INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 1, PP. 87–94
Manuscript received November 15, 2023; revised March, 2024. doi: 10.24425/ijet.2024.149518

Architecture optimization techniques for Convolutional Neural Networks: further experiments and insights

Artur Sobolewski and Kamil Szyc

***Abstract*—In this paper, we have researched implementing convolutional neural network (CNN) models for devices with limited resources, such as smartphones and embedded computers. To optimize the number of parameters of these models, we studied various popular methods that would allow them to operate more efficiently. Specifically, our research focused on the ResNet-101 and VGG-19 architectures, which we modified using techniques specific to model optimization. We aimed to determine which approach would work best for particular requirements for a maximum accepted accuracy drop. Our contribution lies in the comprehensive ablation study, which presents the impact of different approaches on the final results, specifically in terms of reducing model parameters, FLOPS, and the potential decline in accuracy. We explored the feasibility of implementing architecture compression methods that can influence the model's structure. Additionally, we delved into post-training methods, such as pruning and quantization, at various model sparsity levels. This study builds upon our prior research [1] to provide a more comprehensive understanding of the subject matter at hand.** [1]

***Keywords*—Model Compression; Convolutional Neural Network; Computer Vision**

I. Introduction

ADAPTING computer vision to edge devices such as mobile phones or small computing platforms is imperative in today's technological landscape [2]. However, once we have a working computer vision model, adapting it to these edge devices is a significant challenge. Given the constraints and limited resources of these devices, it is critical to understand the intricacies of efficiently achieving this adaptation.

Here, we discuss how to adapt convolutional neural networks (CNNs) [3] effectively to edge devices. Many different solutions have been proposed in the literature. For example, there are popular architectures that have introduced many new approaches, such as MobileNet [4], ShuffleNet [5], or GhostNet [6]. Each of them affects the final results, particularly in terms of reducing model parameters, FLOPS, and the potential loss of accuracy. Our goal was to determine which approaches work best for specific requirements on the maximum acceptable loss of accuracy.

A. Sobolewski and K. Szyc are with Wrocław University of Science and Technology, Wrocław, Poland (e-mail: artur.sobolewski17@gmail.com, kamil.szyc@pwr.edu.pl).

[1]All results are fully reproducible, the source code is available at https://github.com/artur-sobolewski/CNN-compression

In this paper, we extend our previous study. In [1], we tested how different approaches designed for architecture optimization techniques affect the model. We started with the classic ResNet [7] architecture trained on CIFAR-10 and VGG [8] trained on CIFAR-100. Next, we added different optimization techniques and monitored the assumptions tested in our ablation study. Now, we extended our research by testing different (than previous) model-dataset pairs with changed training methodology and adding a new tested optimization method, i.e., DiCE unit [9]. Both of our studies allowed us to choose the right strategy for adapting computer vision models to edge devices.

The paper is organized as follows. Section II delves into related work, providing an overview of the approaches under investigation. Section III delves into the details of our implementation methodologies for these techniques and outlines the results we obtained in our experiments. Section IV serves as a summary, encapsulating the methods we implemented and drawing conclusions about their suitability in light of our research results.

II. Related works

In the literature, ways to optimize computer vision models often involve post-training methods [10]. Turning off some model parameters or reducing their size effectively simplifies the computations. The other solution consists of changes at the architecture level [11]. These are usually invented along with the proposal of a new model. This makes it difficult to assess their overall capabilities, and we see the lack of ablation studies that contribute to this knowledge, or they are focused only on optimizing a particular model adjusting pre-selected mechanism [12].

In this study, our primary focus has been on the CNN architecture [13]. One of the most common approaches to deploying these networks on edge devices is to modify the architecture itself. The authors of GoogLeNet [14] first discovered the potential of 1x1 kernel size convolution in dimensionality reduction, which helped to reduce the computational requirements and limit the size of a model. The pioneer work on the topic of this paper is SqueezeNet [15], which further developed this mechanism and proposed the fundamental element of its architecture – the Fire module.

This unit reduces (”squeezes”) the dimensions of the feature map before performing more expensive operations such as 3x3 kernel size convolution, which results in reducing the number of filters and thus computations. For this ”squeeze” part, they utilized the mentioned low-cost 1x1 convolution. They also used this type of convolution in the standard filtering CNN pipeline to compute half of the module output in the ”expand” part when the second half is obtained with a 3x3 convolution.

