

Efficient CNN Compression via Multi-method Low Rank Factorization and Feature Map Similarity

Milad Kokhazadeh, Georgios Keramidas, and Vasilios Kelefouras

Abstract—Low-Rank Factorization (LRF) is a widely adopted technique for compressing deep neural networks (DNNs). However, it faces several challenges, including optimal rank selection, a vast design space, long fine-tuning times, and limited compatibility with different layer types and decomposition methods. This paper presents an end-to-end Design Space Exploration (DSE) methodology and framework for compressing convolutional neural networks (CNNs) that addresses all these issues. We introduce a novel rank selection strategy based on feature map similarity, which captures non-linear interactions between layer outputs more effectively than traditional weight-based approaches. Unlike prior works, our method uses a one-shot fine-tuning process, significantly reducing the overall fine-tuning time. The proposed framework is fully compatible with all types of convolutional (Conv) and fully connected (FC) layers. To further improve compression, the framework integrates three different LRF techniques for Conv layers and three for FC layers, applying them selectively on a per-layer basis. We demonstrate that combining multiple LRF methods within a single model yields better compression results than using a single method uniformly across all layers. Finally, we provide a comprehensive evaluation and comparison of the six LRF techniques, offering practical insights into their effectiveness across different scenarios. The proposed work is integrated into TensorFlow 2.x, ensuring compatibility with widely used deep learning workflows. Experimental results on 14 CNN models across eight datasets demonstrate that the proposed methodology achieves substantial compression with minimal accuracy loss, outperforming several state-of-the-art techniques.

Impact Statement—Deep neural network (DNN) compression is a critical challenge in artificial intelligence (AI), especially for deploying models in resource-constrained environments. This article addresses the limitations of existing low-rank factorization (LRF) methods, such as suboptimal rank selection, long fine-tuning times, and limited applicability across different layers. A novel end-to-end compression framework is proposed, featuring a feature map similarity-based rank selection strategy, one-shot fine-tuning, and hybrid decomposition support. Unlike prior works, the framework applies different LRF methods per layer and supports six decomposition algorithms across convolutional and fully connected layers. Integrated into TensorFlow 2.x, it achieves superior compression and accuracy trade-offs across 14 CNN models and eight datasets, outperforming state-of-the-art techniques such as Variational Bayesian Matrix Factorization and filter-based pruning. This contribution enables scalable, architecture-agnostic compression and offers a practical tool for accelerating DNN deployment in mobile AI, embedded systems, and edge computing scenarios. It also provides insights that may guide future research on compression-aware design and model optimization.

Index Terms—Convolutional Neural Networks, Low-Rank Factorization, Model Compression, Tensor Decomposition

I. INTRODUCTION

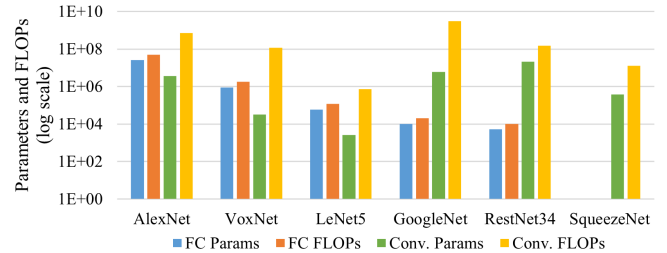


Fig. 1. Breakdown of FLOPs and parameter distribution across Conv and FC layers in various CNN architectures.

DESPITE the recent success of Transformer-based models [1], convolutional neural networks (CNNs) remain widely used in computer vision, including tasks such as image classification, object detection, and semantic segmentation [2]. CNNs are computationally intensive and require substantial memory resources, which limits their deployment on resource-constrained platforms such as mobile devices, embedded systems, and edge devices [3], [4].

Fig. 1 illustrates the number of parameters and Floating-Point Operations (FLOPs) associated with the Fully Connected (FC) and Convolutional (Conv) layers of various widely-used CNN models. As it is evident from Fig. 1, CNN models exhibit diverse characteristics, emphasizing the need for a holistic compression methodology that effectively targets both the FC and Conv layers within a unified framework.

Addressing this challenge has driven the development of numerous model compression techniques aimed at reducing model size and computational complexity while preserving accuracy [5]. Among the various model compression techniques, Low-Rank Factorization (LRF) has emerged as a particularly effective approach [6]. LRF reduces both the number of parameters and the computational cost (FLOPs) of CNNs, making them more suitable for resource-constrained deployments. Compared to other methods such as pruning [7], quantization [8], and knowledge distillation [9], LRF provides a flexible range of decomposition strategies, allowing for a fine-grained balance between memory usage, computational efficiency, and model accuracy [10].

