A CNN Compression Methodology for Layer-Wise Rank Selection Considering Inter-Layer Interactions

Milad Kokhazadeh[*], Georgios Keramidas[*+], Vasilios Kelefouras[†], Iakovos Stamoulis[+]
[*]School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
[+]Think Silicon S.A. An Applied Materials Company, Patras, Greece
[†]School of Engineering, Computing and Mathematics, University of Plymouth, Plymouth, United Kingdom
{kokhazad, gkeramidas}@csd.auth.gr, vasilios.kelefouras@plymouth.ac.uk, iakovos_stamoulis@amat.com

***Abstract*—Convolutional Neural Networks (CNNs) achieve state-of-the-art performance across various application domains but are often resource-intensive, limiting their use on resource-constrained devices. Low-rank factorization (LRF) has emerged as a promising technique to reduce the computational complexity and memory footprint of CNNs, enabling efficient deployment without significant performance loss. However, challenges still remain in optimizing the rank selection problem, balancing memory reduction and accuracy, and integrating LRF into the training process of CNNs. In this paper, a novel and generic methodology for layer-wise rank selection is presented, considering inter-layer interactions. Our approach is compatible with any decomposition method and does not require additional retraining. The proposed methodology is evaluated in thirteen widely-used, CNN models, significantly reducing model parameters and Floating-Point Operations (FLOPs). In particular, our approach achieves up to a 94.6% parameter reduction (82.3% on average) and up to 90.7% FLOPs reduction (59.6% on average), with less than a 1.5% drop in validation accuracy, demonstrating superior performance and scalability compared to existing techniques.**

***Index Terms*—CNN Compression, Low-Rank Factorization, Similarity Analysis, SVD, Tucker Decomposition**

I. Introduction

The advent of Convolutional Neural Networks (CNNs) has revolutionized the field of computer vision and significantly advanced numerous applications. CNNs have demonstrated superior performance in tasks such as image classification, object detection, and segmentation [1]. However, the remarkable performance of CNNs comes at the cost of large model sizes and substantial computational demands [2]. Fig. 1 depicts the number of parameters and Floating-Point Operations (FLOPs) of Fully Connected (FC) and Convolution (conv) parts of various well-known CNN models. As it is evident from Fig. 1, different CNN models exhibit different characteristics. Therefore, devising a CNN compression methodology that targets both the FC and conv layers in a unified framework is of utmost importance. For example, popular CNN compression techniques, like filter or channel pruning are limited to conv layers [3], [4].

To compress CNNs, various techniques have been proposed, which can be broadly categorized into four groups [5]: quantization [6], knowledge distillation [7], pruning [8], and Low-Rank Factorization (LRF) [9]. LRF, the focus of this paper, reduces the number of parameters and arithmetic complexity in CNNs without significantly compromising accuracy [9], by decomposing weight matrices/tensors into lower-rank counterparts. However, implementing LRF in CNNs poses several well-known challenges [10], [11].

The first challenge is that the search space for extracting a suitable LRF configuration for a given CNN layer is vast with each solution offering distinct trade-offs in terms of memory, FLOPs, and validation accuracy. Notably, determining the appropriate rank(s) presents an NP-hard problem [12]. Fig. 2 shows all possible LRF solutions when just two layers of ResNet [13] are compressed with LRF, filter pruning, and

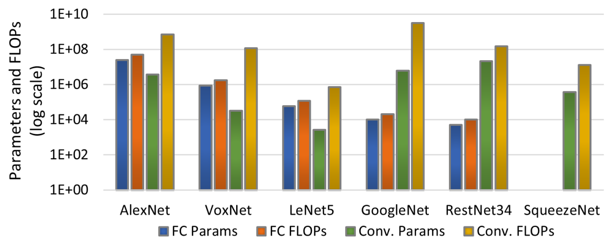

Fig. 1. Parameters and FLOPs of FC and conv parts for various CNN models

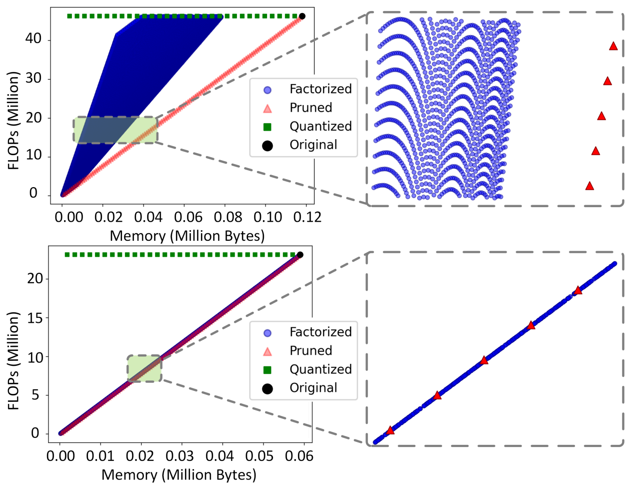

Fig. 2. Different solutions for LRF, pruning, and quantization when employed in two layers of ResNet. The green box shows the solutions with a FLOPs reduction equal to 60%±2%. The top part corresponds to a layer with different input/output feature map spatial sizes, while the bottom part represents a layer with identical input/output feature map sizes

