electronics

MDPI

Article

Deep Learning Architecture Improvement Based on Dynamic Pruning and Layer Fusion

Qi Li †, Hengyi Li † and Lin Meng *,†

Department of Electronic and Computer Engineering, Ritsumeikan University, Kusatsu 525-8577, Japan
* Correspondence: menglin@fc.ritsumei.ac.jp
† Current address: College of Science and Engineering, Ritsumeikan University, 1-1-1, Nojihigashi, Kusatsu 525-8577, Japan.

Abstract: The heavy workload of current deep learning architectures significantly impedes the application of deep learning, especially on resource-constrained devices. Pruning has provided a promising solution to compressing the bloated deep learning models by removing the redundancies of the networks. However, existing pruning methods mainly focus on compressing the superfluous channels without considering layer-level redundancies, which results in the channel-pruned models still suffering from serious redundancies. To mitigate this problem, we propose an effective compression algorithm for deep learning models that uses both the channel-level and layer-level compression techniques to optimize the enormous deep learning models. In detail, the channels are dynamically pruned first, and then the model is further optimized by fusing the redundant layers. Only a minor performance loss results. The experimental results show that the computations of ResNet-110 are reduced by 80.05%, yet the accuracy is only decreased by 0.72%. Forty-eight convolutional layers could be discarded from ResNet-110 with no loss of performance, which fully demonstrates the efficiency of the proposal.

Keywords: convolutional neural network; architecture improvement; dynamic channel pruning; memory access improvement

check for updates

Citation: Li, Q.; Li, H.; Meng, L. Deep Learning Architecture Improvement Based on Dynamic Pruning and Layer Fusion. *Electronics* **2023**, *12*, 1208. https://doi.org/10.3390/electronics12051208

Academic Editor: Donghyeon Cho

Received: 1 February 2023
Revised: 22 February 2023
Accepted: 1 March 2023
Published: 2 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Convolution neural networks (CNNs) have been proven to be effective in various applications [1–3]: object detection [4,5], cultural heritage protection [6], environment monitoring [7], robotics [8–10] and healthcare [11].

CNNs are designed to extract features from the input, which are used to reflect whether a region of the input has certain properties [12]. Based on these features, CNNs can accomplish tasks such as classification or detection. For example, in drone-based disaster management applications, CNNs techniques are used to quickly and accurately extract features of disasters, such as forest fires, landslides, and volcanic eruptions, from images captured by drone camera [13].

However, the enormous numbers of computations and parameters of CNNs hinder further development. Thus, it is not practical to deploy heavy CNNs on resource-constrained computing devices, such as embedded systems and mobile devices [14–16]. To address the problems, substantial research efforts have been devoted to compression techniques: channel pruning [17–20], low-rank decomposition [21–23], and weight quantization [24,25]. Channel pruning is performed by locating and removing redundant channels to reduce the numbers of floating-point operations (FLOPs) and parameters. In addition, the pruned model is intact in parallelism, which contributes to the efficient utilization of hardware resources [26].

After the model is compressed by channel pruning, many convolutional layers are equipped with only a few channels. These layers are defined as thin layers. Channel pruning is designed to remove unimportant channels and keep relatively important ones, so the

remaining channels have valuable contributions. However, since the residual connection is effective, mainstream models [27] (such as ResNet [28], DenseNet [29], and MobileNet [30]) adopt design patterns where multiple layers form the bottleneck structure and residual connection are applied to the input and output of the bottleneck structure. To maintain the functional integrity of the residual connection, most pruning strategies [31,32] do not modify the input channels of the first layer and the output channels of the last layer in the bottleneck structure. Hence, even though the thin layer is important to the model, there is a lot of redundancy in the corresponding bottleneck structure. It is feasible to further compress the model by utilizing the redundancy in the bottleneck structure.

On the other hand, models with fewer layers have more benefits for hardware. In recent years, the use of CNNs on resource-constrained devices has gained attention, e.g., the field-programmable gate array (FPGA). The combination of high performance and high power efficiency is leading to the adoption of FPGAs in a variety of CNN-based applications. However, since CNN models are designed to be bloated, a large number of weights need to be stored in external memory and transferred to the FPGAs during computation [33,34]. This process requires additional energy and time. The energy cost due to the increased memory accesses and data movement even exceeds the energy cost of computation [35–38]. As a result, the implementation of deep learning models on FPGAs or other lightweight devices should be accompanied by optimizations, such as model compression and weight quantization. Removing layers could further contribute to solving this problem by reducing the load of the layer weights and the feature maps. Thus, a method to remove thin layers and the corresponding bottleneck structure while preserving the feature extraction capability is urgently needed [39].

Therefore, this paper proposes an architecture improvement approach for CNNs that aims to improve the performance of a model on resource-constrained devices by optimizing the model at the channel level and layer level. Specifically, first, the model is compressed by dynamic pruning, where highly sparse channels are dynamically removed. Then the channel-level compressed model is further optimized by layer fusion, the redundant structure is removed, and other layers substitute its function. Moreover, knowledge distillation and short–long fine-tuning are introduced to layer fusion to reduce performance loss. As layer fusion proceeds, the optimal architecture for the current task is obtained. The proposal was applied to various models, and experimental results show that the improved models can achieve high performance with fewer computational resources.

The main contributions of the paper are as follows:

- A method for layer-level compression of CNNs is proposed. By introducing knowledge distillation and short–long fine-tuning, redundant layers are removed with lower accuracy loss.
- The proposal may provide an idea for applications that desire to reduce memory access more than reduce computational complexity.

The rest of the paper is organized as follows: Section 2 introduces related works. Section 3 details the methodology. Section 4 shows the experimental results. Section 5 concludes the paper. In addition, all the abbreviations and definitions are listed in the Appendix A.

2. Related Work

This section reviews channel pruning and knowledge distillation, and then gives a short introduction to the related work.