Setting aside evaluation details for now, the top graphs in Fig. 2 compare model parameters and FLOPs for several DNN compression techniques applied to a Conv layer in ResNet50. Filter-based pruning (FBP) (cyan triangles) and quantization (pink squares), while somewhat effective, show limited ability to reduce both memory usage and computational cost simultaneously. For example, quantization reduces model size but does not decrease the number of parameters or FLOPs. FBP can reduce both parameters and FLOPs but falls short in

achieving high compression ratios. In contrast, for the same FLOPs level, various LRF methods can achieve significantly greater compression.

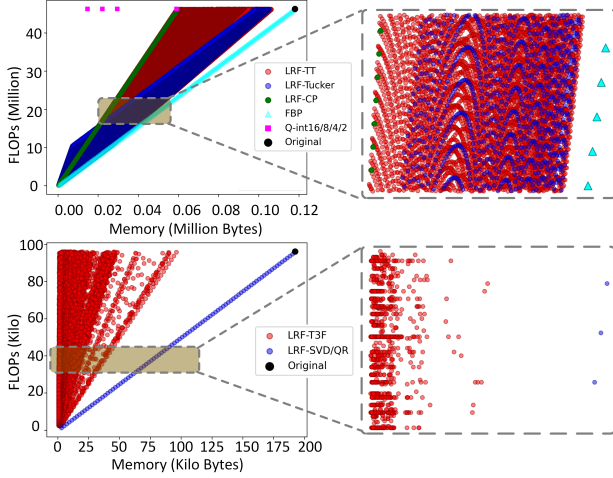


Fig. 2. Exploration space of multiple compression methods in the FLOPs-memory space. The graph at the top illustrates a Conv layer of ResNet50, while the graphs at the bottom shows a FC layer of LeNet5.

However, the practical use of LRF in CNNs comes with significant challenges. First, even a single layer presents a vast design space due to the many possible LRF configurations (Fig. 2). For example, applying Tucker decomposition to one layer of a ResNet model with shape (3, 3, 128, 256) results in 23,766 possible configurations. When extended to an entire model with multiple layers, the complexity becomes unmanageable for manual tuning. Second, selecting the optimal rank for each layer typically requires costly iterative calibration or retraining. For instance, in a relatively small model like LeNet-5, which has only 5 layers, the total number of possible LRF configurations can reach around 38 million. Even under a modest evaluation setup—calibrating each configuration for just 5 epochs at 1 second per epoch—the total search time would exceed 2,193 days, underscoring the impracticality of exhaustive search for optimal settings. Last but not least, existing LRF-based approaches often focus on specific layer types or decomposition methods, limiting their applicability to modern CNN architectures, which feature diverse layer types such as 1D, 2D, or 3D Conv layers, as well as FC layers.

This paper introduces an end-to-end Design Space Exploration (DSE) methodology and framework for CNN compression, addressing all the above challenges¹. The proposed methodology offers several key advantages/contributions over existing approaches:

- **An end-to-end DSE framework** that formulates CNN compression using LRF as a **multi-objective** optimization problem balancing FLOPs, model parameters, overall memory usage, and validation accuracy.
- **Layer-wise, similarity-based rank selection:** We introduce an automated rank selection strategy that dynamically determines the optimal rank for each layer based on feature map similarity, rather than traditional weight

similarity. This approach better captures the nonlinear relationships between features and enables more effective, tailored compression across layers.

- **Efficient fine-tuning process:** Our approach adopts a one-shot fine-tuning strategy that eliminates the need for iterative calibration or extensive retraining.
- **Compatibility with different layer types and decomposition methods:** The proposed framework supports a variety of layer types (e.g., 1D, 2D, and 3D Conv layers and FC layers) and multiple decomposition algorithms, i.e., Tucker decomposition [12], Canonical Polyadic (CP) decomposition [13], Tensor Train (TT) decomposition [14], Singular Value Decomposition (SVD) [15], QR [16], and T3F [17]).
- **Hybrid decomposition:** The proposed framework incorporates a post-processing step that combines three LRF methods for Conv layers (Tucker decomposition, CP decomposition, and TT decomposition) with three LRF methods for FC layers (SVD, QR decomposition, and T3F). This hybrid approach achieves superior compression, enhancing overall model efficiency.
- **Compatibility with other compression techniques:** Our approach is compatible with other compression techniques, like FBP and quantization, allowing seamless integration to achieve further reductions in model size.
- **A modular, easy-to-integrate framework built on TensorFlow 2.x [18],** designed to plug seamlessly into existing deep learning workflows. The framework will be made publicly available upon acceptance to support reproducibility and further research.
- **An analytical study of six LRF methods,** offering insightful observations on their strengths and trade-offs.