quantization. The top part of Fig. 2 corresponds to a layer with different input and output spatial sizes, while the bottom part shows a layer with the same input and output spatial sizes (layers with the same input and output spatial sizes exhibit a similar pattern). As Fig. 2 indicates, for each memory and/or FLOPs level (each relates to a different rank), LRF offers a vast number of solutions maximizing in this way the possibilities to extract a solution that meets specific criteria posed by the user or the target application. Moreover, as the top part of Fig. 2 shows, for a specific level of FLOPs, LRF can offer solutions with a significantly lower number of parameters compared to the popular filter-based pruning (FBP). However, LRF includes a bigger exploration space than FBP and this is why a Design Space Exploration (DSE) methodology is required.

Second, the complexity increases exponentially when LRF is applied to an entire CNN model, which typically consists of dozens or even hundreds of layers, resulting in an enormous exploration space. Additionally, certain components, such as neurons or filters, that appear essential at the layer level may be deemed redundant when evaluated within the context of the entire model [14]. The third challenge is the lack of an end-to-end methodology that is able to accommodate different layer types, such as 1D, 2D, and 3D conv, as well as FC layers in a unified framework.

In this paper, we propose a DSE methodology for applying LRF in both conv and FC layers of CNNs. A key feature of our approach is the introduction of a novel similarity-based strategy for selecting the optimal rank for each CNN layer, which eliminates the need for lengthy re-training phases. Our methodology supports different layer types (e.g., 1D, 2D, and 3D conv layers, FC layers), as well as various matrix/tensor decomposition algorithms, such as Tucker decomposition [15] and Singular Value Decomposition (SVD) [16], within a unified framework. Furthermore, as demonstrated in our experimental results, another distinguishing feature of our approach is its ability to significantly reduce the design space. This leads to a smaller set of LRF solutions requiring fine-tuning (typically only 5-10 epochs).

The main contributions of this paper are: **i)** An end-to-end methodology that formulates the employment of LRF in CNNs as a DSE problem (FLOPs vs. memory size vs. validation accuracy or a combination of these metrics) and is compatible with different LRF algorithms and layer types, **ii)** A new similarity strategy (based on cosine similarity) that considers the nonlinear effects of each layer and effectively determines the appropriate rank for each layer, **iii)** A thorough experimental evaluation using 13 widely-used CNN models and seven different datasets, and **iv)** A modular methodology that can be easily integrated in typical neural network frameworks[1].

The remainder of this paper is organized as follows. Section II provides the necessary background information. Section III presents the proposed methodology. Section IV outlines our evaluation framework. Section V presents the experimental results, while Section VI reviews relevant literature. Finally, Section VII concludes the paper.

[1] Our implementation is based on TensorFlow 2.X [17].

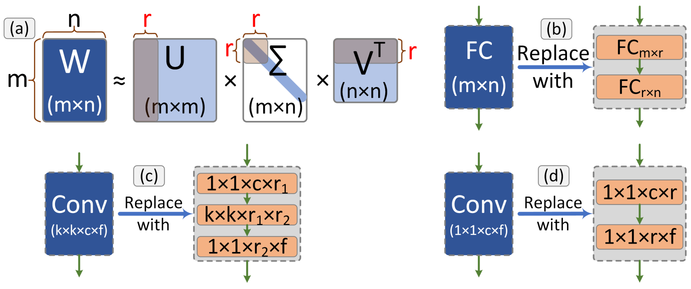

Fig. 3. SVD process (a) targeting to substitute a FC layer with two thinner FC layers (b). Substitution of a conv layer with three thinner layers based on Tucker Decomposition (c). Substitution of a point-wise conv layer with two thinner point-wise conv layers based on SVD (d).

II. Background

LRF entails decomposing a high-dimensional tensor or matrix into products of lower-dimensional tensors or matrices, with the rank determining the compression ratio. The main objective is to approximate the original matrix or tensor with fewer parameters and reduced computational demands. In the context of CNNs, LRF is typically applied to the weights of conv and/or FC layers.

Regarding matrices (2D-arrays), the most popular LRF approaches are SVD [16], QR decomposition [18], interpolative decomposition [19], and non-negative factorization [20]. For tensors, the most popular LRF approaches are Tucker Decomposition (TD) [15], Canonical Polyadic (CP) Decomposition [21], and tensor-train decomposition [22]. Another way to decompose a tensor is to transform it into a 2D matrix and then apply the matrix decomposition process [23]. In the context of this work, we rely on SVD to compress the FC layers of the input CNNs (Fig. 3.a and Fig. 3.b) and on TD for conv layers (Fig. 3.c). In addition, the SVD method is used for point-wise convolutions ($K_h = K_w = 1$), since the weight tensor is simplified to a 2D matrix in this case (Fig. 3.d). Exploring alternative LRF techniques is left for future work.

III. Proposed Methodology

This paper introduces an LRF DSE methodology and the corresponding design tool, aimed at achieving the highest compression rate while preserving accuracy and minimizing the processing time required for applying LRF in CNNs. The main steps of the proposed methodology are shown in the left part of Fig. 4. Each step is explained below.

