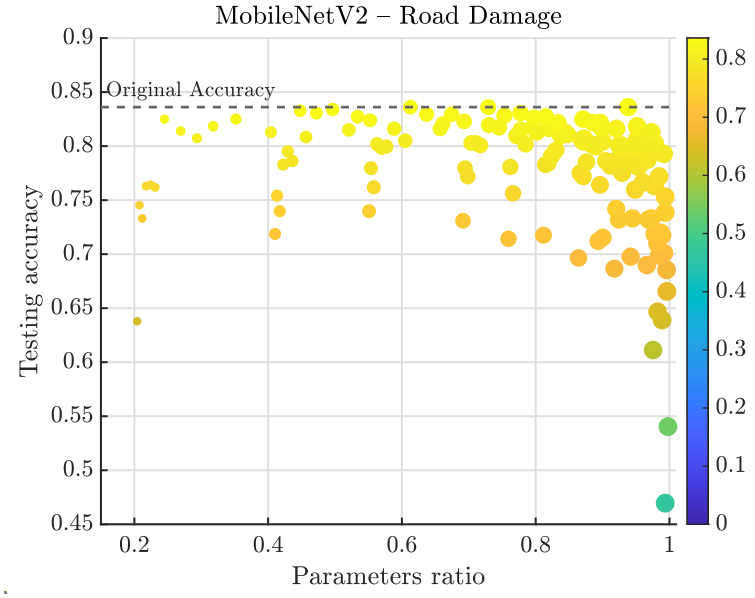

Figure 25. MobileNetV2 sub-models trained on the road damage dataset.

Table 10, shows the performance and information of the top five generated sub-models, selected according to their accuracy reached and params ratio. In the first column of Table 10, the names of the models are shown. The first part of the name indicates the origin layer of the pruning, and the second part of the name stands for the destination layer of the pruning.

Table 10. MobileNetV2 Top-5 Road Damage sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
block6 to conv2	**0.2450**	**0.2948 G (48.06%)**	**0.8250**	0.8592
block7 to conv2	0.2693	0.3158 G (51.48%)	0.8140	0.8568
block8 to conv2	0.2935	0.3367 G (54.89%)	0.8073	0.8587
block9 to conv2	0.3178	0.3577 G (58.31%)	0.8184	**0.8596**
block10 to conv2	0.3519	0.3845 G (62.68%)	**0.8250**	0.8559

As it can be seen in Table 10, the MobileNetV2 shows excellent performance as a feature extractor such as the MobileNet in the previous experiments. Furthermore, one of the top-accuracy sub-models achieves a remarkable parameters reduction, and their accuracy is barely different from the original. To visualize the network compression, this sub-model is displayed in Figure 26.

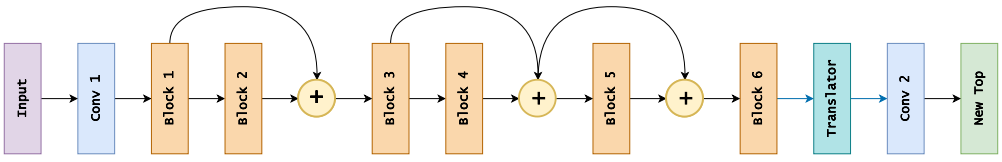

Figure 26. Top sub-model generated from the MobileNetV2 architecture. Block 7 to Block 16 are pruned, thus, Block 6 and Conv 2 (the last layer) are reconnected by the translator block.

For a better illustration of the experimental results produced, in Figure 27 the heatmaps of the experiment are presented.