We evaluate our approach on 14 popular CNNs across eight diverse datasets. Results show significant compression, averaging 77.8%, 71.2%, and 76% for Tucker, CP, and TT (Conv layers), and 79.1%, 79.7%, and 81% for SVD, QR, and T3F (FC layers), with under 1.5% accuracy loss, outperforming several state-of-the-art techniques like VBMMF [19]. Our hybrid approach achieves 82.5% and 92.7% average parameter reduction in Conv and FC layers, respectively.

The remainder of this paper is organized as follows. Section II provides the necessary background on LRF and CNN compression techniques, while Section III reviews related literature. Section IV evaluates the six studied methods, while Section V provides the proposed methodology and hybrid decomposition approach. Section VI describes the evaluation framework, and Section VII presents the experimental results. Finally, Section VIII concludes the paper.

II. BACKGROUND

LRF is a fundamental technique for compressing DNNs by exploiting the redundancy in their weight matrices and tensors. By approximating high-dimensional tensors or matrices with smaller, factorized components, these methods reduce both the number of parameters and FLOPs. Broadly, decomposition methods can be categorized into matrix and tensor decomposition methods.

¹ This work is an extension of previous conference paper presented at the DATE 2025 conference [11]

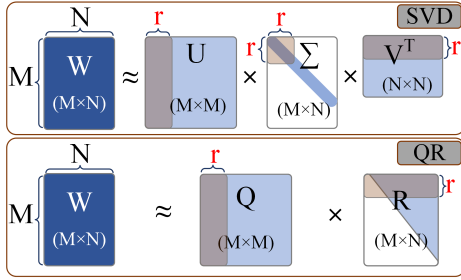


Fig. 3. Matrix decomposition using SVD (top) and QR decomposition (bottom)

A. Matrix Decomposition Methods

SVD: The SVD method [15] is depicted in the top of Fig. 3. It factorizes a given matrix W into three matrices whose size is smaller than the original matrix for a given rank r . When these matrices are multiplied together, they reconstruct the original matrix W .

QR Decomposition: The QR decomposition method [16], shown in the bottom of Fig. 3, factors a matrix W into two components: an orthogonal matrix Q and an upper triangular matrix R .

B. Tensor Decomposition Methods

Tucker Decomposition: Tucker decomposition [12] is depicted at the top of Fig. 4. It decomposes a high-dimensional tensor X into a smaller core tensor G , along with multiple factor matrices (A , B , and C as shown at the top of Fig. 4).

CP Decomposition: CP decomposition [13], illustrated at the middle of Fig. 4, represents a high-order tensor as a sum of *rank-1* tensors, effectively factorizing it into a set of vectors along each mode. Indeed, as shown in the middle of Fig. 4 this results in a sequence of 2D matrices.

TT Decomposition and T3F: TT decomposition [14], shown at the bottom of Fig. 4, represents a high-dimensional tensor as a sequence of smaller, lower-dimensional tensors, commonly referred to as “cores,” which are interconnected by contracted indices². T3F [17] is a library for employing TT decomposition to FC layers. T3F factorizes a FC layer by reshaping its 2D weight matrix into a higher-dimensional tensor. It then uses TT decomposition and Kronecker product to factorize a FC layer [20]³.

C. Decomposing Fully Connected (FC) layers

Applying matrix decomposition methods to FC layers is straightforward since the weight matrix is inherently a 2D array. However, tensor decomposition methods, such as TT decomposition, can also be applied by reshaping the weight matrix of a FC layer into a tensor [17], but this introduces an overhead.

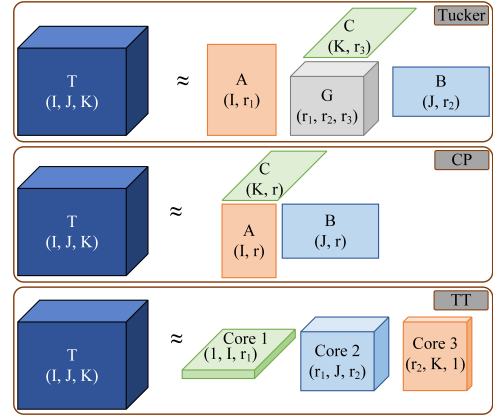


Fig. 4. Tensor decomposition of Conv2D layers using Tucker (top), CP (middle), and TT (bottom) methods.