The standard of modern CNNs uses the Global Average Pooling (GAP) [16] layer. In the earlier CNN development approaches the final activation map was processed directly by fully connected (FC) layers of the classifiers, which significantly increased the number of parameters. This type of pooling takes the average of each channel in the feature map to create a vector. Applying GAP layers with a final linear layer typically replaces several. This simple change significantly improves model optimization and reduces the risk of overfitting, to which the classifier’s FC layers are particularly prone. In the original Network in Network paper [16], the authors proposed an architecture that feeds the resulting vector directly into the softmax layer without fully connected layers at all. This solution, in addition to parameter reduction, is intended to make the feature extraction part of the network better suited to the task than when the FC layer is largely responsible for classification.

For further optimization, some papers suggest solutions that focus on low-level modifications. The proposed Inception v3 [17] model uses a factorization of the standard convolution, replacing a single nxn filter with two successive nx1 and 1xn filters. Despite a single 3x3 kernel, containing 9 parameters, the model stores 3x1 and 1x3 for a total of 6 parameters. This modification reduces the number of parameters by 33% for the convolution layer with 3x3 filters, and the savings increase drastically with the size of n. This asymmetric factorization of the convolution maintains the same feature map size, receptive fields, and similar efficiency, with a noticeable optimization effect.

Another technique for optimizing individual layers is depthwise separable convolution (we sometimes refer to it as DWconv). The first model to popularize the use of this solution was Xception [18]. This solution assembles two successive convolutions: depthwise and pointwise. The authors proved that it is possible to map cross-channel correlations (by pointwise convolution) and extract spatial features separately. Furthermore, it is possible to perform this filtering on non-overlapping channel-wise segments of an input, which is called depthwise convolution. This solution is a significant step towards reducing the number of operations. The main incentives for this approach are the lower computational complexity of the feature filtering and the cheap control over the size of the feature maps.

The authors of ShuffleNet [5] further developed the depthwise separable convolution. Despite the already good efficiency of the previous version, the fact that each filter of the pointwise convolution layer operates on all channels of the feature map is the main reason for the high resource consumption. The implemented idea was to use a convolution grouping mechanism. They used pointwise convolutions performed only on a subset (group) of channels. Depthwise convolutions were also performed in groups, but first, the channels of the resulting pointwise grouped convolution feature map were shuffled by a separate layer. This mechanism ensures that potentially important channel-wise feature correlations are not restricted to a particular segment but are accessible in all groups. This approach significantly reduced the number of parameters and computations. We will refer to it later as grouped pointwise convolution.

One of the most popular deep neural network architectures designed specifically for efficient resource utilization is MobileNet [19]. The primary goal of the authors in this paper was to develop a complete class of configurable network architectures that take into account latency optimization and potentially high accuracy. A small size of a model was achieved by using the already mentioned DWconv. A more effective modification was proposed in the second version of this architecture – MobileNetV2 [20]. In terms of our research, it uses residual bottleneck blocks, what ResNet [7] proposed. However, unlike the original, the authors evolved this mechanism by modifying its connections. They called them ”inverted residual bottlenecks”, which now maintain narrow connections between bottlenecks and expand the channels of a feature map to inner filtering with a 3x3 kernel.

Some approaches look for ways to optimize a model by making minor structural changes in certain structural elements of the network. In the MobileNetV3 paper [4], the researchers found that spatial filtering could be omitted in the last bottleneck block. Because the feature map had already been expanded by its pointwise convolution, they decided to use the output close to its final form and feed it directly into the Global Average Pooling layer. This obviates the necessity for costly channel expansion for output generation. The first layer is also one of the slowest layers due to the massive computations required to perform the convolution at the highest resolution of the image. It has been experimentally proven that there is no need for a high number of filters in the first stage of processing, and it has a marginal impact on the final result.

The idea of splitting the map of internal features and performing calculations on them simultaneously was further taken up by the authors of DiCENet [9]. Like the well-known depthwise separable convolution, the basis of the proposed mechanism was to apply this lightweight feature extraction but in a more efficient way to overcome the computational bottleneck that this block still contains. This is due to the linear fusion of channel-wise features by point-wise convolution. The proposed DiCE unit consists of two parts: DimConv and DimFuse. For DimConv, the output is a concatenation of three simultaneous convolution layers. In each layer, the filters work dimension-wise in separate groups. Thus, the input tensor is locally filtered channel-wise, width-wise, and height-wise, respectively. In the DimFuse part, the dimension-wise features are first integrated using a shuffle mechanism and grouped point-wise convolution. This ensures local correlation of the features. Finally, global fusion is performed by depthwise convolution, which integrates spatial information. Additionally, the authors used the Squeeze-and-Excitation (SE connection) [21] block to weight channels according to their importance.

DimFusion is expected to be a much more efficient alternative to the popular pointwise convolution for computing channel-wise relationships.