We start by selecting a subset of layers as the target for factorization. Although the proposed methodology can be applied to all layers, focusing on a subset of layers allows for more efficient convergence to an optimal solution. In this study, we choose to factorize 90% of the layers (including both conv and FC layers) based on the number of parameters. This approach balances computational efficiency with model performance.

1. Initial solutions (rank-1): Since the goal of the proposed methodology is to find the most compressed solution, the solution with the highest compression for all layers is evaluated first, i.e., we use rank equal to one for all layers. This

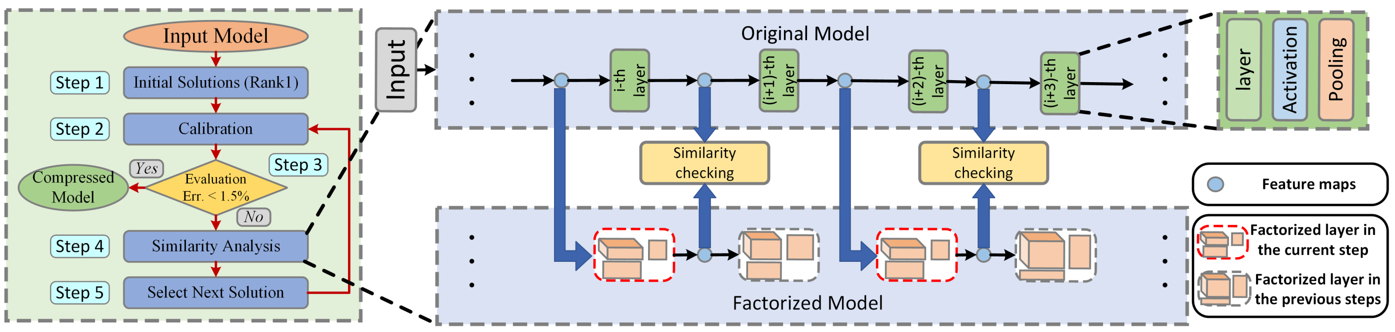

Fig. 4. Proposed methodology

initial model has the fewest parameters and FLOPs among all factorized models considered in the proposed methodology.

2. Calibration: To compensate for the accuracy loss caused by factorization, each LRF solution requires a calibration phase, which is the most time-consuming part of the LRF process. To address this problem, we avoid re-training or iteratively fine-tuning the model after each layer is factorized. Instead, we employ a one-shot, fine-tuning approach, where the factorized model is fine-tuned for a limited number of epochs, e.g., 10 epochs, only once, after all target layers have been factorized. As a result, the calibration time is significantly reduced.

3. Evaluation: After fine-tuning the factorized model, its performance is evaluated. This can be assessed in various ways, e.g., validation loss, accuracy, or a combination of metrics. In this study, we use validation accuracy to compare the factorized model against the original one. Additionally, we apply a threshold to determine the acceptable performance difference between the factorized and original models. Specifically, we set a user-defined threshold (in this paper is set to 1.5%), allowing for up to a 1.5% drop in accuracy. If the factorized model meets this accuracy constraint, the process stops and returns the factorized model. If the factorized model fails to meet the accuracy requirement, the process proceeds to the next steps to explore alternative LRF solutions.

4. Similarity analysis: After fine-tuning, if the factorized model does not meet the accuracy constraint, we must reduce the compression ratio by increasing the rank. Since each layer impacts accuracy differently, we face a question; how to adjust each layer's rank to maximize overall compression while maintaining accuracy close to the original model? Previous methods [24], [25] consider layers' weights individually without accounting for their interactions within the model. Another method [26] focuses on the similarity between the factorized and original weights. This approach is time-consuming, because it requires reconstructing the weight from the factorized components and also overlooks layer interactions and calibration effects. To address these issues, we propose a novel similarity strategy based on cosine similarity (CoSim).

The proposed similarity strategy focuses on feature maps rather than weight tensors or matrices, providing a better understanding of each layer's impact on model accuracy. Furthermore, we consider feature maps after subsequent operations such as activation, pooling, and normalization. To obtain the feature maps, we randomly select a subset of training data (1,000 samples in this study), feed them into the original model, and save the resulting feature maps. To calculate the similarity between the new factorized layers and the original ones, we follow this procedure (illustrated in the right part of Fig. 4): each factorized layer receives the corresponding feature map from the original model as input and its output is compared against the feature map of the original layer to determine similarity.

This similarity score is used to assess the impact of each factorized layer on the model's overall accuracy. The rationale for this is twofold: first, to ensure that similarity is not influenced by previously factorized layers, and second, to identify which layers have the greatest effect on the model's accuracy. For example, consider two factorized layers: the first has a high impact on accuracy, while the second has a low impact. In this scenario, our method will leverage this information, applying a higher compression rate to the second layer.

The CoSim between two vectors is calculated as:

$$cosine.similarity = \frac{\mathbf{A.B}}{\|\mathbf{A}\|\|\ \mathbf{B}\|} \tag{1}$$

where **A.B** is the dot product of the vectors, and $\|\mathbf{A}\|$ and $\|\ \mathbf{B}\|$ are the Euclidean norms of **A** and **B**, respectively.