D. Decomposing Convolution (Conv) layers

Conv layers naturally operate on tensors, making tensor decomposition methods more suitable and straightforward to apply. Conversely, matrix decomposition methods are less commonly used in Conv layers due to their 2D structure and lack of alignment with the multi-dimensional nature of Conv operations [21]. However, for 1×1 Conv layers, the weight tensor reduces to a 2D array, allowing matrix decomposition methods to be applied directly.

III. RELATED WORK

The application of LRF in CNNs has been extensively studied in the literature [22], [23], [24], [25], with research efforts primarily addressing three key challenges: rank selection and large exploration space (ES), the lengthy fine-tuning/re-training process, and compatibility issues. Specifically, selecting the appropriate ranks is critical for balancing compression and accuracy but since this is an NP-complete problem several studies resort to manual rank selection [22], [23], [24], [25]. The large ES introduces complexity, making it difficult to efficiently explore all possible solutions. The fine-tuning or re-training process is often time-consuming, limiting the practicality of LRF in real-world applications. Finally, compatibility remains a challenge, as no single LRF method can be universally applied to all CNN layers.

To address the issue of manual rank selection, various automated rank selection approaches have been developed, including reinforcement learning [26] and genetic algorithms [27]. However, nearly all of them encounter a similar issue: as the compression ratio increases, the time required also rises. This is due to the non-linear increase in complexity involved in finding the optimal LRF combination [28].

Several papers utilize analytical methods like Variational Bayesian Matrix Factorization (VBMF) [19] and machine learning-based global optimization techniques, such as Bayesian Optimization [29], to automatically select the rank of each layer [30], [31]. These techniques focus on the weight tensor of individual layers, without taking into consideration: i) the influence of subsequent layers’ outputs, e.g., activation, pooling, or batch normalization layers and ii) the effects of interactions among the multiple layers within the model.

² For better visualization a 3D tensor is used in these cases

³ Since the resulting factors are 4D tensors in the T3F, it is difficult to visualize it in a 2D figure, and thus we do not present them graphically.

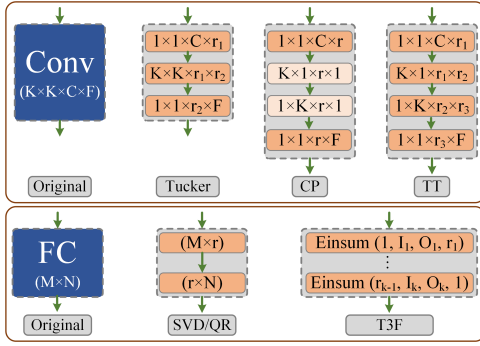


Fig. 5. LRF process for Conv2D layers using Tucker, CP, and TT decomposition (top), and for FC layers using SVD, QR, and T3F (bottom)

To address the aforementioned problems, while some studies attempt to train a low-rank network from scratch [32], [33], others address the optimal rank selection problem by defining LRF within the context of network architecture search (NAS) process [34]. Nevertheless, all these methods require retraining the model, which is time-intensive. The works [35], [10], [36] address the challenge of large search space and rank selection in DNNs through a DSE methodology. However, their solutions are limited to FC layers and apply the same compression ratio across different FC layers, which may not fully optimize the compression and performance trade-offs for diverse network architectures.

A closely related study to the proposed methodology is presented in [37], where feature maps and their norms are utilized to estimate the rank of each layer. However, this approach treats each layer independently, without considering inter-layer interactions. Additionally, unlike our proposed one-shot fine-tuning strategy, their method employs a slower iterative process to refine the rank selection for each layer.

The first systematic comparison of decomposition methods is [38], which selects layer-wise ranks based on tensor approximation error. However, it lacks automatic rank selection, uses fixed compression ratios per method, treats methods in isolation, and does not offer a unified compression framework.

To enhance accuracy retention in low-rank Vision Transformers, an iterative, greedy selection metric is employed, incorporating cosine similarity to determine the rank of each layer [39]. However, this approach is limited to the self-attention components of a specific Transformer model and focuses solely on the weights of each layer.

In the previous work [11], a one-shot and self-adaptive rank selection technique is proposed. This work extends and

enhances it by integrating six LRF methods into the framework, analyzing the impact of key parameters, providing an analytical comparison of the LRF methods, and introducing a hybrid compression strategy that combines six decompositions within a single model.