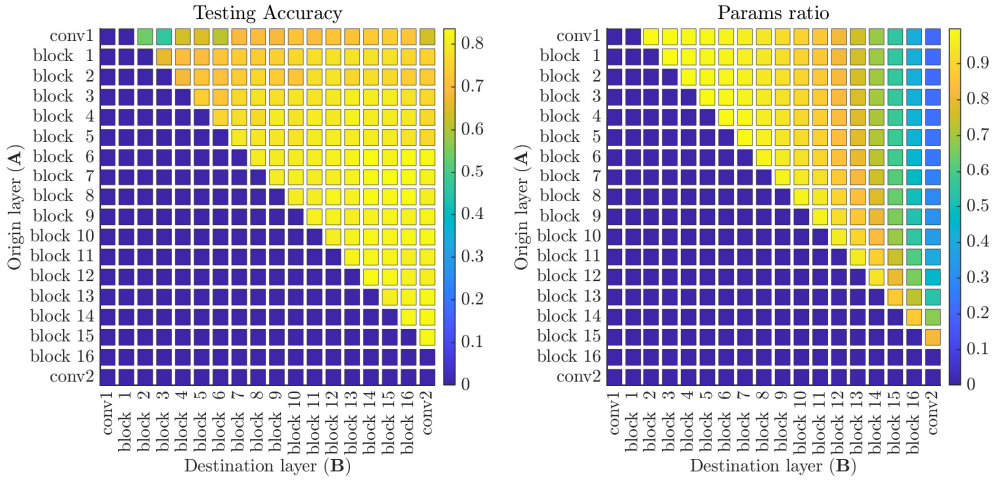

Figure 27. MobileNetv2 Road Damage sub-models heatmaps.

5.3.2. Flowers

In this subsection, experimental results for recognition of flowers using the images of the dataset [42], classified by MobileNetV2 sub-models generated using the neuroplasticity-based pruning method, are presented.

In Figure 28, the sub-models generated from the MobileNetV2 trained on the flowers dataset are displayed.

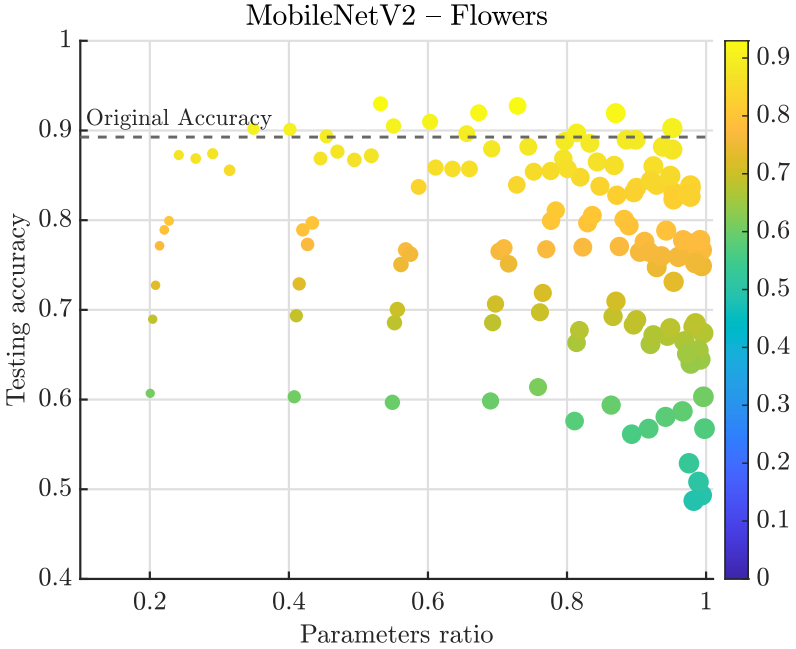

Figure 28. MobileNetV2 sub-models trained on the flowers dataset.

Table 11 presents the performance and information of the top five generated sub-models, selected according to their accuracy reached and params ratio. Once more, the models are named using the first part of the name as the origin layer, and the second part of the name refers to the destination layer of the pruning.

Table 11. MobileNetV2 Top-5 Flowers sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
block6 to conv2	**0.2417**	**0.2948 G (48.06%)**	0.8726	0.9378
block7 to conv2	0.2660	0.3158 G (51.48%)	0.8687	0.9457
block8 to conv2	0.2904	0.3367 G (54.89%)	0.8741	0.9474
block9 to conv2	0.3147	0.3577 G (58.31%)	0.8556	0.9507
block10 to conv2	0.3490	0.3845 G (62.68%)	**0.9019**	**0.9666**

For a better illustration of the experimental results produced, in Figure 29 the heatmaps of the experiment are presented.