Expansive Convolutional Neural Networks typically provide a better understanding of the problem thanks to a complex hierarchy of extracted feature maps. However, the authors of GhostNet [6] found that there is often redundancy in the feature maps obtained at different stages of image processing. This fact creates the possibility of obtaining a single channel by performing an inexpensive operation on another channel. Capitalizing on this observation, they decided to increase the efficiency of the network by using simple affine operations to create potentially redundant channels. The proposed ghost module allows to replace of standard filtering layers. The first step is to obtain the internal feature map with a small number of channels. This is usually done by pointwise convolution. Then, depthwise convolution is used to create new "ghost" feature channels. The module output is a tensor formed by concatenating both the ghost features and the preceding unmodified features. We used complete components (ghost bottlenecks) or separate ghost modules.

Research on convolutional networks often focuses on developing solutions that lead to better model performance in various tasks. One of the proposed solutions is the attention mechanism, which implements the previously mentioned Squeeze-and-Excitation block. This approach has been used in modern and efficient architectures such as MobileNetV3 and DiCENet or EfficientNet [22], which optimize computational complexity. Drastic model reduction usually leads to an unwanted loss of accuracy. Therefore, we look for a possible model improvement with the SE connection. This mechanism consists of GAP and other linear layers and activation functions. The resulting vector is expected to learn the relevance of a given channel of the input feature map and adjust its relevant parts. In this way, the model is guided to construct meaningful representations of an image and overall accuracy.

After the model has been trained, it is possible to apply methods for model optimization. Among the most commonly used approaches are pruning and additional quantization [10]. Pruning consists of discarding components that contribute the least to the final prediction result by setting their value to 0. This makes the network sparse, which does not actually reduce the model size but speeds up computation. Quantization involves reducing the precision of the parameters, which are typically represented as 32-bit floating-point integers, by changing them to 8-bit integers.

III. Research on Compression Methods

This section describes experiments that extend our study of architecture optimization techniques, which we presented in paper [1]. We have included the necessary implementation details of selected methods presented in Section II, indicating a possible way to perform the optimization.

In this article, we evaluate how the selected methods affect the final performance of models based on CNN architectures well-known in the literature. For this purpose, we chose ResNet-101 and VGG-19. We trained them on popular image classification benchmarks. To extend previous experiments, we trained ResNet-101 on CIFAR-100 and VGG-19 on CIFAR-10 in this research. Both datasets contain 50,000 images in the training subset and 10,000 images in the test subset. Each image has a resolution of 32x32 pixels in the RGB color model, and they are equally distributed in 100 and 10 classes, respectively. These different scenarios, with previous results, were intended to provide a more comprehensive view of the effectiveness of the methods, taking into account different cases of data. As before, the choice of models was motivated by their well-studied architecture in the literature, which primarily does not include components that reduce computational complexity.

The main part of the experiments deals with model architecture changes, which results were presented in tables I and II. Each approach was tested for changes in the following:

– the number of parameters expressed in millions and as a percentage of the baseline;
– the number of FLOPS expressed in millions and as a percentage of the baseline;
– the total size of the model after saving, expressed in megabytes;
– the final accuracy achieved by the model.

In the second part, we have chosen three models on different levels of optimization, on which we tested iterative pruning with fine-tuning. Additionally, we performed quantization on those 3 unpruned and pruned models (Table III).

To investigate the application of the described methods, we assumed that we want to keep the significant network properties the same when optimizing the model. The general evaluation of selected approaches was possible by replacing the primary computational units, e.g., standard 3x3 convolutions or bottleneck blocks, but not the high-level CNN processing pipeline. Thus, the depth (number of filter layers), the resolution (width and height) of each feature map of the model, and the number of channels in the activation tensor are preserved if not required by the applied solution. Considering this, choosing a specific compression method depends on the unique characteristics of individual CNN architectures.

When reworking the ResNet-101 model, we took into account its layers, which assemble convolution units with the same hyperparameters, i.e., the number of filters and the shape of the processed feature map. The high-level structure and its bottleneck blocks with residual skip connections were preserved as much as possible. In the case of the VGG modification, as in the previous research, its main features were not changed. It does not contain skip connections or advanced block structures such as residual bottlenecks, so we decided not to change the main pipeline.

Since the last experiments, we changed the training procedure. Again, we utilized the SGD optimizer, but this time, we used the "1cycle" learning rate policy [23] with a maximum number of epochs equal to 150. In short, it allows to obtain representative results with a fast model convergence. This would allow for a broader view of the multi-level optimization problem overview. Previously, we used Reduce-LR-On-Plateau [24] and an early stopping mechanism.