5. Select next solution: After calculating the CoSim between the factorized layers and their original counterparts, we must determine whether to retain the factorized layer as it is or adjust its rank based on a threshold. Specifically, if the average similarity exceeds 0.92[2], we keep the current rank and LRF solution for this layer (this is the final rank for this layer). If the average similarity is below 0.92, we increase the rank based on a predefined compression step (*step-size*). In our experiments, we use a *step-size* of 5%, meaning that in each iteration, we decrease the compression ratio of the target layers by 5%.

The rationale for using a predefined step of 5% is as follows. Given that the design space is vast (even for a single layer) and grows exponentially when considering multiple layers, evaluating all LRF solutions becomes impractical, as each solution must undergo a calibration phase. Therefore, we introduce a *step-size* to select a subset of LRF solutions for each layer. The chosen *step-size* value introduces a trade-off between achieving

[2] The threshold is set to 0.96 for the layers in the non-sequential parts of the model.

a higher compression ratio and processing time; a smaller step size results in more LRF candidates for a layer, necessitating more time for evaluation.

It is important to note that in certain decomposition methods, such as TD, multiple LRF solutions may exist for a given compression ratio. In such cases, we select three different solutions for each ratio: the one with the maximum FLOPs, the one with the minimum FLOPs, and the one with the same rank across all dimensions. The first two solutions are chosen for the reasons outlined in [9], while the third is selected because it results in a lower approximation error [11][3].

After selecting the next solution, we return to step 4 to re-calibrate the factorized model. This process is repeated until we meet the accuracy constraints. This methodology ensures that layers with a greater impact on the overall accuracy (more sensitive layers) are compressed less, while layers with a lesser impact (less sensitive layers) are compressed more.

IV. Experimental Setup

We evaluate our approach using various CNN models with different basic blocks i.e., i) sequential-blocks: LeNet5, AlexNet, VGG11/16/19, Traffic Sign, VoxNet (3D conv), ii) residual-blocks: ResNet-18/34/50, iii) inception-blocks: GoogleNet, iv) fire-blocks: SqueezeNet, and v) dense-blocks: DenseNet, on different datasets i.e., MNIST (M), Fashion-MNIST (FM), ModelNet10 (MN10), ModelNet40 (MN40), GTSRB, CIFAR10 (C10), and CIFAR100 (C100). All models are trained from scratch for 100 epochs using random weights, Adam optimizer by setting the initial learning rate to 1e-4, and we utilized the ReduceLROnPlateau learning rate scheduler configured with a patience parameter of 10 epochs, a factor of 0.1, and a batch size equals to 32. The factorized models are fine-tuned for 10 epochs using the entire training dataset. In all cases we put a 1.5% accuracy drop constraint as an acceptable accuracy degradation. Also, in order to streamline the process, we used the same similarity thresholds for all models i.e., 0.92 for sequential blocks and 0.96 for non-sequential blocks.

As there are no existing LRF tools for direct comparison, we evaluate our methodology against the following approaches:

- Original model: In all cases, we report parameter reduction, FLOPs reduction, and accuracy loss of the factorized models wrt. the corresponding original model.
- Variational Bayesian Matrix Factorization (VBMF) [24]: VBMF is a probabilistic approach for matrix factorization, which leverages Variational Bayesian inference to estimate the distributions of the latent factors and noise in the data. The advantage of VBMF over other matrix factorization techniques is its ability to automatically determine the rank. Given that our approach can be viewed as a rank selection method, comparing it to VBMF is a fair and relevant evaluation. In contrast to our rank selection methodology, which is based on feature map similarity rather than weights, VBMF automatically estimates the rank of each layer based on its weights.
- Filter-Based Pruning (FBP) [8]: FBP is a well-known pruning technique used to reduce the parameters of CNNs by removing entire filters (or channels) from the conv layers. Filters are pruned based on certain criteria to determine the ones contributing the least to the network's performance. In this work, we guide the FBP process using three different metrics (L1-norm, L2-norm, and Geometric Median Distance (GMD) [8]). For a fair comparison, the metric that offers the highest compression ratio, while keeping the accuracy drop to less than 1.5%, is selected.

The experiments were conducted on a machine with an Intel Xeon Silver CPU (16 cores, 32 threads, 2.80 GHz, 20 MB L2, 24 MB L3) and an NVIDIA A40 GPU with 46GB GDDR6 memory. We used TensorFlow 2.15, Python 3.9.18, Tensorly 0.8.1 (for tensor decomposition), and NumPy 1.24.0 (for matrix decomposition). To ensure fairness and reproducibility, all reported processing times and overheads were normalized to a single epoch of training on the dataset.

V. Experimental Results

Comparison against the original Model: Fig. 5 depicts parameter reduction, FLOPs reduction, and relative accuracy achieved by our methodology compared to the original model. As mentioned, an accuracy drop constraint equal to 1.5% is assumed in all cases. The horizontal black line in the graph corresponds to the accuracy of the original models. The x-axis shows the studied models/datasets.