Channel pruning is an efficient approach to compressing CNN models. The challenge of channel pruning is to remove channels with the minimal performance loss. He et al. [40] proposed a new channel-pruning method. Inspired by tensor factorization improvement based on feature-map reconstruction, the proposal fully exploits the redundancy of feature maps between channels. Specifically, for the trained model, it aims to reduce the dimensions of the input feature maps of the layer, while minimizing the reconstruction error of the output feature maps, to achieve pruning of the layer. The minimization problem is solved

by two key steps: in the first step, the most representative channels are selected based on lasso regression, and the redundant channels are pruned. In the other step, the output of the remaining channels are reconstructed with linear least squares. Experiments showed that a 2× speed-up is achieved with a 1% accuracy loss.

As a representative method for model compression and acceleration, knowledge distillation effectively can learn small student models from large teacher models [41]. Huang et al. [42] proposed a new type of knowledge from the teacher model and transferred it to the student model. Specifically, the selective knowledge of neurons was exploited. Each neuron essentially extracts a certain pattern from the raw input. If a neuron is activated, that suggests some common property in the corresponding region that is relevant to the target task. Such information is valuable for the student model, as it provides an explanation for the prediction results of the teacher model. Hence, they proposed to align the distribution of neuron selectivity pattern between the student model and the teacher model. The maximum mean difference was introduced as a loss function to measure the discrepancy between the output feature maps of the teacher and student intermediate layers. The experimental results indicate that the proposal improves the performance of the student model significantly.

In some studies, knowledge distillation and channel pruning are combined. Aghli et al. [43] proposed a compression method for CNNs by combining knowledge distillation and weight pruning based on activation analysis. In detail, a select number of the layers in ResNet are pruned to avoid breaking the network's structure. Then, a new knowledge distillation architecture and loss function are used to compress the layers that were untouched in the previous step. The proposal was applied to the image classification task of head pose. Experimental results show that the model was significantly compressed while maintaining accuracy close to the baseline.

3. Methods

This paper intends to improve the architecture of CNNs. The proposals include dynamic pruning and layer fusion. First, the unimportant channels in the trained model are removed by dynamic pruning. Then, the redundant layers in the pruned model are further removed by layer fusion. The flow of the proposed method is described in Figure 1. In the next section, the details of each part are explained.

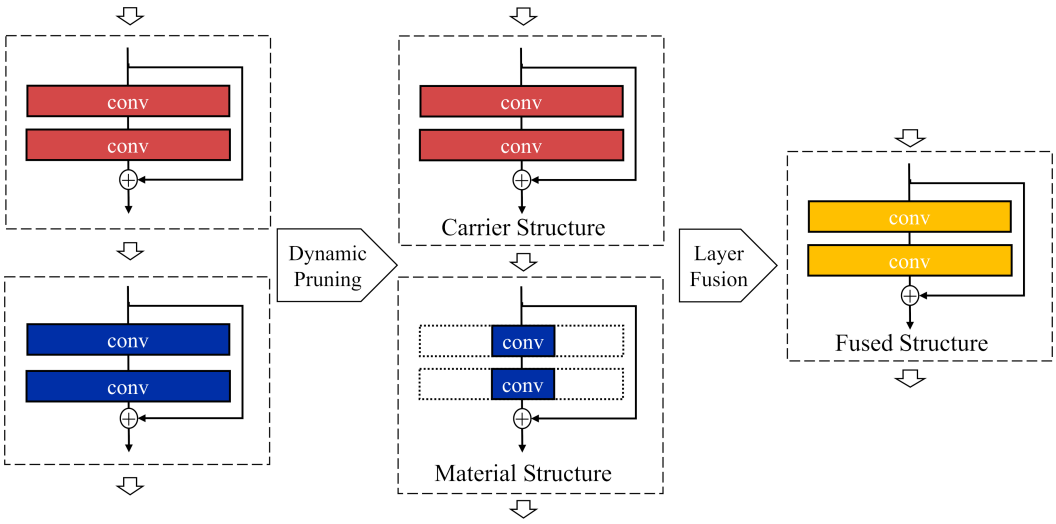


Figure 1. A diagram about the flow of proposed architecture-improvement strategy. The part enclosed by the dotted line indicates the compressed channel.

3.1. Preliminary

First, the convolution operation is introduced. $\mathbf{F}_{ij}$ indicates the filter that connects the ith input channel to the jth output channel. With the batch size set to 1, the feature map is a two-dimensional matrix that propagates between layers. If $\mathbf{M}_i$ denotes the input feature

map on the ith channel, $\circledast$ and b denote the convolution operator and the bias, and then the jth output feature map $\mathbf{O}_j$ is generated as

$$\sum_{i=1}^{C} \mathbf{M}_i \circledast \mathbf{F}_{ij} + b = \mathbf{O}_j. \tag{1}$$

The sparsity is introduced to describe the percentage of redundant data (i.e., zero elements) in the feature map. Since the output feature map is the sum of the convolution results of the input feature map and the filter, if the input feature map has high sparsity, the convolution result of the corresponding element is close to zero and has no effect on the output, which means the redundancy of input channels can be measured by the sparsity of the input feature map and removes redundant channels while having a minor impact on the model.