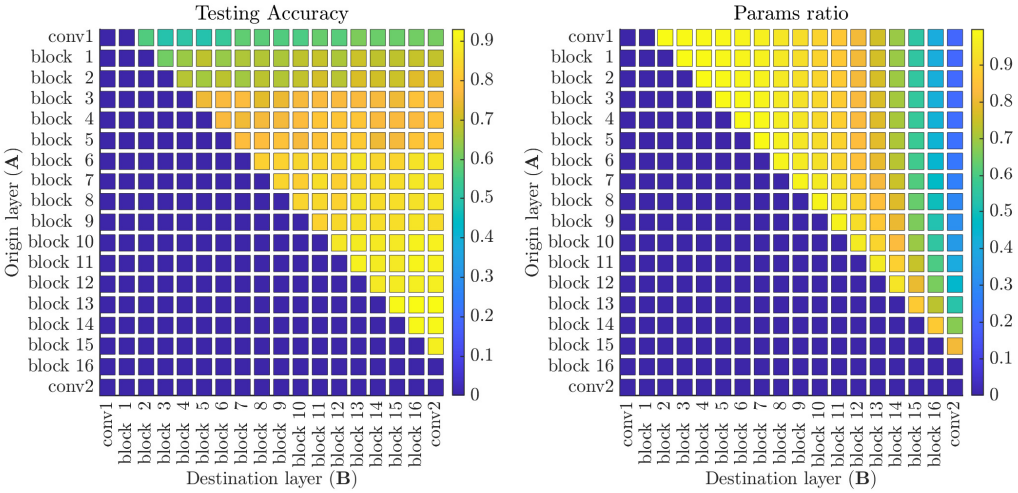

Figure 29. MobileNetV2 Flowers sub-models heatmaps.

Figures 28 and 29 shows that, as in the previous experiment, the MobileNetV2 is a robust feature extractor. Furthermore, Table 11 shows that in this experiment, a compression of around 80% is achieved again, besides reaching accuracy levels close to or even better than the originally reported by the base model.

Contrary to the VGG16 and MobileNet experimental results with this dataset, the MobileNetV2 demonstrates that even with an unbalanced dataset is possible to extract relevant features transformations and embed them into the translator without losing accuracy.

5.3.3. Caltech 101

In this subsection, experimental results for recognition of 101 different objects using the images of the dataset [43], classified by MobileNetV2 sub-models generated using the neuroplasticity-based pruning method, are presented.

For this experiment, only models with connections to later sub-structures are created. As the experimental results suggest, the models with later connections on the network have low params ratio, and most of the time, they achieve an excellent performance. Thus, in Figure 30, the 31 sub-models generated from the MobileNetV2 trained on the Caltech-101 dataset are shown.

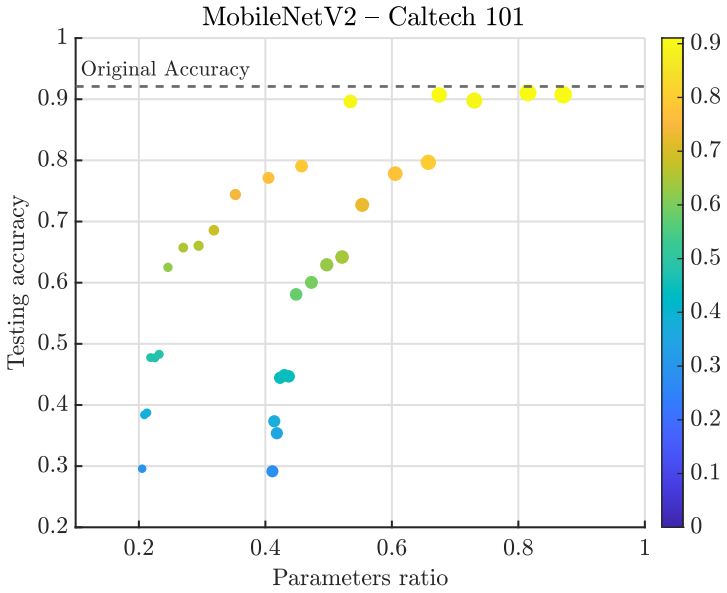

Figure 30. MobileNetV2 sub-models trained on the caltech-101 dataset.

As can be observed in the scatter models from Figure 30 and the top five sub-models reported in Table 12, once more, the models are named using the first part of the name as the origin layer, and the second part of the name refers to the destination layer of the pruning.

Table 12. MobileNetV2 Top-5 Caltech-101 sub-models.

Re-Connection (A to B)	Params Ratio	FLOPs (%)	Testing Accuracy	Training Accuracy
block10 to conv2	**0.3527**	**0.3845 G (62.68%)**	0.7441	0.8500
block11 to conv2	0.4049	0.4304 G (70.17%)	0.7414	0.8861
block12 to conv2	0.4049	0.4763 G (77.65%)	0.7906	0.8861
block11 to block16	0.6054	0.4751 G (77.45%)	0.7783	0.9045
block13 to conv2	0.5345	0.5098 G (83.11%)	**0.8963**	**0.9921**

Once again, the experimental results support that the MobileNetV2 architecture is a robust feature extractor. In this experiment, there is a significant compression achieved compared to the previous experiment with the same dataset, and the accuracy loss is marginal.