As Fig. 5 indicates, our approach achieves an average parameter reduction of 77.4% (up to 94.6% in AlexNet/C10) and an average FLOPs reduction of 55.5% (up to 90.7%). In addition, it is important to note that given the tight accuracy constraint of 1.5%, the proposed methodology manages to output a highly compressed solution in all studied models/datasets. Moreover, an increase in the accuracy of the factorized models can be observed in specific cases e.g., in VGG19/C100. However, there is one case in which our approach end-ups with a meager compression ratio. In particular, less than 10% reduction in both FLOPs and parameters is reported in SqueezeNet model on C100 dataset. Based on our investigation, the culprit is the static 5% threshold value being used by our methodology. Our analysis revealed (not shown in this work due space limitations) that alternative threshold values can reduce the number of SqueezeNet parameters by more than 50%. Devising an adaptive threshold selection policy is part of our future work.

LRF vs. VBMF and FBP: Our aim in this part is to illustrate the efficiency of the proposed approach against VBMF (automatic rank selection method) and FBP (widely-used CNN compression technique). Note that in all cases the target layers, the number of epochs for fine-tuning, and the accuracy drop limit are the same. Also, as mentioned three different metrics to prune the filters in FBP are used. Among the three options, the best-performing one is selected. Fig. 6 shows the relative parameter reduction achieved by our methodology[4] over VBMF and FBP, respectively. A capital "X" in Fig. 6 indicates that both

[3] Since Tucker decomposition can be applied along different tensor dimensions, the selected ranks may vary (in this work, we focus on two dimensions).

[4] Due to page limitations, we show only the relative reduction of parameters; the reduction in FLOPs exhibit a similar behavior.

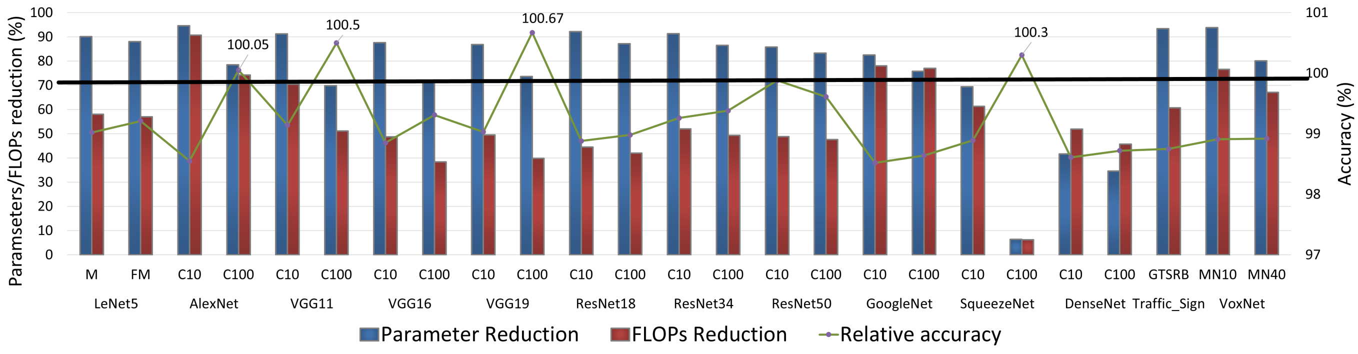

Fig. 5. Parameters and FLOPs reduction, and error w.r.t. to the original model

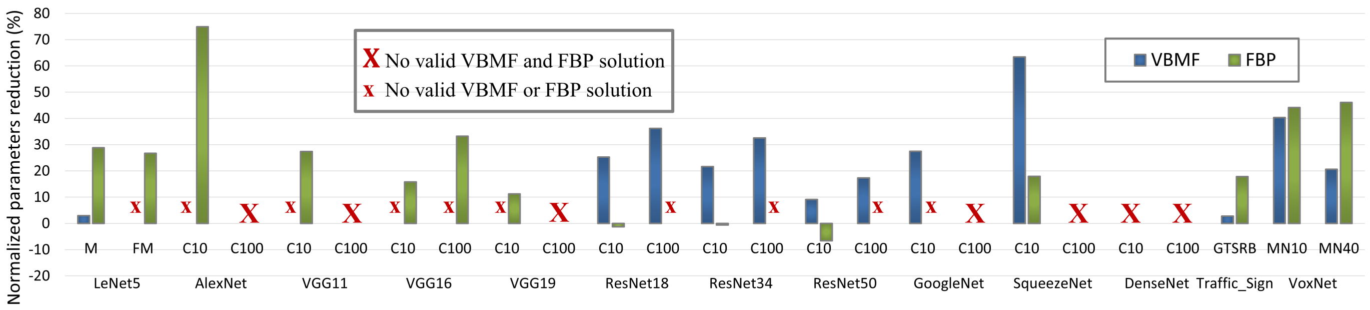

Fig. 6. Parameters reduction against VBMF and FBP with less than 1.5% accuracy drop

VBMF and FBP fail to meet the accuracy drop limit for the particular model/dataset. A lowercase "x" annotates the cases in which either VBMF or FBP fail to provide a solution.