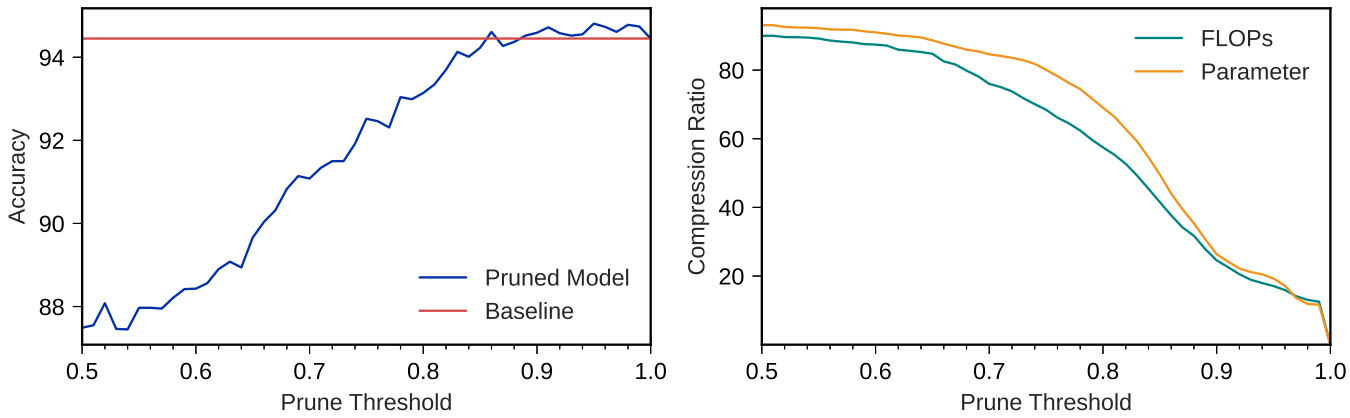
In addition, in CNNs, layers are tightly connected to each other by channels. Removing an input channel is also removing the corresponding output channel from the previous layer. As the highly sparse feature maps are generated from the corresponding output channel of the previous layer, the loss of the entire output channel is acceptable to the model. In detail, when the channel is removed, the corresponding filter is pruned and the model is compressed.

3.2. Dynamic Pruning

In the previous section, the sparsity and redundancy were explained. This part proposes a method to dynamically determine the pruning target based on the sparsity of the feature map. Let the model infer the entire validation set, and calculate the average sparsity of each feature map in each input channel. The average sparsity is defined as channel sparsity, which is used to evaluate the importance of the channel. Here, the pruning threshold is introduced to distinguish high sparsity from low sparsity. When the value of channel sparsity is greater than the pruning threshold, the corresponding channel is considered to be the pruning target. After pruning the model, fine-tune the remaining parts of the model to restore accuracy. If a low pruning threshold is set, major parts of the model are removed, which leads to difficulties in recovering the accuracy. Thus, determining an appropriate pruning threshold is critical.

The investigation of different thresholds on pruning results is conducted. Figure 2 shows the results. It should be noted that the accuracy of the model after fine-tuning is given in the figure. It can be seen that fine-tuning accuracy increases as the pruning threshold increases. When the accuracy is close to the baseline, the effect of increasing the pruning threshold is slight. In addition, according to Figure 2b, it can be concluded that the smaller the pruning threshold, the greater the compression ratio.

We empirically summarize the following: (1) When the fine-tuning accuracy is lower than the target accuracy, increasing the pruning threshold improves the fine-tuning accuracy. (2) When the fine-tuning accuracy is higher than the target accuracy, the compression ratio could be further enhanced by lowering the threshold slightly while the fine-tuning accuracy remains at the same level. Note that the target accuracy is not the baseline. Considering that the fine-tuning accuracy is unstable, target accuracy was set to slightly below the baseline in the experiment.



(**a**) (**b**)

Figure 2. Results of pruning experiments on CIFAR10 with pruning thresholds from 0.5 to 1.0. ResNet-56 model was pruned in the experiment. (**a**) Accuracy. (**b**) Compression ratio of FLOPs and parameters.

Therefore, the binary search algorithm is introduced to adjust the pruning threshold based on feedback from fine-tuning. Algorithm 1 describes the proposal in detail. P_u and P_l are the two endpoints of the search interval for the optimal pruning threshold and are initialized to 1 and 0.5, respectively. P_c is the current pruning threshold and is initialized to the midpoint of the search interval. The pruning process is simplified into the following steps: (1) Load the original trained model, or the compressed model, from the previous iteration. (2) One pruning attempt is performed based on P_c, and then the model is fine-tuned to get the corresponding accuracy. (3) If the fine-tuning accuracy is higher than the target accuracy, the upper endpoint, P_u, is updated to P_c. Then, P_c is reduced by a quarter of the search interval. If the fine-tuning accuracy is lower than the target accuracy, the lower endpoint P_l is updated to the value of P_c. Then, P_c is increased by half of the search interval. The above steps repeat until the gap between P_l and P_u is less than 0.03, an empirically determined termination condition. When the loop ends, the compression result of this iteration is obtained.

Algorithm 1: Algorithm for dynamic pruning.

Data: Pre-trained network
Result: The compressed network
initialization: the current pruning threshold P_c;
upper endpoint of target interval $P_u = 1.0$;
lower endpoint of target interval $P_l = 0.5$;
$P_c \leftarrow (1/2P_u + 1/2P_l)$;
while $P_u - P_l > 0.03$ **do**
  load trained network;
  select and delete channels based on P_c;
  fine-tune the pruned network and measure accuracy;
  if *accuracy* $\leq$ *target accuracy* **then**
    $P_u \leftarrow P_c$;
    $P_c \leftarrow (3/4P_u + 1/4P_l)$;
    save current network as the result;
  else
    $P_l \leftarrow P_c$;
    $P_c \leftarrow (1/2P_u + 1/2P_l)$;
  end
end

In addition, dynamic pruning is iterative to get better compression results. The compressed model of the previous iteration is the original model of this iteration. By analyzing the experimental data, too many pruning iterations make little improvement on the compression result; thus, the iterations were set to 3 by us.

3.3. Layer Fusion

After dynamic pruning, the pruned model has multiple thin layers. This part intends to deprecate these thin layers by layer fusion, causing slight performance loss.

The bottleneck structure where the thin layer lies is defined as the material structure. Another bottleneck structure in the fusion operation is defined as the carrier structure (described in Figure 1). The challenge of layer fusion is that the impact of losing an entire layer on feature propagation is serious; thus, the carrier structure should undertake the function of the material structure. Therefore, the nearest bottleneck structure of the material structure is chosen as the carrier structure. Normally, carrier structure is the layer before the material structure. Then, the model is fine-tuned to adjust the output of the carrier structure to be similar to that of the material structure, so the carrier structure is functionally equivalent to the two layers before fusion. If x represents the input, and $\mathcal{F}()$ and $\mathcal{G}()$ represent the carrier structure and the material structure, respectively, then the fused layer $\mathcal{H}()$ should function as:

$$\mathcal{H}(x) \quad = \quad \mathcal{G}(\mathcal{F}(x)). \tag{2}$$

There are two key points in layer-fusion fine-tuning: knowledge distillation and short–long fine-tuning.