TABLE I
OPTIMIZATION TECHNIQUES FOR RESNET-101 TRAINED ON CIFAR-100

Approach \ No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fire modules	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
No FC classifier	-	-	-	-	-	-	✓	✓	✓	-	✓	✓	✓	✓	-
Nx1 - 1xN Conv.	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-
DWconv	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Grouped pointwise conv.	-	-	-	-	✓	-	-	-	-	-	-	-	✓	✓	✓
Inv. bottlenecks	-	-	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Optimization before GAP	-	-	-	-	-	-	-	✓	✓	-	✓	✓	✓	✓	-
Less channels on the early stage	-	-	-	-	-	-	-	-	✓	-	✓	✓	✓	✓	-
Ghost bottleneck	-	-	-	-	-	-	-	-	-	✓	✓	✓	✓	✓	-
SE connections	-	-	-	-	-	-	-	-	-	-	-	✓	-	✓	✓
DiCE unit	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
Parameters [M]	42.697	59.245	35.582	21.425	7.721	2.439	1.233	0.919	0.915	0.458	0.408	1.432	0.159	1.182	3.007
Parameters [%]	100.00	138.76	83.34	50.18	18.08	5.71	2.89	2.15	2.14	1.07	0.96	3.35	0.37	2.77	7.04
FLOPS [M]	2520.41	3480.20	2141.77	1279.72	420.93	79.39	63.14	58.11	53.75	32.83	29.89	31.61	12.93	14.65	160.40
FLOPS [%]	100.00	138.08	84.98	50.77	16.70	3.15	2.51	2.31	2.13	1.30	1.19	1.25	0.51	0.58	6.36
Size [MB]	163.47	226.12	136.34	82.33	30.06	9.66	5.04	3.83	3.81	1.99	1.79	5.73	0.85	4.78	12.15
Accuracy [%]	67.51	70.81	71.19	62.80	67.32	67.86	67.18	66.52	66.41	65.65	65.49	66.35	59.92	61.45	63.58

A. *ResNet-101 on CIFAR-100*

In our experiments, we tested 14 cases of modifications to the ResNet-101 architecture, shown in Table I. Column 1 shows the results obtained for the baseline. The remaining columns show individual experiments, the approaches used in them, and their measured performance. In this subsection, we provide essential information about the application of selected methods in the ResNet-101 architecture and comment on their impact on this model, also in light of our previous research.

a) Fire Module: As in the SqueezeNet paper, we replaced all spatial filtering units (in this case, full bottleneck blocks) with Fire modules, except for the first convolution. We extend the original Fire module by adding a skip connection to preserve the main properties of this model. The "squeeze" part reduces the number of dimensions of the input tensor. This causes the 3x3 convolution layer to operate on a feature map that is 4 times smaller than in the baseline bottleneck blocks. The total number of 1x1 and 3x3 convolution filters of the "expand" part was the same as that of the inner 3x3 convolution in the ResNet-101 bottleneck. This implementation significantly increased the complexity of the model. The improvement in model accuracy may be due to its increased capacity. This solution may be outdated and unsuitable for optimizing newer architectures such as ResNet. Therefore, we have discontinued further use of the Fire module on this model.

b) Nx1 – 1xN Convolution: Replacing the 3x3 convolution with two successive 3x1 and 1x3 convolution reduces the size and FLOPS of ResNet-101 by $\approx$15%. Experiment no. 2 in Table I also shows a noticeable increase in accuracy, which contradicts previous results of our study. We believe that this is due to the fact that limiting the number of training iterations with other hyperparameters of this process was not sufficient to obtain optimal classification accuracy in the case of the baseline model.

c) Depthwise Separable Convolution: Splitting the filters into all 3x3 convolution layers to filter on each channel of the input tensor separately effectively halved the model parameters and FLOPS. This method proved to be effective in terms of model reduction but significantly reduced its accuracy. This degradation is much more significant than we observed for this version of the model trained with the previous strategy on CIFAR-10 and for the other, even more compressed models. A possible reason for such discrepancies is the different requirements for the learning process. To keep the research consistent with our previous work, we used this type of convolution with other methods that gave us better model effectiveness than in this case.

d) Grouped pointwise convolution: We used grouped pointwise convolution with channel shuffling to replace each 1x1 convolution present in the ResNet-101 architecture and the ghost bottleneck experiments. We set the *group* parameter to 4, based on the value we observed in the ShuffleNet paper and the number of channels. The combination of this method and DWconv reduced the computational complexity of the model to 16.7% of the baseline FLOPS and 18.08% of the parameters. With this solution, we were able to reduce the parameters by a factor of 3, even when they were already small, as shown in experiments no. 11 and 13 (Table I). We can highlight the second one as the highest achieved optimization of the ResNet-101 model with $\approx$ 269-fold parameter reduction. Unfortunately, such a high degree of compression is associated with a significant drop in classification efficiency, the largest of all obtained. We speculate that this may be due to the inefficiency of this mechanism in generating intermediate representations and the negligible number of trainable parameters. These conclusions are supported by experiment 14, in which the use of the attention mechanism did not lead to a sufficient improvement in accuracy for the increased model capacity. These observations are consistent with those of a previous study.