As Fig. 6 shows, both VBMF and FBP are not able to extract a specific configuration that meet the accuracy drop constraint in many cases (in 13 cases for VBMF and in 11 cases for FBP out of 25). This an important observation proving the robustness of our methodology. For the remaining cases, our approach offers always a higher compression ratio compared to VBMF (e.g., 60% additional compression in SqueezeNet on C10 dataset). This is mainly because our approach takes into consideration the impact of compressing a particular layer when compressing subsequent layers. On the contrary, VBMF operates solely on the weights of individual layers and ignores any kind of inter-layer dependencies or interactions.

A similar behavior can be seen when our approach is compared against FBP. In principle, a key difference between LRF and the various pruning techniques is that LRF approximates the weights of an input model, while pruning removes redundant information (e.g., filters) from the network. According to Fig. 6, the proposed technique offers higher compression ratios from 12% up to 75% over FBP. However, there are only three cases (ResNet18/34/50 on C10 dataset) that FBP outperforms our scheme by up to 8%; note that in this particular cases, our scheme achieves more than 85% reduction in parameters wrt. the original model (Fig. 5). While a more sophisticated threshold selection policy can address this problem, it is important to understand that FBP comes with an inherent advantage. When a specific layer is compressed via FBP, then a number of channels from the subsequent layer are also eliminated, further increasing the compression ratio. On the contrary, LRF keeps the input and output channels of all layers intact.

FBP on top of LRF: A key advantage of the proposed methodology is its compatibility with other compression techniques (they are orthogonal), such as FBP. To demonstrate this, Fig. 7 illustrates how applying FBP to the factorized model enables additional compression. Specifically, after applying LRF, further compression can be achieved by applying FBP to the factorized layers.

Since FBP is applicable only to conv layers and affects the shape of subsequent layers, we showcase this scenario by applying FBP to the rightmost conv layer only, which connects to a relatively large FC layer. In this process, the model is first factorized using the proposed methodology (including both conv and FC layers) and as a second step FBP is applied to the rightmost conv layer. Consequently, both the conv and subsequent FC layers experience further compression. Fig. 7 displays the additional reduction in parameters and FLOPs normalized to the factorized model. As shown, combining these techniques yields an additional memory reduction ranging from 21% to 80%. Developing a more sophisticated methodology to efficiently combine these techniques across multiple layers is reserved for future work.

Overheads: Unlike other works that need to re-train the model for different configurations to find a suitable solution [27] or iteratively fine-tune the model after factorizing each layer [28], our approach uses an one-shot, fine-tuning strategy. This implies that the factorized model is fine-tuned only once at

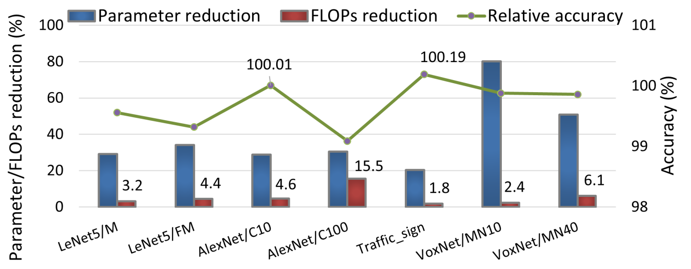

Fig. 7. Parameters and FLOPs reduction when FBP is employed in a factorized model

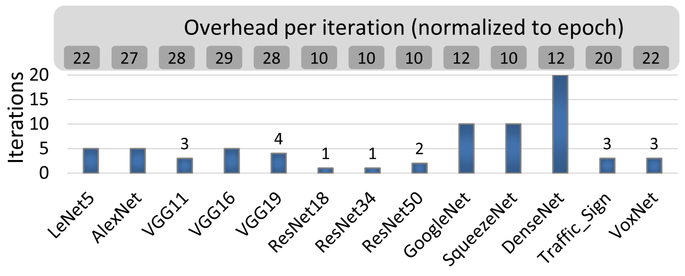

Fig. 8. Number of iterations and overhead time (norm. to the time needed for one epoch) of our methodology. For space reasons, only one dataset is shown.

each step i.e., after all layers in a particular step are factorized. Fig. 8 shows the number of iterations and epochs needed for fine-tuning plus the additional overhead time required by our approach. Overhead time includes the overall time that the proposed methodology needs to build a factorized model e.g., decomposing the weights or calculating the similarity between factorized and original models. The overhead time is normalized to the time needed for one epoch. According to Fig. 8, the proposed methodology is highly efficient and lightweight as it involves a streamlined process where a minimal calibration phase is required with minimal additional overheads.

VI. Related Work

The application of LRF in CNNs has been extensively studied [21], [29], [30]. As one of the pioneering works, [21] proposed a straightforward two-step approach for applying LRF in CNNs. In the first step, they select a 2D conv layer and replace it with a sequence of four convs. layers with smaller kernels, using CP decomposition and predefined ranks. In the second step, they fine-tune the entire model before proceeding to the next layer.