3.3.1. Knowledge Distillation

Knowledge distillation [44] is a method for transferring knowledge from a complex teacher network to a simple student network. A critical part of knowledge distillation is the soft label, which is a learning objective obtained from the output of the teacher network. The soft label is defined as:

$$q_i \quad = \quad \frac{\exp\left(z_i/T\right)}{\sum_j \exp\left(z_j/T\right)}. \tag{3}$$

Here, z_i is the probability of the ith class and T is the temperature of knowledge distillation. In knowledge distillation, the student network is optimized according to the soft labels and the ground-truth labels to get better training results. Benefiting from knowledge distillation, the student network is trained to generalize in the same way as the teacher network, and the training difficulty of the student network is reduced.

Therefore, in layer fusion, knowledge distillation is introduced as the fine-tuning method. After a material structure is removed, a sub-network with a simpler structure is obtained. The model compressed by dynamic pruning is considered as the teacher network, and the sub-network is considered as the student network. The sub-network is fine-tuned with knowledge distillation after removing one material structure. Fine-tuning with knowledge distillation could optimize the output of the carrier structure to close to the material structure, which means the loss of layer fusion is minimal.

3.3.2. Short–Long Fine-Tuning

In general, training the model optimizes all of the parameters. However, after the material structure is removed, only the carrier structure needs to be fine-tuned. Optimizing the entire network would increase the difficulty of searching for the optimal solution. This part is intended to keep the fine-tuning focused on the output of the carrier structure. Thus, after removing the material structure, only the weights of the carrier structure are tuned, and other layers are frozen. As a result, it is enough to take a small number of epochs for

fine-tuning only the carrier structure of the network, the process of which is denoted as short fine-tuning. The short fine-tuning is done after each material structure is removed.

In addition, it is difficult to fine-tune the output of the carrier structure to be exactly the same as that of the material structure. A tiny offset remains in the fused layers after short fine-tuning. When the offsets accumulate too much, the performance of the model is severely degraded. Thus, after four bottleneck structures are fused, the model is fine-tuned without freezing. This process requires more iterations and is defined as long fine-tuning. Introducing long–short fine-tuning contributes to providing layer fusion with a low performance loss.

3.3.3. Iterative Layer Fusion

Layer fusion is an iterative process. After one bottleneck structure is fused, the next set of material and carrier structures is searched by the new model.

Algorithm 2 describes the flow of layer fusion. In detail, the duplicate of the pruned model always serves as the teacher model for knowledge distillation. First, find the bottleneck structure with the lowest number of channels as the material structure, and select the previous bottleneck structure as the carrier structure. Then, remove the material structure and short fine-tune the model. After repeating these two steps four times, a long fine-tuning is conducted. The above steps are repeated until the drop in accuracy is greater than 3%. Although the architecture is compressed by layer fusion, the model performance does not keep decreasing. When the model is modified to the appropriate architecture, the model's performance increases. Both increases and decreases in model accuracy are possible after each long fine-tuning, so a loose termination condition is adopted. The results after each long fine-tuning are kept, and we evaluate them in terms of compression ratio and performance.

Algorithm 2: Flow of layer fusion.

```
Data: The pruned model
Result: The model with improved architecture
duplicate the pruned model as the teacher model;
while accuracy drop less than 3% do
    for iterations to 4 do
        select the bottleneck structure with the least number of channels as the
          material structure;
        select the previous layer of the element layer as the carrier structure;
        remove the material structure from model;
        freeze weights except for the carrier structure;
        short fine-tuning with knowledge distillation;
    end
    long fine-tuning with knowledge distillation;
    save model;
end
```

4. Experimental

4.1. Experimental Configuration

The proposal was applied to the ResNet and DenseNet models to evaluate the improvement effect. CIFAR10 and ImageNet50 [45] were adopted as experimental datasets. CIFAR10 contains 50k training images and 10k test images, all of which are 32×32. ImageNet50 consists of 50 random classes chosen from the ILSVRC2012 dataset. It contains 51,614 training images, 6490 validation images, and 6440 test images, all of which are 224×224. Common ResNet models, such as ResNet-50 and ResNet-101, are designed for ImageNet, and their architectures are too complex for CIFAR10. ResNet-56 and ResNet-110 have been designed for CIFAR10 with a simpler architecture, and they expect an input size of 32×32. Therefore, ResNet-56 and ResNet-110 have been adopted as the base model for

the CIFAR10 compression experiment. For the same reason, DenseNet-40 was adopted as the base model for the CIFAR10, and DenseNet-121 was compressed in the experiment on ImageNet50.

Each base model was trained on the datasets with a 5-epoch warm-up and 320 epochs, from scratch. The momentum was set to 0.9, the weight decay factor was 10^{-4}, and the batch size was 64. Experiments were conducted on the Nvidia GeForce GTX 3080 Ti GPU and Intel i9-10900 CPU, and the models were implemented by pytorch.

In the dynamic pruning phase, the optimizer SGD with a learning rate initialized to 0.01 was adopted. The learning rate was decayed by cosine annealing [46] with a period of 320 epochs and restarted at epoch 160. The network was fine-tuned on the training set, and the number of epochs was set to 320. When the best accuracy was not updated for more than 20 epochs, the fine-tuning was stopped. About the pruning strategy, only the input channels of second convolutional layers in bottleneck structures were selected as pruning targets. In DenseNet40, since a more dense structure than the bottleneck structure which contains one convolutional layer is adopted, the highly sparse output channels of each convolutional layer and all the corresponding input channels are removed.