e) Inverted Bottlenecks: This relatively simple approach, changing the number of 1x1 projection convolution filters – the last in the bottleneck block – in addition to a large reduction in size, allowed to maintain a fairly high efficiency. From the original number of channels of feature maps in the links of these blocks (256, 512, 1024, 2048), we changed to 10, 22, 42, and 86. We adjusted this ratio based on the structure of the MobileNetV3 model [4]. Based on experiments 6, 7, and 8, we can see that most of the remaining parameters come from the FC layer and the final expansion of channels to 2048 (this was added in experiment 6 to preserve the original structure of the ResNet-101 classifier). We can see a slight increase in accuracy compared to the unoptimized model. In the paper [1] we achieved only 1.24 pp (percentage points) accuracy loss on CIFAR-10.

f) Optimization at early and late model stages: Contrary to the results obtained in the previous study, optimizations in the last part of the CNN architecture led to larger decreases in accuracy. We suggest that the complete removal of the fully connected layers of the classifier and the optimization before the GAP layer may not be appropriate for more complex problems. Both resulted in a decrease in accuracy of 1.34 pp, while the corresponding models in our previous research decreased by only 0.08 pp. We find that this is only worth considering in the case of strong compression. These modifications resulted in much smaller efficiency losses when the model was already optimized to about 1% of the baseline (losses in the range of 0.16-0.27 pp in the case of ResNet-101). But in a shallow network like VGG-19 and a more demanding task like CIFAR-100, the loss reached almost 2 pp. Reducing the number of filters in the first convolution usually reduces the effectiveness slightly.

g) Ghost Bottlenecks: We replaced the original ResNet bottleneck blocks with ghost bottlenecks consisting of 2 previously mentioned ghost modules (the first expands the channels; the second reduces them) and a skip connection. For downsampling, we used extra DWconv with stride = 2 in between. We conducted this and subsequent experiments using the inverted bottleneck approach because of its high effectiveness and compatibility with other solutions. For the ResNet-101, this solution allowed for the highest reduction to 0.37% of baseline parameters and 0.51% of FLOPS (Experiment no. 13 from Table I – the ghost bottlenecks with shuffle mechanism), which significantly reduces the accuracy. Besides, the ghost bottlenecks are the most suitable for the ResNet architecture in terms of optimization objectives.

h) Squeeze-and-Excitation: We used this channel-wise attention mechanism in experiments where the architectures were highly compressed – for no. 11 and no. 13 (Table I). Our implementation of this unit consists of two linear layers, the ReLu activation function and the sigmoid on its output. Our goal was to improve the filtering of the second ghost module by adjusting the channels of the extended feature map at its input. Both (experiments no. 12 and no. 14) resulted in higher model capacity, which we expected. SE connections increased the parameters by about 1M, but with a small impact on the number of computations (FLOPS) – only 0.06-0.07 pp. In order to our previous work, it improved accuracy by still that gains are not commensurate with the increased parameters of a model.

i) DiCE unit: In this research, a new mechanism of dimension-wise separable convolution and cheap spatial and channel-wise feature integration implemented in the DiCE unit was investigated. In the ResNet model, we replaced the previous 3x3 convolution layers and subsequent 1x1 convolution layers of all bottleneck blocks with this unit. This module contains grouped depthwise and pointwise convolutions with channel shuffling and SE connection, which we showed in the table I in experiment no. 15. Note that the degree of optimization (7.04% of baseline parameters and 6.36% of baseline FLOPS) was lower than in experiment no. 6 (5.71% and 3.15%, respectively), which also used narrow block connections, but with the standard depthwise separable convolution. It also caused higher accuracy losses. In our opinion, the reason for this could be that our learning algorithm is inadequate, as we observed in Experiment 4.

Under the same learning conditions, we observed a much more stable learning process for models no. 7, 8, 9, 11, 12, 13, and 14, based on the loss function on the validation subset.