IV. ANALYTICAL STUDY OF LRF TECHNIQUES AS A BASIS FOR THE PROPOSED FRAMEWORK

In this Subsection, we conduct a thorough analysis of the six LRF methods studied. This analysis is essential for developing the proposed framework, as it involves three key steps: first, deriving the mathematical equations for parameter count, overall memory, FLOPs, and the exploration space (ES) of each method; second, identifying the unique characteristics of each technique, such as which dimensions to factorize and the corresponding rank configurations; and third, establishing the foundation for the hybrid decomposition step. We also provide insightful observations on the characteristics and strengths of the six, studied decomposition methods and deduct specific conclusions regarding their suitability for various scenarios.

Rank Configuration. To fully understand the decomposition process, we first analyze which dimensions can be factorized and how many ranks can be used in each method. Fig. 5 illustrates how an uncompressed layer is decomposed into multiple 'smaller' layers for Conv (top) and FC (bottom) layers, respectively.

For Conv layers, CP and TT factorize all tensor dimensions, while Tucker decomposition allows selective factorization across different dimensions, enabling decomposition of one, two, or more dimensions as needed. In this paper, we apply Tucker only to the input and output channel dimensions, leaving the spatial (kernel) dimensions intact, as kernel sizes are typically small and offer limited compressibility. For example, given a layer of shape (3, 3, 128, 256), we decompose the dimensions of size 128 and 256, corresponding to the input channels and filters, respectively.

CP employs a single rank value, whereas TT uses five rank values where the first and last ranks are always equal to 1. Regarding Tucker, we choose to use two rank values in this paper. As we will discuss later, using multiple ranks enhances flexibility by enabling independent control over the factorization along different tensor dimensions. This is important because the redundancy in weight tensors is often unevenly distributed, e.g., some dimensions (e.g., output channels) may be more compressible than others (e.g., spatial

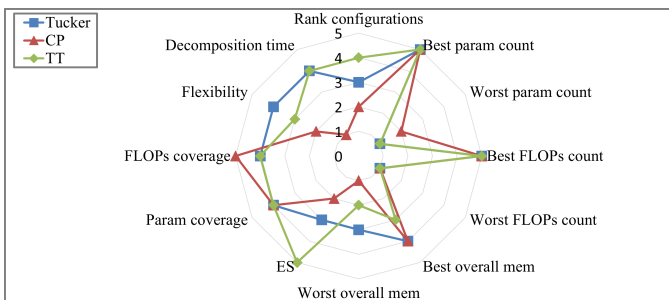


Fig. 6. Evaluation of the three LRF methods used for Conv layers (higher score means better).

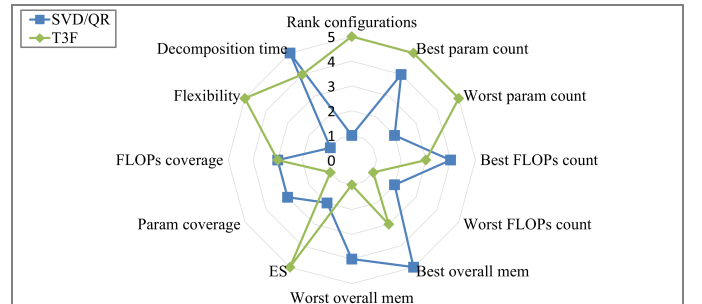


Fig. 7. Evaluation of the three LRF methods used for FC layers (higher score means better).

TABLE I
EVALUATION IN TERMS OF PARAMETERS (P), FEATURE MAPS (FM), FLOPS (F), AND COMPLEXITY (O).