In the layer-fusion phase, normally, the carrier structure is the previous bottleneck structure of the material structure. However, layer fusion should not fuse two layers with different sizes of output feature maps. When the output size of the previous bottleneck structure is different, the following one is picked as the carrier structure. The temperature of the knowledge distillation was set to 4. The number of epochs for short fine-tuning was set to 50, and 200 for long fine-tuning. The other settings of fine-tuning were the same as in dynamic pruning.

4.2. Experiments on CIFAR10

The experimental results of dynamic pruning on CIFAR10 are shown in Table 1. Top-1 accuracy and FLOPs are the focuses. The compression effect is noticeable on the ResNet series, especially ResNet110, which compresses 75.75% of FLOPs with a 0.42% drop in precision. In addition, layers with less than six output channels are considered thin layers, and the number of such layers is listed in the table. It can be seen that the number of thin layers after pruning was considerable, especially for ResNet110, which had 31 thin layers. In the next phase, layer fusion was mainly focused on these thin layers.

Table 1. Results of dynamic pruning on CIFAR10. "Acc." indicates accuracy. "Acc. ↓" and "FLOPs ↓" denote reductions compared to the base models. The other tables and figures follow the same conventions.

	Baseline (%)	Pruned Acc. (%)	Acc. ↓ (%)	FLOPs (M)	FLOPs ↓ (%)	Thin Layers
ResNet-56	93.52	93.1	0.42	50.92	60.10	10
ResNet-110	93.76	93.34	0.42	62.34	75.75	31
DenseNet-40	94.53	94.07	0.46	210.56	28.03	10

Table 2 details the experimental results of layer fusion. The effect of layer fusion was most significant on ResNet-56 and DenseNet-40. Compared to the results of dynamic pruning, the compression ratio was improved by 12.01% on ResNet-56, and the accuracy was further reduced by 0.35%. Additionally, FLOPs were further reduced in number by 11.78% in DenseNet-40, along with a further loss of 0.41% in accuracy. For ResNet-110, although only 4.3% of FLOPs were eliminated by layer fusion, up to 64 convolutional layers were fused, and the accuracy reduction was 0.3%. The results of the layer fusion, in order to prioritize the performance, are also listed in the table. It indicates that multiple layers were removed from the models with less than 0.07% in accuracy degradation. Specifically, after fusing the 48 convolutional layers in ResNet-110, the accuracy rose by 0.04% compared to the pruned model.

Table 2. Results of layer fusion on CIFAR10. The depth and fused layers refer to numbers of convolutional layers. ResNet-56* indicates the layer fusion results of the priority selection according to the precision, and same for ResNet-110* and DenseNet-40*.

	Fused Acc. (%)	Acc. ↓ (%)	FLOPs (M)	FLOPs ↓ (%)	Depth	Fused Layers
ResNet-56	92.75	0.77	35.59	72.11	24	32
ResNet-110	93.04	0.72	51.28	80.05	46	64
DenseNet-40	93.66	0.87	176.10	39.81	24	16
ResNet-56*	93.04	0.48	46.85	63.29	40	16
ResNet-110*	93.38	0.38	57.14	77.77	62	48
DenseNet-40*	94.07	0.46	187.54	35.90	28	12

4.3. Experiments on ImageNet50

The proposal achieved satisfactory results on CIFAR10. However, mainstream models are considered too complex for CIFAR10, which means there are plenty of redundant structures that can be easily removed from the model. Therefore, as a complement, the experiments were performed on ImageNet50, which consists of large input images. Since ImageNet50 is more complex compared to CIFAR10, it is challenging to compress the models without accuracy loss. Table 3 provides the results obtained in the experiment. As can be seen in the table, 34.54% of the FLOPs of DenseNet121 were compressed, and there was a 0.65% accuracy reduction. Then, layer fusion improved the compression of FLOPs to 36.48% while resulting in an overall accuracy loss of 0.92%. In detail, 32 convolutional layers of DenseNet121 were fused in the layer fusion.

Table 3. Compression results of DenseNet-121 on ImageNet50. "After layer fusion" indicates the result of layer fusion after dynamic pruning.

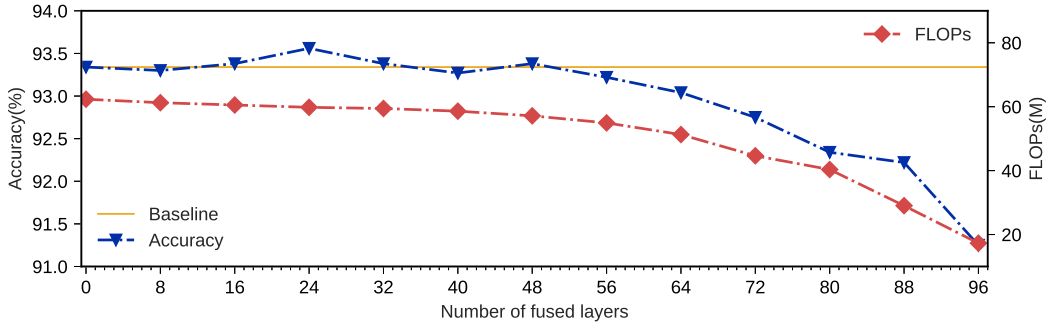
	Acc. (%)	Acc. ↓ (%)	FLOPs (M)	FLOPs ↓ (%)
Baseline	90.21	-	59.2	-
After dynamic pruning	89.56	0.65	38.75	34.54
After layer fusion	89.29	0.92	37.60	36.48