B. VGG-19 on CIFAR-10

In accordance with our previous studies [1], we trained 12 versions of the VGG model on the CIFAR-10 dataset, as well as the implementation case for the new solution – DiCE unit. Differences in the research strategy of selected methods on the VGG model are due to the assumptions made and the characteristics of its architecture. Their effectiveness was evaluated using the same measures as in the case of ResNet-101. All of them are presented in Table II.

a) Global Average Pooling: The VGG is an older type of model that relies heavily on fully connected layers in the final stages of processing in various prediction tasks. As a first step in our research, we enhanced the underlying architecture by replacing two of the original fully connected layers with a global average pooling layer while retaining a single layer dedicated to classification. This change reduced the parameters to 51.44% of the baseline with a decrease in accuracy of only 0.14 pp. Previously, on CIFAR-100, the decrease was also marginal (0.18 pp). We retained this change in further experiments.

TABLE II
OPTIMIZATION TECHNIQUES FOR VGG-19 TRAINED ON CIFAR-10

Approach \ No.	1	2	3	4	5	6	7	8	9	10	11	12	13
GAP	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Nx1 - 1xN Conv.	-	-	✓	-	-	-	-	-	-	-	-	-	-
DWconv	-	-	-	✓	-	✓	✓	✓	✓	✓	✓	✓	✓
Fire modules	-	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓
No FC classifier	-	-	-	-	-	-	✓	✓	✓	✓	✓	✓	-
Grouped pointwise conv.	-	-	-	-	-	-	-	✓	-	✓	✓	✓	✓
Ghost modules	-	-	-	-	-	-	-	-	✓	✓	✓	-	-
SE connections	-	-	-	-	-	-	-	-	-	-	✓	✓	✓
DiCE unit	-	-	-	-	-	-	-	-	-	-	-	-	✓
Parameters [M]	38.959	20.040	13.103	2.285	1.801	0.474	0.474	0.151	0.397	0.342	1.542	1.351	0.261
Parameters [%]	100.00	51.44	33.63	5.87	4.62	1.22	1.22	0.39	1.02	0.88	3.96	3.47	0.67
FLOPS [M]	417.65	398.74	256.03	46.76	40.22	10.99	11.00	4.80	9.43	8.55	10.05	6.31	7.33
FLOPS [%]	100.00	95.47	61.30	11.20	9.63	2.63	2.63	1.15	2.26	2.05	2.41	1.51	1.76
Size [MB]	148.70	76.53	50.07	8.80	7.03	1.96	1.96	0.73	1.67	1.46	6.06	5.32	1.26
Accuracy [%]	92.83	92.69	92.72	89.13	90.81	87.91	87.74	81.71	87.09	86.66	87.22	83.65	66.93

b) Replacement of convolutions: In experiments 3 and 4 (Table II), we replaced all 3x3 convolutions with two successive 3x1 and 1x3 convolutions, while in the second, we replaced the depthwise convolution with a successive projecting 1x1 pointwise convolution. Both reduced the complexity of the baseline with GAP, but as before, the 3x1 - 1x3 convolution underperformed the depthwise separable convolution in terms of compression. However, this resulted in a much larger loss of efficiency (by 3.56 pp of the accuracy of model 2), while the 3x1 - 1x3 convolution actually increased the accuracy of its previous version. The effectiveness of these methods is similar to that found in our previous study.

c) Fire Modules: The Fire module proved to be an efficient solution for optimizing the VGG architecture. We preserved the original width of the model in the connections of these units, and the internal filtering by 3x3 and 1x1 convolutions was performed on a tensor with 8-fold reduced channels. These modules produced a model more than 11 times smaller than model no. 2 with a loss in accuracy of about 2 pp relative to the baseline. After applying DWconv instead of the standard 3x3 convolution in the ”expand” part (experiment no. 6 in table II), in addition to a stronger reduction of the model, the effectiveness of the model decreased by 4.91 pp, showing a similar trend as we obtained in the paper [1]. The Fire modules provide opportunities for further optimization with other techniques, so they were used in all other cases.

d) No FC classifier: Experiment no. 7, where we completely discarded the fully connected layer of the classifier, showed that the accuracy losses are highly related to the difficulty of the problem to be solved. It decreased the accuracy by 0.17 pp, while on the CIFAR-100 this version of VGG worsened the effectiveness by 1.97 pp. It also shows that for a small number of classes in a dataset, such a change may not provide a significant benefit in terms of optimization and accuracy. Effective optimization occurs only by reducing the size of the final feature map before the GAP layer, which was necessary in the case of the ResNet-101 model.

e) Grouped pointwise convolution: The highest optimization we achieved was after applying the grouped pointwise convolution with channel shuffling mechanism to the previous version of the model. We replaced all 1x1 convolution with this unit. To ensure that the information is not limited in separate groups, we performed channel shuffling on the concatenated tensor computed by an ”expand” part. This resulted in a model with 0.151 M parameters and 1.15 M FLOPS, but with the lowest accuracy (experiment no. 8 in Table II), as in the case of our previous study.