Several papers aim to select ranks manually [21], [29], [31], [32]. The challenge is to determine the optimal rank for each layer to achieve a suitable balance between compression and accuracy. Another issue is the processing time, as each solution must undergo a re-training or iterative fine-tuning phase. To address the issue of manual rank selection, various automated rank selection approaches have been developed, including techniques based on reinforcement learning [33] and genetic algorithms [30]. However, almost all of them encounter a similar issue: as compression ratio increases, the overhead time also rises. This is due to the non-linear increase in complexity involved in finding the optimal LRF combination [34].

Several papers utilize analytical methods like VBMF [24] and machine learning-based global optimization techniques, such as Bayesian Optimization [25], to automatically select the rank of each layer [35], [36]. The main limitation of these techniques is their focus on the weight tensor of individual layers, without taking into consideration: i) the non-linear responses of each layer and ii) the effects of interactions among the multiple layers within the model.

To address the aforementioned problems, while some studies attempt to train a low-rank model from scratch [10], [27], others address the optimal rank selection problem by defining LRF within the context of network architecture search (NAS) process [11]. Nevertheless, these methods require retraining the model, which is time-intensive. [9], [37], [38] address the challenge of large search space and rank selection in DNNs through a design space exploration methodology. However, their solutions are limited to FC layers and apply the same compression ratio in all FC layers, which may not fully optimize the compression and performance trade-offs for diverse model architectures. In addition, in [26], an iterative, greedy selection metric is employed to determine the rank of each layer, incorporating CoSim to better preserve the accuracy of low-rank Vision Transformers. However, this approach is limited to the self-attention components of a specific Transformer model and focuses solely on the weights of each layer.

Compared to the aforementioned techniques, our approach proposes an automatic rank selection methodology by utilizing the non-linear responses of each layer and selects different ranks for different CNN layers. As an one shot fine-tuning method, our methodology does not suffer from cumbersome re-training or iteratively fine-tuning processes. Furthermore, the proposed approach is applied to both conv and FC layers and is not limited to a specific decomposition algorithm.

VII. Conclusions

In this work we propose an LRF methodology for CNNs that effectively leverages the similarity of feature maps to select suitable, per-layer ranks, enabling a more efficient and streamlined compression process. By utilizing one-shot, fine-tuning, our approach significantly reduces the need for extensive re-training or iterative fine-tuning phases, thereby accelerating the optimization process while maintaining the performance (accuracy) of the input CNN models. Our experimental analysis, conducted on seven popular datasets with 13 different CNNs, showcases that not only the proposed methodology outperforms the existing methods (including other low-rank compression approaches and the widely-used pruning approaches), but also it is orthogonal to other CNN compression techniques.

Acknowledgment

This work is part of R-PODID project, supported by the Chips Joint Undertaking and its members, including the top-up funding by National Authorities of Italy, Turkey, Portugal, The Netherlands, Czech Republic, Latvia, Greece, and Romania under grant agreement No 101112338.

References

[1] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, and H. Ghayvat. CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. Electronics 10, 2021

[2] D. Kolosov, V. Kelefouras, P. Kourtessis and I. Mporas. Anatomy of Deep Learning Image Classification and Object Detection on Commercial Edge Devices: A Case Study on Face Mask Detection. Journal of IEEE Access, 2022

[3] F. Meng, H. Cheng, K. Li, H. Luo, X. Guo, G. Lu, and X. Sun. Pruning Filter in Filter. Advances in Neural Information Processing Systems 33, 2020

[4] C. Yang and H. Liu. Channel Pruning Based on Convolutional Neural Network Sensitivity. Neurocomputing 507, 2022

[5] G.C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca. Deep Neural Networks Compression: A Comparative Survey and Choice Recommendations. Journal of Neurocomputing, 2023

[6] H. Jin, D. Wu, S. Zhang, X. Zou, S. Jin, D. Tao, Q. Liao, and W. Xia. Design of a Quantization-based DNN Delta Compression Framework for Model Snapshots and Federated Learning. Transactions on Parallel and Distributed Systems, 2023

[7] F. Karimzadeh and A. Raychowdhury. Towards Energy Efficient DNN Accelerator via Sparsified Gradual Knowledge Distillation. International Conference on Very Large Scale Integration, 2022

[8] K. Gkrispanis, N. Gkalelis, and V. Mezaris. Filter-Pruning of Lightweight Face Detectors Using a Geometric Median Criterion. Winter Conference on Applications of Computer Vision, 2024

[9] M. Kokhazadeh, G. Keramidas, V. Kelefouras, and I. Stamoulis. A Practical Approach for Employing Tensor Train Decomposition in Edge Devices. International Journal of Parallel Programming, 2024

[10] M. Eo, S. Kang, and W. Rhee. An Effective Low-Rank Compression with a Joint Rank Selection Followed by a Compression-Friendly Training. Neural Networks, 2023

[11] J. Xiao, C. Zhang, Y. Gong, M. Yin, Y. Sui, L. Xiang, D. Tao, and B. Yuan. HALOC: Hardware-Aware Automatic Low-Rank Compression for Compact Neural Networks. AAAI Conference on Artificial Intelligence, 2023