4.4. Analysis

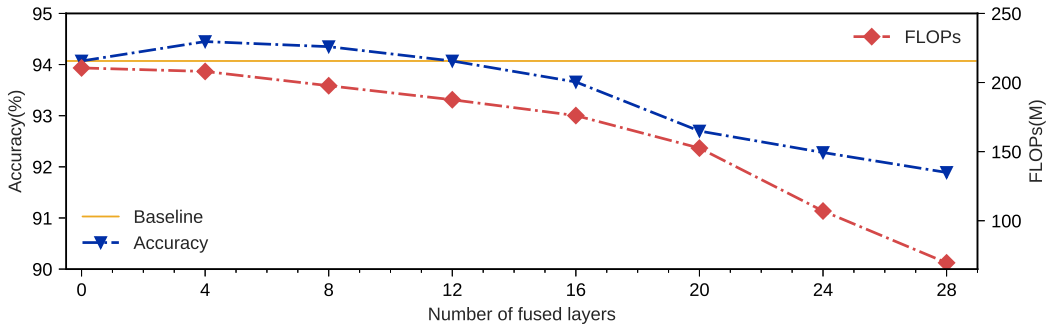
The results of all experiments are summarized in Table 4. In addition, the intermediate results of layer fusion are analyzed. Figure 3 presents the layer fusion details of the experiments on CIFAR10. The baseline is the model's accuracy after dynamic pruning. It can be noticed that the accuracy is not continuously decreasing as the layers are fused. In the layer-fusion experiments on ResNet-110, the accuracy was higher than the baseline four times. Additionally, for DenseNet-40, there was a significant increase in accuracy of 0.38% after the first four layers were fused. Afterward, when the 12 convolutional layers were discarded, the accuracy was the same as the baseline. These data suggest that it is feasible to improve the compressed model's performance by layer fusion. Moreover, the variations in FLOPs are also shown in the figure. It can be found that since the bottleneck structure with the fewest channels was fused, the reduction in computational resources by layer fusion was not significant. However, considering that it was a further reduction of FLOPs from a compressed model, the enhanced compression ratio is valuable.

Table 4. All compression results by our proposed method.

Dataset	Model	Original Acc. (%)	Compressed Acc. (%)	Acc. ↓ (%)	Original FLOPs (M)	Compressed FLOPs (M)	FLOPs ↓ (%)
CIFAR10	ResNet-56	93.52	92.75	0.77	127.62	35.59	72.11
	ResNet-110	93.76	93.04	0.72	257.09	51.28	80.05
	DenseNet-40	94.53	93.66	0.87	292.56	176.1	39.80
ImageNet50	DenseNet-121	90.21	89.29	0.92	59.20	37.60	36.48



(**a**)



(**b**)

Figure 3. Layer fusion details of ResNet-110 and DenseNet-40. The accuracy and FLOPs of the model after all four layers are fused are shown in the figure. (**a**) ResNet-110. (**b**) DenseNet-40.

Moreover, comparison experiments were conducted to analyze the proposal:

- Knowledge distillation was replaced by cross-entropy loss in the fine-tuning.
- No short fine-tuning was performed after each structure was fused; only long fine-tuning was conducted after four layers were fused.
- We trained models from scratch with the optimized architectures.

Figure 4 presents the experiment without knowledge distillation on DenseNet-40. From the figure, it can be seen that without the benefit of knowledge distillation, it is difficult to recover the model accuracy to a satisfactory level. After 20 layers are fused, the model's accuracy degrades more seriously. Additionally, Figure 5 presents the experiment with no short fine-tuning on ResNet-110. After removing short fine-tuning, seven fine-tuning results were worse than before, and the accuracy dropped more severely after 44 layers were fused.

Since the complexity of the model architectures was reduced without a significant drop in performance, we can say that they nearly retain the performance of the complex models even though the architectures are relatively simple. To demonstrate that, models with architectures the same as those of the compressed models were built and then trained from scratch with the same training settings as the base models. The results are listed in Table 5. Acc. improved indicates the accuracy improved by the proposal compared to training

from scratch. These data show that with similar complexity, the models compressed by the proposed method have higher performance.

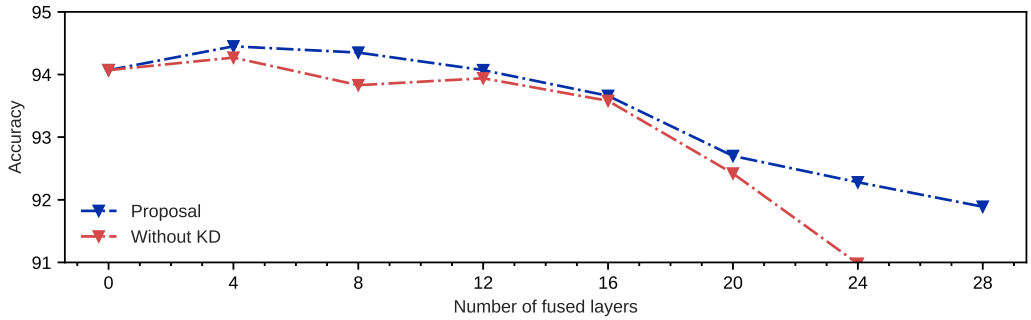


Figure 4. Details of the layer-fusion experiment on DenseNet-40 without knowledge distillation.

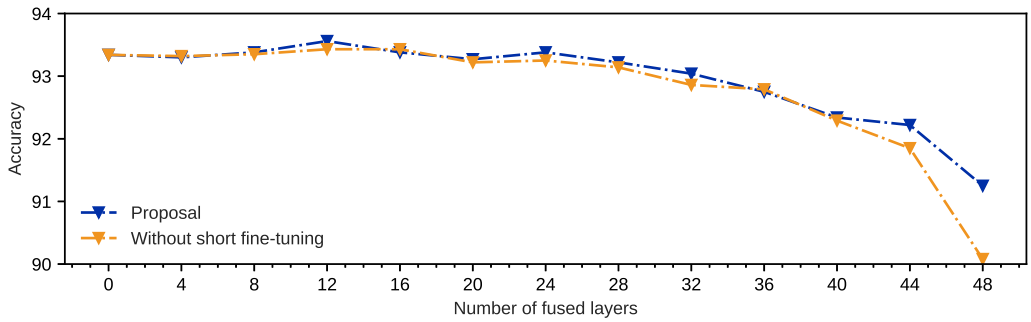


Figure 5. Details of the layer-fusion experiment on ResNet-110 without short fine-tuning.