f) Ghost modules: We used the ghost modules proposed in the GhostNet paper to replace the 1x1 convolution of the ”squeeze” part and the further 3x3 convolution. The other version (no. 10) performs this first projection step using the cheap pointwise convolution with shuffling. In both cases, the optimization effect is similar to our first research, but for the second model with the grouped pointwise convolution, this time we do not observe the accuracy improvement.

g) Squeeze-and-Excitation: By applying the SE connection after each Fire module, we were able to improve the performance of the model and thus its accuracy. It is worth noting that despite the increased number of parameters, the number of FLOPS was at a much lower level. However, the other solutions with a similar reduction of parameters achieved higher accuracy.

h) DiCE unit: In the configuration no. 13, a ”squeeze” convolution layer with filter grouping and shuffle mechanism was implemented, while the 3x3 ”expand” convolution was replaced by a DiCE unit. These changes resulted in a signif-

TABLE III
PRUNING TECHNIQUE FOR RESNET-101 TRAINED ON CIFAR-100

Sparsity [%]	no. 1 - baseline		no. 11		no. 13	
	Non-zeroed	**Acc [%]**	**Non-zeroed**	**Acc [%]**	**Non-zeroed**	**Acc [%]**
0.0	42 697 380	67.51	408 035	65.49	158 815	59.92
20.0	34 114 509	67.44	310 870	65.10	111 609	59.37
35.9	27 332 567	66.76	248 696	64.35	89 287	57.60
48.7	21 907 014	65.73	198 957	63.31	71 430	54.12
58.9	17 566 571	65.52	159 166	61.95	57 144	48.56
67.1	14 094 217	64.53	127 333	59.15	45 715	41.86
73.7	11 316 334	63.37	101 866	56.14	36 572	35.88
78.9	9 094 027	61.34	81 493	52.13	29 258	29.64
83.1	7 316 182	59.69	65 194	47.47	23 406	24.25
86.5	5 893 906	56.39	52 155	43.93	18 725	18.39
89.1	4 756 085	53.46	41 724	39.21	14 980	13.14

icant reduction in model complexity, but the loss of accuracy was extremely high. After observing the loss function and the changes in model efficiency during learning, we noticed a much slower convergence to the optimal solution. Based on our results and the training conditions studied by the authors of the DiCENet paper, we considered that this architecture might need to be trained much longer or with a higher learning rate.

Related to our previous study, the further investigation of architecture-based optimization techniques has led to consistent conclusions about their effects. Moreover, the different training conditions allowed to observe their requirements on this process. Approaches such as dimension-wise separation of convolution or strong reduction of model capacity (e.g., ghost modules/bottlenecks) require more attention to the training strategy.

C. Post-Training Compression

We performed an iterative, post-training, unstructured pruning with L1 norm and amount equal to 0.2. After each subsequent pruning of the remaining parameters, we fine-tuned the model with SGD and constant LR = $1e-4$, and early stopping (we limited the maximum learning time to 50 epochs). Table III shows the results obtained by pruning the ResNet-101 baseline and 2 other versions with 0.408 M trainable parameters (no. 11) and 0.159 M (no. 13). As in the previous research, we see an increasing decrease in accuracy with increasing model sparsity. In the more complex classification problem – CIFAR-100 – the losses grow faster.

Post-training static quantization was applied to the same 3 models without pruning and with a sparsity rate of 58.9%. In PyTorch's implementation of this mechanism, it was necessary to modify the model code by adding an extra quantization and dequantization layer to encode the input and decode the output of the model, and the same for other operations such as concatenation or addition. The unedited version of the models reduced their accuracies to 58.53%, 64.42%, and 58.57% for the baseline, no. 11, and no. 13, respectively. Quantization of these pruned models resulted in a decrease to 54.83%, 60.66%, and 46.53%, respectively.

The losses caused by the amount of pruning are highly dependent on the number of parameters on which the prediction is based. The relationship between the quantization and the resulting loss of accuracy remains unclear. We obtained similar decreases for models with significantly different numbers of parameters. The use of an 8-bit data representation is commonly supported by hardware vendors to accelerate computations. In the context of deploying a Convolutional Neural Network (CNN) model on edge devices, optimizing memory utilization and power consumption becomes imperative. As a result, reducing the size of the model becomes critical.

IV. SUMMARY

The goal of our study was to evaluate the impact of the compression techniques used in the architecture on CNN models, as explained in section II. We investigated the applicability of these techniques to a variety of models, including ResNet-101 and VGG-19. These approaches can lead to more optimized models with fewer parameters, resulting in faster computations and lower FLOPS.