In the two previous experiments, the top-accuracy sub-models use up to block 10 as the origin sub-structure for the network compression. However, for this experiment, it can be observed in Figure 31 that more blocks were required to match the compression rates and accuracy levels. The above may suggest that it is possible to identify the optimal origin and destination sub-structures over the network according to the dataset. For instance, in the VGG16 experiments, using layers 4-1 and forward as origin, we ensure that the translator will be able to remap the knowledge and recover the original accuracy levels after pruning.

From the MobileNetV2 experiments, we observed that this architecture is a power-full feature extractor because the sub-models almost achieved the original accuracy using only 20% of the number of params from the base model. Therefore, using the proposed one-shot pruning and retraining method, it is possible to use the network architecture to create efficient and straightforward sub-models with the proper number of blocks according to a specific dataset to be used.

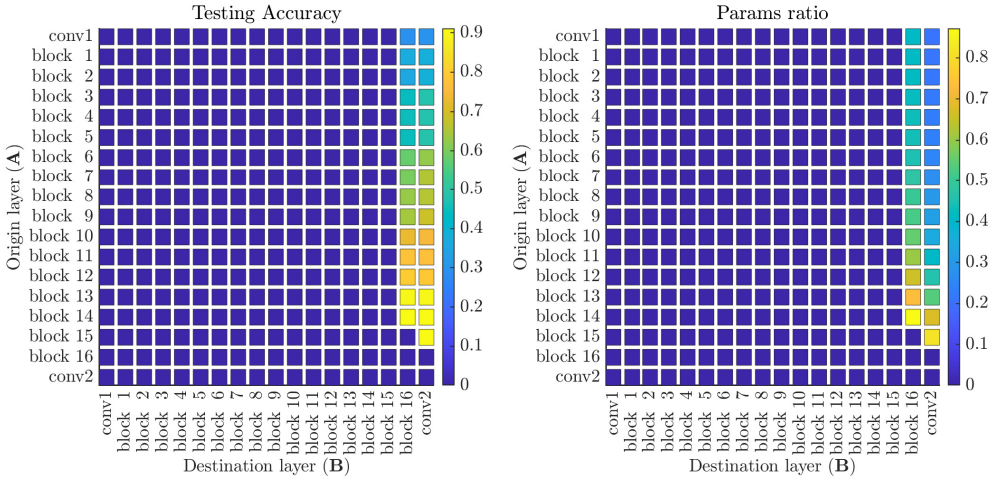

Figure 31. MobileNetV2 Caltech-101 sub-models heatmaps.

6. Conclusions and Future Work

In this work, we presented a novel bio-inspired structured pruning method designed to remove redundancy and unimportant connections by pruning entire layers or blocks from the network architecture. As a result, lightweight and efficient sub-networks are created from the base models.

In contrast to other pruning methods, in our approach, is not necessary to define a threshold or metric as the criterion to prune the network. Instead, only the origin and destination sub-structures are defined for the translator connection. Therefore, to test the impact of removing different sub-structures from the network, all the possible sub-models within different architectures were created, and their performance was measured.

In our approach, we propose using a one-shot pruning and retraining strategy. In other words, first, when the origin and destination structures are selected, the network is pruned and reconnected with the translator. After that, retraining is applied using the same hyper-parameters as in the regular training, but with all the layers frozen except by the translator. The above allows keeping intact the knowledge from the previous training phase and compress the acquired knowledge into the translator block. Furthermore, if we want to explore the impact of removing different network sub-structures, the pruning-retraining process may be run out in parallel mode as a grid search algorithm.

Heuristically, we found that models with connections between the middle and last layers present a low params ratio and excellent performances. Furthermore, according to the dataset used, a given architecture may require more or fewer layers/blocks to recover the accuracy level from the base model successfully. Therefore, to achieve competitive compression rates and performances, the origin and destination structures are strongly related to the features from the dataset.