Table 5. Accuracy of the compressed models and models trained from scratch with compressed architectures.

	Acc. by Proposal (%)	Acc. from Scratch (%)	Acc. Improved (%)
ResNet-56	92.75	91.65	1.10
ResNet-110	93.04	92.31	0.73
DenseNet-40	93.66	93.25	0.41

5. Conclusions

In this paper, we proposed a CNN architecture-improvement approach to optimize redundant models at the channel level and the layer level. First, the binary search method is used to dynamically determine the appropriate pruning threshold, and then redundant channels are removed based on the threshold. Then, bottleneck structures with only a few channels are eliminated by layer fusion to compress the model at the layer level. Knowledge distillation and short–long fine-tuning were introduced to layer fusion to enhance the performance of the fused models. The experimental results show the efficiency of the proposal: in terms of ResNet-56, 72.11% of FLOPs were eliminated, and there was a 0.77% drop in accuracy; as for ResNet-110, 80.05% of FLOPs were eliminated, and there was a drop in accuracy of 0.72%. In detail, the data demonstrate that there are 48 convolutional layers that could be removed from ResNet110 by our method without harming the model. We focused on the analysis of the proposal in the classification task. In future work, the effects of detection and segmentation tasks will be analyzed, since compression is more difficult for these tasks. Furthermore, the models with the optimized architecture are planned to be implemented on FPGAs to evaluate the compression effect on resource-constrained devices.

Author Contributions: Conceptualization, L.M. and H.L.; methodology, Q.L.; software, Q.L.; validation, Q.L. and H.L; formal analysis, Q.L.; investigation, Q.L.; resources, L.M.; data curation, Q.L.; writing—original draft preparation, Q.L.; writing—review and editing, H.L. and L.M.; visualization, Q.L.; supervision, L.M.; project administration, L.M.; funding acquisition, L.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. List of abbreviations and definitions.

Abbreviation	Explanation
FLOP	Floating-point operation
CNN	Convolution neural network
FPGA	Field-programmable gate array
lasso	Least absolute shrinkage and selection operator
$\mathbf{F}_{ij}$	Filter that connects the ith input channel to the jth output channel
$\mathbf{M}_i$	Input feature map on the ith channel
$\mathbf{O}_i$	Output feature map on the ith channel
$\circledast$	Convolution operator
b	Bias
$\mathbf{P}_c$	Current demarcation point of En-sparsity
$\mathbf{P}_u$	Upper endpoint of target interval
$\mathbf{P}_l$	Lower endpoint of target interval
x	Input of bottleneck structure
$\mathcal{F}()$	Carrier structure
$\mathcal{G}()$	Material structure
$\mathcal{H}()$	Fused layer
z_i	The probability of the ith class
T	The temperature of knowledge distillation
q_i	The soft label of the ith class
GPU	Graphics processing unit
CPU	Central Processing Unit
Acc.	Top-1 accuracy
Acc. $\downarrow$	Reduction in accuracy compared to the base model
FLOPs $\downarrow$	Reduction in FLOP compared to the base model
SGD	Stochastic gradient descent algorithm

References

1. Li, Z.; Meng, L. Research on Deep Learning-based Cross-disciplinary Applications. In Proceedings of the 2022 International Conference on Advanced Mechatronic Systems (ICAMechS), Toyama, Japan, 17–20 December 2022; pp. 221–224. [CrossRef]
2. Chen, X.; Liu, L.; Tan, X. Robust Pedestrian Detection Based on Multi-Spectral Image Fusion and Convolutional Neural Networks. *Electronics* **2022**, *11*, 1. [CrossRef]
3. Avazov, K.; Mukhiddinov, M.; Makhmudov, F.; Cho, Y.I. Fire Detection Method in Smart City Environments Using a Deep-Learning-Based Approach. *Electronics* **2022**, *11*, 73. [CrossRef]
4. Yue, X.; Li, H.; Shimizu, M.; Kawamura, S.; Meng, L. YOLO-GD: A Deep Learning-Based Object Detection Algorithm for Empty-Dish Recycling Robots. *Machines* **2022**, *10*, 294. [CrossRef]
5. Ge, Y.; Yue, X.; Meng, L. YOLO-GG: A slight object detection model for empty-dish recycling robot. In Proceedings of the 2022 International Conference on Advanced Mechatronic Systems (ICAMechS), Toyama, Japan, 17–20 December 2022; pp. 59–63.
6. Yue, X.; Li, H.; Fujikawa, Y.; Meng, L. Dynamic Dataset Augmentation for Deep Learning-Based Oracle Bone Inscriptions Recognition. *J. Comput. Cult. Herit.* **2022**, *15*, 76. [CrossRef]