Convolution layer			FC layer		
Method		Equation	Method		Equation
Original	P	$K_1 \times K_2 \times C \times F$	Original	P	$M \times N$
	FM	$X' \times Y' \times F$		FM	M
	F	$2 \times X' \times Y' \times F \times K_1 \times K_2 \times C$		F	$2 \times M \times N$
	O	$O(X'Y'FCK_1K_2)$		O	$O(MN)$
Tucker	P	$(C \times r_1) + (K_1 \times K_2 \times r_1 \times r_2) + (r_2 \times F)$	SVD	P	$(M \times r) + (r \times N)$
	FM	$(X \times Y \times r_1) + (X' \times Y' \times r_2) + (X' \times Y' \times F)$		FM	$r + M$
	F	$2 \times [(X \times Y \times r_1 \times C) + (X' \times Y' \times r_2 \times K_1 \times K_2 \times r_1) + (X' \times Y' \times F \times r_2)]$		F	$2 \times [(M \times r) + (r \times N)]$
	O	$O(XYC r_1 + X'Y'r_1r_2K_1K_2 + X'Y'Fr_2)$		O	$O(r(M+N))$
CP	P	$(C \times r) + (K_1 \times r) + (K_2 \times r) + (r \times F)$	QR	P	$(M \times r) + (r \times N)$
	FM	$(X \times Y \times r) + (X' \times Y \times r) + (X' \times Y' \times r) + (X' \times Y' \times F)$		FM	$r + M$
	F	$2 \times [(X \times Y \times r \times C) + (X' \times Y \times K_1 \times r) + (X' \times Y' \times K_2 \times r) + (X' \times Y' \times F \times r)]$		F	$2 \times [(M \times r) + (r \times N)]$
	O	$O(XYC r + X'Y'rK_1 + X'Y'rK_2 + X'Y'Fr)$		O	$O(r(M+N))$
TT	P	$(C \times r_1) + (K_1 \times r_1 \times r_2) + (K_2 \times r_2 \times r_3) + (r_3 \times F)$	T3F	P	$\sum_{t=1}^d (r_{t-1} \cdot m_t \cdot n_t \cdot r_t)$
	FM	$(X \times Y \times r_1) + (X' \times Y \times r_2) + (X' \times Y' \times r_3) + (X' \times Y' \times F)$		FM	$\sum_{t=d}^1 m_t \cdot r_{t-1} \cdot (X_{t+1} / (n_d \cdot r_d))$
	F	$2 \times [(X \times Y \times r_1 \times C) + (X' \times Y \times r_2 \times K_1 \times r_1) + (X' \times Y' \times r_3 \times K_2 \times r_2) + (X' \times Y' \times F \times r_3)]$		F	$\sum_{t=1}^d 2 \cdot r_t \cdot r_{t-1} \cdot m_t \dots m_d \cdot n_1 \dots n_t$
	O	$O(XYC r_1 + X'Y'r_1r_2K_1 + X'Y'r_2r_3K_2 + X'Y'Fr_3)$		O	$O(d \max\{r_i\}^2 \max\{n_i\} \max\{M, N\})$

dimensions). By assigning separate rank values per dimension, the decomposition can better exploit this anisotropy, leading to more effective compression without uniformly degrading performance.

In the case of FC layers, SVD and QR factorize both dimensions and employ a single rank value (Fig. 5). In contrast, TT, as implemented in the T3F library [17] for FC layers, requires a list of ranks along with a predefined tensor shape to which the original 2D matrix is transformed. Notably, similar to standard TT decomposition, the first and last ranks in T3F are always constrained to one.

Parameter Count. Table I summarizes the mathematical expressions for calculating the number of parameters (P), FLOPs (F), feature map (FM) elements, and computational complexity (Big O notation), for each decomposition method.

To compare the six LRF methods in terms of memory and FLOPs, we adopt two complementary approaches:

- 1) Max/Min Analysis: We compute the maximum and minimum achievable parameter count, overall memory and FLOPs for each method (Fig. 6 and Fig. 7).
- 2) Constraint-based Analysis: We evaluate them under specific memory and FLOPs constraints. Specifically, which method achieves the lowest FLOPs for a given level of memory compression, and which method achieves the best parameter count for a fixed reduction in FLOPs.

Fig. 6 and Fig. 7 present the average values across all layers from 14 CNN architectures studied in this work (detailed in the Appendix). Note that the memory and FLOPs equations for SVD and QR are identical (Table I); therefore, a single line (blue) represents both methods in Fig. 7. It is important to note that this analysis does not take into account the model's accuracy. To the best of our knowledge, this is the

first work that provides a quantitative comparison of various LRF methods.

Regarding the first way of evaluation and Conv layers, all methods exhibit similar best achievable parameter count (with a maximum difference of only 1%). However, their worst-case compression varies significantly (Fig. 6 and Fig. 7). For FC layers, T3F achieves the highest parameter reduction, benefiting from its ability to exploit multi-dimensional redundancies more effectively than the matrix decomposition methods.

In the second evaluation method (not shown in Fig. 6 and Fig. 7), LRF methods yield significantly different results under fixed memory or FLOPs constraints, depending on the layer's shape. For example, for a layer of shape (3, 3, 256, 512) and 60% parameter reduction, Tucker offers two solutions with 47% and 53% FLOPs reduction, CP gives one with 20%, and TT provides 141 solutions ranging from 1% to 57%. Similarly, fixing FLOPs can lead to large variations in parameter count across methods.

Overall Memory. The overall memory footprint consists of the memory required for both the parameters and the FMs.

In Conv layers, FMs have a minor impact on overall memory when spatial dimensions are small but can dominate memory usage when spatial dimensions are large, often exceeding the memory required for the parameters. Moreover, at low compression levels, TT and CP can produce FMs larger than those of the original layer. In FC layers, SVD and QR decompositions generate minimal FM memory, making their contribution to the overall footprint negligible. In contrast, T3F produces multiple high-memory FMs, significantly increasing overall memory.