[12] C.J. Hillar and L.H. Lim. Most Tensor Problems are NP-Hard. Journal of the ACM, 2013

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. Conference on Computer Vision and Pattern Recognition, 2016

[14] E. Ozen, and A. Orailoglu. Squeezing Correlated Neurons for Resource-Efficient Deep Neural Networks. European Conference on Machine Learning and Knowledge Discovery in Databases, 2021

[15] L.R. Tucker. Some Mathematical Notes on Three-Mode Factor Analysis. Psychometrika, 1966

[16] G. Golub and W. Kahan. Calculating the Singular Values and Pseudo-Inverse of a Matrix. Journal of the Society for Industrial and Applied Mathematics, 1965

[17] https://www.tensorflow.org/

[18] A. Chorti and D. Picard. Rate Analysis and Deep Neural Network Detectors for SEFDM FTN Systems. arXiv preprint arXiv:2103.02306, 2021

[19] J. Chee, M. Renz, A. Damle, and C.D. Sa. Pruning Neural Networks with Interpolative Decompositions. arXiv preprint arXiv:2108.00065, 2021

[20] T. K. Chan, C. S. Chin, and Y. Li. Non-Negative Matrix Factorization-Convolutional Neural Network (NMF-CNN) for Sound Event Detection. arXiv preprint arXiv:2001.07874, 2020

[21] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-Up Convolutional Neural Networks Using Fine-Tuned CP-Decomposition. International Conference on Learning Representations, 2015

[22] A. Novikov, D. Podoprikhin, A. Osokin, and D.P. Vetrov. Tensorizing Neural Networks. Conference on Advances in Neural Information Processing Systems, 2015

[23] Y. Idelbayev and M.A. Carreira-Perpinán. Low-Rank Compression of Neural Nets: Learning the Rank of each Layer. Conference on Computer Vision and Pattern Recognition, 2020

[24] S. Nakajima, M. Sugiyama, S. D. Babacan, and R. Tomioka. Global Analytic Solution of Fully-Observed Variational Bayesian Matrix Factorization. Journal of Machine Learning Research, 2013

[25] T. Kim, J. Lee, and Y. Choe. Bayesian Optimization-Based Global Optimal Rank Selection for Compression of Convolutional Neural Networks. IEEE Access, 2020

[26] Y.J. Luo, Y.S. Tai, M.G. Lin, and A.Y.A. Wu. Similarity-Aware Fast Low-Rank Decomposition Framework for Vision Transformers. International Symposium on Circuits and Systems, 2024

[27] Y. Sui, M. Yin, Y. Gong, J. Xiao, H. Phan, and B. Yuan. ELRT: Efficient Low-Rank Training for Compact Convolutional Neural Networks. arXiv preprint arXiv:2401.10341, 2024

[28] M. Astrid and S.I. Lee. Deep Compression of Convolutional Neural Networks with Low-Rank Approximation. ETRI Journal, 2018

[29] A.Mai, L. Tran, L. Tran, and N. Trinh. VGG Deep Neural Network Compression via SVD and CUR Decomposition Techniques. Conference on Information and Computer Science, 2020

[30] C. Dai, H. Cheng, and X. Liu. A Tucker Decomposition Based on Adaptive Genetic Algorithm for Efficient Deep Model Compression. International Conference on High Performance Computing and Communications; International Conference on Smart City; International Conference on Data Science and Systems, 2020

[31] M. Astrid and S.I. Lee. CP-Decomposition with Tensor Power Method for Convolutional Neural Networks Compression. International Conference on Big Data and Smart Computing, 2017

[32] A. H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavský, V. Glukhov, I. Oseledets, and A. Cichocki. Stable Low-Rank Tensor Decomposition for Compression of Convolutional Neural Network. European Conference Computer Vision, 2020

[33] Z. Cheng, B. Li, Y. Fan, and Y. Bao. A Novel Rank Selection Scheme in Tensor Ring Decomposition Based on Reinforcement Learning for Deep Neural Networks. International Conference on Acoustics, Speech, and Signal Processing, 2020

[34] C. Yang and H. Liu. Stable Low-Rank CP Decomposition for Compression of Convolutional Neural Networks Based on Sensitivity. Applied Sciences, 2024

[35] W. Ahmed, H. Hajimolahoseini, A. Wen, and Y. Liu. Speeding Up Resnet Architecture with Layers Targeted Low Rank Decomposition. arXiv preprint arXiv:2309.12412, 2023

[36] M. Astrid, S. I. Lee, and B. S. Seo. Rank Selection of CP-Decomposed Convolutional Layers with Variational Bayesian Matrix Factorization. International Conference on Advanced Communication Technology, 2018

[37] M. Kokhazadeh, G. Keramidas, V. Kelefouras, and I. Stamoulis. A Design Space Exploration Methodology for Enabling Tensor Train Decomposition in Edge Devices. International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, 2022

[38] M. Kokhazadeh, G. Keramidas, V. Kelefouras, and I. Stamoulis. Denseflex: A Low Rank Factorization Methodology for Adaptable Dense Layers in DNNs. International Conference on Computing Frontiers, 2024