After pruning and retraining, in different works it is suggested to apply fine-tuning to increase the network performance. Besides, multiple works suggest combining pruning and compression techniques to improve the compression rate if it is necessary. In our experiments, we did not use any of these techniques. However, our approach is suitable for all of these methods because the resultant network is a compressed architecture of the base model. Furthermore, there is not necessary to use extra libraries or tools for sparse or complex graph computations.

Future Work

To use our proposed method in a practical application, we are designing a metric to compute the optimal origin and destination structures within a network architecture.

In addition, we are working on a Python module designed to automate the one-shot pruning and retraining process with our proposed method.

Author Contributions: Conceptualization, C.V.; Formal analysis, C.L.-F. and N.A.-D.; Funding acquisition, C.L.-F. and N.A.-D.; Investigation, J.D.C. and C.V.; Methodology, C.V. and N.A.-D.; Project administration, N.A.-D.; Software, J.D.C.; Supervision, C.V. and N.A.-D.; Validation, C.L.-F.; Writing—original draft, J.D.C.; Writing—review & editing, C.V. and N.A.-D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CONACyT CB-2015-258068.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results

References

1. Hussain, M.; Bird, J.J.; Faria, D.R. A study on cnn transfer learning for image classification. In *UK Workshop on Computational Intelligence*; Springer: Cham, Switzerland, 2018; pp. 191–202.
2. Lee, H.; Kwon, H. Going deeper with contextual CNN for hyperspectral image classification. *IEEE Trans. Image Process.* **2017**, *26*, 4843–4855. [CrossRef] [PubMed]
3. Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef] [PubMed]
4. Dhillon, A.; Verma, G.K. Convolutional neural network: A review of models, methodologies and applications to object detection. *Prog. Artif. Intell.* **2020**, *9*, 85–112. [CrossRef]
5. Zhong, Y.; Gao, J.; Lei, Q.; Zhou, Y. A Vision-Based Counting and Recognition System for Flying Insects in Intelligent Agriculture. *Sensors* **2018**, *18*, 1489. [CrossRef]
6. Zhong, Y.; Chen, X.; Jiang, J.; Ren, F. A cascade reconstruction model with generalization ability evaluation for anomaly detection in videos. *Pattern Recognit.* **2022**, *122*, 108336. [CrossRef]
7. Guo, Y.; Liu, Y.; Georgiou, T.; Lew, M.S. A review of semantic segmentation using deep neural networks. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 87–93. [CrossRef]
8. Wang, P.; Chen, P.; Yuan, Y.; Liu, D.; Huang, Z.; Hou, X.; Cottrell, G. Understanding convolution for semantic segmentation. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; pp. 1451–1460.
9. Martinez-Soltero, G.; Alanis, A.Y.; Arana-Daniel, N.; Lopez-Franco, C. Semantic Segmentation for Aerial Mapping. *Mathematics* **2020**, *8*, 1456. [CrossRef]
10. Schlemper, J.; Caballero, J.; Hajnal, J.V.; Price, A.N.; Rueckert, D. A deep cascade of convolutional neural networks for dynamic MR image reconstruction. *IEEE Trans. Med Imaging* **2017**, *37*, 491–503. [CrossRef]
11. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
12. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv* **2017**, arXiv:1710.09282. [CrossRef]
13. Lin, S.; Ji, R.; Chen, C.; Tao, D.; Luo, J. Holistic cnn compression via low-rank decomposition with knowledge transfer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 2889–2905. [CrossRef]
14. Wen, W.; Xu, C.; Wu, C.; Wang, Y.; Chen, Y.; Li, H. Coordinating filters for faster deep neural networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 658–666.
15. Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; Cheng, J. Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4820–4828.
16. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713.
17. Lin, X.; Zhao, C.; Pan, W. Towards accurate binary convolutional neural network. In Proceedings of the Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
18. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2016**, arXiv:1510.00149. [CrossRef]
19. Hu, H.; Peng, R.; Tai, Y.W.; Tang, C.K. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv* **2016**, arXiv:1607.03250. [CrossRef]
20. Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; Kautz, J. Pruning convolutional neural networks for resource efficient inference. *arXiv* **2016**, arXiv:1611.06440. [CrossRef]
21. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. *arXiv* **2016**, arXiv:1608.08710. [CrossRef]