Regarding the first way of evaluation (max/min analysis), no single decomposition method consistently delivers the

TABLE II
EVALUATION IN TERMS OF ES FOR SIX CONV LAYERS.

Layer ($K_1, \dots, K_d, C, F$)	All/Selected LRF Solutions			Comp. ratio	Solutions per step		
	Tucker	CP	TT		Tucker	CP	TT
1D-(3,512,1024)	5.2E+5	1.5E+3	5.2E+5	85%	3	1	3
	4.1E+5	1.0E+3	4.1E+5	60%	2	1	2
				25%	2	1	2
2D-(3,3,256,512)	1.3E+5	2.3E+3	1.0E+8	85%	2	1	34
	1.2E+5	7.6E+2	5.5E+7	60%	2	1	142
				25%	1	0	0
2D-(3,3,512,512)	2.6E+5	4.6E+3	4.0E+8	85%	3	1	483
	2.5E+5	2.2E+3	2.6E+8	60%	3	1	485
				25%	3	1	720
2D-(5,5,96,256)	2.4E+4	2.4E+3	1.1E+7	85%	2	1	19
	2.4E+4	1.6E+3	8.8E+6	60%	3	1	11
				25%	1	1	1
2D-(3,3,384,256)	9.8E+4	2.3E+3	7.5E+7	85%	2	1	57
	9.5E+4	1.3E+3	5.3E+7	60%	2	1	33
				25%	2	1	22
3D-(3,3,3,32,32)	1.0E+3	8.6E+2	9.4E+6	85%	3	1	423
	1.0E+3	1.0E+2	3.1E+6	60%	3	1	115
				25%	3	1	0

lowest overall memory footprint across all Conv layers, as the outcome depends on the specific layer shape. However, across the 14 CNNs studied in this work, Tucker achieves the lowest average overall memory footprint. This is primarily because Tucker results in fewer Conv layers, and therefore fewer FMs, compared to the other methods.

For example, in 2D Conv layers, Tucker produces three Conv layers, while the other methods yield four. This reduction in layer count leads to smaller FM memory consumption for Tucker. The advantage is even greater in 3D Conv layers: Tucker still uses only three layers, whereas alternative methods can produce up to five, further increasing Tucker’s memory efficiency.

When FMs occupy more memory than parameters (typically in cases with large spatial dimensions), Tucker shows a clear advantage in minimizing total memory usage. In FC layers, SVD and QR consistently yield the lowest overall memory footprint. Although T3F achieves lower parameter counts, its high FM memory makes it less efficient overall.

Regarding the second way of evaluation (constraint-based analysis), LRF methods yield significantly different overall memory values under fixed parameter or FLOPs constraints, depending on the layer’s shape.

FLOPs. Regarding the first way of evaluation (max/min analysis), all methods apart from T3F exhibit similar best and worst achievable FLOPs, with a maximum difference of only 1% (Fig. 6 and Fig. 7).

Regarding the constraint-based analysis, for a specific parameter count, CP yields the lowest FLOPs due to its use of depthwise convolution, followed by TT. In contrast, Tucker results in the highest FLOPs; however, the difference between Tucker and TT is minor.

T3F results in a higher FLOP count than SVD/QR due to its use of tensor contraction instead of conventional matrix-matrix multiplication. Additionally, T3F introduces additional reshape layers, which increase memory accesses and further degrade the inference time [36].

Exploration space (ES). The ES represents the total number of solutions that can be generated by a given decomposition method. For most methods (excluding T3F), the ES is determined by the number of unique rank configurations the method supports. In contrast, T3F offers not only a variety of rank so-

TABLE III
EVALUATION IN TERMS OF ES FOR THREE FC LAYERS.

Layer shape	LRF solutions in overall			Comp. ratio	Solutions per step		
	SVD	QR	T3F		SVD	QR	T3F
(400,120)	1E+2	1E+2	8E+5	85%	1	1	4
	9E+1	9E+1	3E+4	60%	1	1	1
				25%	1	1	0
(512,512)	5E+2	5E+2	3E+6	85%	1	1	2
	2E+2	2E+2	9E+4	60%	1	1	3
				25%	1	1	0
(512,256)	2E+2	2E+2	1E+6	85%	1	1	6
	1E+2	1E+2	3E+4	60%	1	1	2
				25%	1	1	0

lutions but also a diverse range of tensor shape configurations (i.e., combination shapes), dramatically expanding its ES.