Considering both our results from this paper and the previous one [1], we came to the following conclusions. We can observe different degrees of the highest possible optimization, assuming a maximum loss of accuracy of 5 pp compared to the baseline. For ResNet-101 trained on CIFAR-10, we were able to reduce the number of parameters and FLOPS by 281 and 196 times, respectively, and for CIFAR-100 by 104 and 84 times. For VGG-19 fitted on CIFAR-10, we reduced the number of parameters by 82 times, the FLOPS by 38 times, and CIFAR-100 by 21 and 10 times, respectively. All results were calculated before the post-training compression. This clarifies the interplay between the architectural backbone, the task complexity, and the potential scope for optimization. It has come to our attention that the selected solution may necessitate a tailored approach to the learning process.

After conducting our research, we have taken into consideration the findings and more general conclusions. Both of our studies have significant overlap in results despite changes in model-dataset pairs and modifications in learning. Although it is difficult to point to a universal solution to this optimization problem, we recommend the following. First, we suggest using methods commonly used in modern CNN architectures, such as GAP and Depthwise Separable Convolution, and combining them with more generic approaches, for example, inverted bottlenecks. This approach can effectively reduce the number of parameters while maintaining a similar level of accuracy. Our research suggests that both high-level building blocks, especially ghost bottlenecks, and low-level modifications, such as the shuffle mechanism, can also yield positive results. The DiCE unit was not suitable for our strict training process. We also recommend carefully optimizing critical parts of architectures, such as the classifier. For more complex problems, the complete removal of the last fully connected layers results in a more noticeable model reduction at the expense of a higher loss of accuracy. Pruning proved to be effective even for highly optimized models, while quantization may only be suitable for less compressed models - otherwise, the losses may be too high.

References

[1] A. Sobolewski and K. Szyc, "A study of architecture optimization techniques for convolutional neural networks," in *International Conference on Dependability and Complex Systems*. Springer, 2023, pp. 273–283. [Online]. Available: doi:10.1007/978-3-031-37720-4_25

[2] S. Alyamkin, M. Ardi, A. C. Berg, A. Brighton, B. Chen, Y. Chen, H.-P. Cheng, Z. Fan, C. Feng, B. Fu *et al.*, "Low-power computer vision: Status, challenges, and opportunities," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 411–421, 2019. [Online]. Available: doi:10.1109/JETCAS.2019.2911899

[3] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," *Remote Sensing*, vol. 13, no. 22, p. 4712, 2021. [Online]. Available: doi:10.3390/rs13224712

[4] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019. [Online]. Available: doi:10.1109/ICCV.2019.00140

[5] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. [Online]. Available: doi:10.1109/CVPR.2018.00716

[6] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020. [Online]. Available: doi:10.1109/CVPR42600.2020.00165

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. [Online]. Available: doi:10.1109%2Fcvpr.2016.90

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. [Online]. Available: doi:10.48550/arXiv.1409.1556

[9] S. Mehta, H. Hajishirzi, and M. Rastegari, "Dicenet: Dimension-wise convolutions for efficient networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2416–2425, 2020. [Online]. Available: doi:10.1109/TPAMI.2020.3041871

[10] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, 2021. [Online]. Available: doi:10.1016/j.neucom.2021.07.045

[11] I. Rodriguez-Conde, C. Campos, and F. Fdez-Riverola, "Optimized convolutional neural network architectures for efficient on-device vision-based object detection," *Neural Computing and Applications*, vol. 34, no. 13, pp. 10 469–10 501, 2022. [Online]. Available: doi:10.1007/s00521-021-06830-w

[12] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*. IEEE, 2018. [Online]. Available: doi:10.1109/CCWC.2018.8301729

[13] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021. [Online]. Available: doi:10.1109/TNNLS.2021.3084827

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9. [Online]. Available: doi:10.1109/CVPR.2015.7298594

[15] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016. [Online]. Available: doi:10.48550/arXiv.1602.07360

[16] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013. [Online]. Available: doi:10.48550/arXiv.1312.4400

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. [Online]. Available: doi:10.1109/CVPR.2016.308

[18] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. [Online]. Available: doi:10.1109/CVPR.2017.195

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017. [Online]. Available: doi:10.48550/arXiv.1704.04861

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. [Online]. Available: doi:10.1109/CVPR.2018.00474

[21] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. [Online]. Available: doi:10.1109/TPAMI.2019.2913372

[22] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114. [Online]. Available: doi:10.48550/arXiv.1905.11946

[23] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006. SPIE, 2019, pp. 369–386. [Online]. Available: doi:10.48550/arXiv.1708.07120

[24] PyTorch. (2023) Pytorch: Reducelronplateau. PyTorch 2.0 documentation. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html