22. Guo, Y.; Yao, A.; Chen, Y. Dynamic network surgery for efficient dnns. In Proceedings of the Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016; Volume 29.
23. Luo, J.H.; Wu, J. An entropy-based pruning method for cnn compression. *arXiv* **2017**, arXiv:1706.05791. [CrossRef]
24. Wu, H.; Tang, Y.; Zhang, X. A pruning method based on the measurement of feature extraction ability. *Mach. Vis. Appl.* **2021**, *32*, 20. [CrossRef]
25. French, R.M. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.* **1999**, *3*, 128–135. [CrossRef]
26. Mateos-Aparicio, P.; Rodríguez-Moreno, A. The impact of studying brain plasticity. *Front. Cell. Neurosci.* **2019**, *13*, 66. [CrossRef]
27. Carlson, R.A. Restructuring in Learning. In *Encyclopedia of the Sciences of Learning*; Seel, N.M., Ed.; Springer: Boston, MA, USA, 2012; pp. 2853–2856. [CrossRef]
28. Cramer, S.C.; Nelles, G.; Benson, R.R.; Kaplan, J.D.; Parker, R.A.; Kwong, K.K.; Kennedy, D.N.; Finklestein, S.P.; Rosen, B.R. A functional MRI study of subjects recovered from hemiparetic stroke. *Stroke* **1997**, *28*, 2518–2527. [CrossRef]
29. Teasell, R.; Bayona, N.A.; Bitensky, J. Plasticity and reorganization of the brain post stroke. *Top. Stroke Rehabil.* **2005**, *12*, 11–26. [CrossRef]
30. Murphy, T.H.; Corbett, D. Plasticity during stroke recovery: From synapse to behaviour. *Nat. Rev. Neurosci.* **2009**, *10*, 861–872. [CrossRef] [PubMed]
31. Chrol-Cannon, J.; Jin, Y. Computational modeling of neural plasticity for self-organization of neural networks. *Biosystems* **2014**, *125*, 43–54. [CrossRef] [PubMed]
32. Op de Beeck, H.P.; Baker, C.I. The neural basis of visual object learning. *Trends Cogn. Sci.* **2010**, *14*, 22–30. [CrossRef] [PubMed]
33. Nudo, R.J. Recovery after brain injury: Mechanisms and principles. *Front. Hum. Neurosci.* **2013**, *7*, 887. [CrossRef]
34. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. In Proceedings of the Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2015, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
35. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems: 26th Annual Conference on Neural Information Processing Systems 2012, Lake Tahoe, NV, USA, 3–6 December 2012; Volume *25*.
36. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556. [CrossRef]
37. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861. [CrossRef]
38. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
39. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis. (IJCV)* **2015**, *115*, 211–252. [CrossRef]
40. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available online: https://www.tensorflow.org (accessed on 10 May 2022).
41. Angulo, A.; Vega-Fernández, J.A.; Aguilar-Lobo, L.M.; Natraj, S.; Ochoa-Ruiz, G. Road damage detection acquisition system based on deep neural networks for physical asset management. In Proceedings of the Mexican International Conference on Artificial Intelligence, Xalapa, Mexico, 27 October–2 November 2019; Springer: Cham, Switzerland, 2019; pp. 3–14.
42. Mamaev, A. Flowers Recognition. Dataset Retrived from kaggle.com. 2018. Available online: https://www.kaggle.com/alxmamaev/flowers-recognition/version/2 (accessed on 10 May 2022).
43. Li, F.-F.; Fergus, R.; Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop, Washington, DC, USA, 27 June–2 July 2004; p. 178.
44. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
45. Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.* **2017**, *18*, 6765–6816.
46. O'Malley, T.; Bursztein, E.; Long, J.; Chollet, F.; Jin, H.; Invernizzi, L. KerasTuner. 2019. Available online: https://github.com/keras-team/keras-tuner (accessed on 10 May 2022).
47. Camacho, J.D.; Villaseñor, C.; Alanis, A.Y.; Lopez-Franco, C.; Arana-Daniel, N. sKAdam: An improved scalar extension of KAdam for function optimization. *Intell. Data Anal.* **2020**, *24*, 87–104. [CrossRef]
48. Tokusumi, T. KerasFlops. 2020. Available online: https://github.com/tokusumi/keras-flops (accessed on 10 May 2022).
49. Scheidegger, F.; Istrate, R.; Mariani, G.; Benini, L.; Bekas, C.; Malossi, C. Efficient image dataset classification difficulty estimation for predicting deep-learning accuracy. *Vis. Comput.* **2021**, *37*, 1593–1610. [CrossRef]