7. Meng, L.; Hirayama, T.; Oyanagi, S. Underwater-drone with panoramic camera for automatic fish recognition based on deep learning. *IEEE Access* **2018**, *6*, 17880–17886. [CrossRef]
8. Deng, M.; Inoue, A.; Shibata, Y.; Sekiguchi, K.; Ueki, N. An obstacle avoidance method for two wheeled mobile robot. In Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, London, UK, 15–17 April 2007; pp. 689–692. [CrossRef]
9. Wen, S.; Deng, M.; Inoue, A. Operator-based robust non-linear control for gantry crane system with soft measurement of swing angle. *Int. J. Model. Identif. Control* **2012**, *16*, 86–96. [CrossRef]
10. Bergerman, M.; van Henten, E.; Billingsley, J.; Reid, J.F.; Deng, M. IEEE Robotics and Automation Society Technical Committee on Agricultural Robotics and Automation. *IEEE Robot. Autom. Mag.* **2013**, *20*, 20–125. [CrossRef]
11. Yue, X.; Lyu, B.; Li, H.; Meng, L.; Furumoto, K. Real-time medicine packet recognition system in dispensing medicines for the elderly. *Meas. Sens.* **2021**, *18*, 100072. [CrossRef]
12. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
13. Daud, S.M.S.M.; Yusof, M.Y.P.M.; Heo, C.C.; Khoo, L.S.; Singh, M.K.C.; Mahmood, M.S.; Nawawi, H. Applications of drone in disaster management: A scoping review. *Sci. Justice* **2022**, *62*, 30–42. [CrossRef]
14. Ghimire, D.; Kil, D.; Kim, S.H. A Survey on Efficient Convolutional Neural Networks and Hardware Acceleration. *Electronics* **2022**, *11*, 945. [CrossRef]
15. Ahamad, A.; Sun, C.C.; Kuo, W.K. Quantized Semantic Segmentation Deep Architecture for Deployment on an Edge Computing Device for Image Segmentation. *Electronics* **2022**, *11*, 3561. [CrossRef]
16. Zhao, M.; Li, M.; Peng, S.L.; Li, J. A Novel Deep Learning Model Compression Algorithm. *Electronics* **2022**, *11*, 1066. [CrossRef]
17. Zhou, H.; Alvarez, J.M.; Porikli, F. Less is More: Towards Compact CNNs. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 662–677.
18. Li, H.; Yue, X.; Wang, Z.; Chai, Z.; Wang, W.; Tomiyama, H.; Meng, L. Optimizing the deep neural networks by layer-wise refined pruning and the acceleration on FPGA. *Comput. Intell. Neurosci.* **2022**, *2022*, 8039281. [CrossRef] [PubMed]
19. Jordao, A.; Lie, M.; Schwartz, W.R. Discriminative Layer Pruning for Convolutional Neural Networks. *IEEE J. Sel. Top. Signal Process.* **2020**, *14*, 828–837. [CrossRef]
20. Yuan, S.; Du, Y.; Liu, M.; Yue, S.; Li, B.; Zhang, H. YOLOv5-Ytiny: A Miniature Aggregate Detection and Classification Model. *Electronics* **2022**, *11*, 1743. [CrossRef]
21. 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.* **2019**, *41*, 2889–2905. [CrossRef]
22. Li, H.; Wang, Z.; Yue, X.; Wang, W.; Hiroyuki, T.; Meng, L. A Comprehensive Analysis of Low-Impact Computations in Deep Learning Workloads. In Proceedings of the 2021 on Great Lakes Symposium on VLSI, GLSVLSI '21, Virtual Event, 22–25 June 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 385–390. [CrossRef]
23. Hao, Z.; Li, Z.; Dang, X.; Ma, Z.; Liu, G. MM-LMF: A Low-Rank Multimodal Fusion Dangerous Driving Behavior Recognition Method Based on FMCW Signals. *Electronics* **2022**, *11*, 3800. [CrossRef]
24. Gong, C.; Chen, Y.; Lu, Y.; Li, T.; Hao, C.; Chen, D. VecQ: Minimal Loss DNN Model Compression With Vectorized Weight Quantization. *IEEE Trans. Comput.* **2021**, *70*, 696–710. [CrossRef]
25. Husham Almukhtar, F.; Abbas Ajwad, A.; Kamil, A.S.; Jaleel, R.A.; Adil Kamil, R.; Jalal Mosa, S. Deep Learning Techniques for Pattern Recognition in EEG Audio Signal-Processing-Based Eye-Closed and Eye-Open Cases. *Electronics* **2022**, *11*, 4029. [CrossRef]
26. Guo, K.; Zeng, S.; Yu, J.; Wang, Y.; Yang, H. [DL] A survey of FPGA-based neural network inference accelerators. *ACM Trans. Reconfigurable Technol. Syst. TRETS* **2019**, *12*, 1–26. [CrossRef]
27. Li, H.; Yue, X.; Wang, Z.; Wang, W.; Tomiyama, H.; Meng, L. A survey of Convolutional Neural Networks —From software to hardware and the applications in measurement. *Meas. Sens.* **2021**, *18*, 100080. [CrossRef]
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
29. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
30. 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.
31. Kuang, J.; Shao, M.; Wang, R.; Zuo, W.; Ding, W. Network pruning via probing the importance of filters. *Int. J. Mach. Learn. Cybern.* **2022**, *13*, 2403–2414. [CrossRef]
32. Li, Y.; Gu, S.; Mayer, C.; Gool, L.V.; Timofte, R. Group sparsity: The hinge between filter pruning and decomposition for network compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8018–8027.
33. Shawahna, A.; Sait, S.M.; El-Maleh, A. FPGA-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access* **2018**, *7*, 7823–7859. [CrossRef]

34. Misra, J.; Saha, I. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing* **2010**, *74*, 239–255. [CrossRef]
35. Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. EIE: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 243–254. [CrossRef]
36. Chen, Y.H.; Emer, J.; Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 367–379. [CrossRef]
37. Hameed, R.; Qadeer, W.; Wachs, M.; Azizi, O.; Solomatnikov, A.; Lee, B.C.; Richardson, S.; Kozyrakis, C.; Horowitz, M. Understanding sources of inefficiency in general-purpose chips. In Proceedings of the 37th Annual International Symposium on Computer Architecture, Saint-Malo, France, 19–23 June 2010; pp. 37–47.
38. Keckler, S.W.; Dally, W.J.; Khailany, B.; Garland, M.; Glasco, D. GPUs and the Future of Parallel Computing. *IEEE Micro* **2011**, *31*, 7–17. [CrossRef]
39. Chen, S.; Zhao, Q. Shallowing Deep Networks: Layer-Wise Pruning Based on Feature Representations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 3048–3056. [CrossRef]
40. He, Y.; Zhang, X.; Sun, J. Channel Pruning for Accelerating Very Deep Neural Networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
41. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
42. Huang, Z.; Wang, N. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv* **2017**, arXiv:1707.01219.
43. Aghli, N.; Ribeiro, E. Combining weight pruning and knowledge distillation for cnn compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 3191–3198.
44. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
45. CIFAR-10 and CIFAR-100 Datasets. Available online: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 3 October 2021).
46. Loshchilov, I.; Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv* **2016**, arXiv:1608.03983.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.