Table II and Table III present a comprehensive analysis of all valid LRF configurations for six Conv and three FC layers, respectively. The selected layers reflect common configurations within the 14 selected CNNs (more than 80% of the layers exhibit similar structural properties and decomposition responses). Table II and Table III show the number of LRF configurations that result in lower parameters and FLOPs compared to the original layer. Solutions are grouped into three compression levels, i.e., low (25%), medium (60%), and high (85%), to illustrate the trade-off space and diversity of viable options under practical constraints.

The higher the number of ranks, the greater the number of possible rank combinations, and thus, the larger the ES. For Conv layers, TT yields the largest ES due to its use of the most rank values, followed by Tucker, which allows for multiple ranks (Table II). For FC layers, SVD and QR use a single rank, resulting in a small ES (Table III). On the other hand, T3F provides an exceptionally large ES, as it supports multiple ranks and a wide variety of tensor shapes and configurations [20] (Table III). However, this versatility makes T3F a powerful tool for optimizing FC layers, enabling more diverse and efficient configurations.

The higher the ES, the longer the time needed to extract and process all possible solutions (‘solution generation time’ in Table IV and Table V). The results indicate that, in some cases, specially in TT decomposition, the ‘solution generation time’ can take several hours. The ‘solution generation time’ is defined as the time needed to obtain all solutions and store them into memory. All runtime measurements are obtained on the system described in Section VI.

Parameter and FLOPs Coverage Across the ES. It is important to note that the distribution of solutions in the ES is not uniform across the Memory-FLOPs space (Fig. 6 and Fig. 7). For example, T3F solutions only appear in high memory reduction scenarios, with none present when memory reduction is low (Fig. 2). In Fig. 2, SVD provides a higher memory coverage than T3F. In Conv layers, all methods provide the same parameter coverage while in FC layers all methods provide the same FLOPs coverage.

Flexibility. Flexibility, in this context, refers to the ability to adjust factorization independently across tensor dimensions, allowing for better adaptation to data structure and redundancy. For Conv layers, Tucker is the most flexible, supporting selective decomposition and separate rank values per dimension. TT also allows multiple ranks, making it more flexible than CP,

TABLE IV

EVALUATION IN TERMS OF EXTRACTION AND DECOMPOSITION TIME FOR SIX CONV LAYERS.

Layer ($K_1, \dots, K_d, C, F$)	Solution generation time (s)			Comp. ratio	Decomposition time (s)		
	Tucker	CP	TT		Tucker	CP	TT
1D-(3,512,1024)	39	0.1	48	85%	51	3926	82
				60%	229	20747	83
				25%	402	-	97
2D-(3,3,256,512)	21.6	0.4	3724	85%	66	192	29
				60%	74	411	50
				25%	97	-	-
2D-(3,3,512,512)	31.7	0.7	28872	85%	146	687	40
				60%	194	2495	45
				25%	181	3890	40
2D-(5,5,96,256)	4	0.4	598	85%	51	889	28
				60%	42	3049	8
				25%	34	6554	22
2D-(3,3,384,256)	11.1	0.4	3390	85%	62	1462	9
				60%	79	3572	20
				25%	103	6819	42
3D-(3,3,3,32,32)	0.2	0.2	203	85%	1	25	1
				60%	3	149	2
				25%	2	277	-

which uses a single rank across all dimensions. For FC layers, T3F offers greater flexibility than SVD and QR by supporting multiple ranks and various tensor reshaping options, unlike SVD and QR which rely on a single rank and fixed structure. **Tensor/Matrix Decomposition process.** The time required for tensor or matrix decomposition is a critical factor, often taking several hours in some decomposition methods (Table IV). Decomposition time varies significantly across methods due to differences in tensor size, selected rank(s), computational complexity, optimization strategies, and factorization processes.

Among tensor decomposition methods, TT is the fastest due to its non-iterative, SVD-based process. CP is the slowest, requiring time-consuming iterative optimization and often facing convergence issues, sometimes taking hours to decompose a single tensor. Tucker is also iterative but faster than CP, as we limit decomposition to two dimensions. All tensor decompositions were performed using the Tensorly library [40]. In summary, TT is fastest, followed by Tucker, with CP being the slowest (Table IV); the difference between TT and Tucker is minor.

For FC layers, SVD and QR are more efficient than tensor methods, operating on simpler 2D matrices (Table V). SVD is deterministic but computationally heavier for large layers due to quadratic complexity. QR is generally faster and non-iterative, though rank truncation may be needed for compression. T3F offers greater flexibility but incurs extra cost from tensorization, though still under one second.

Key Observations:

- 1) Different LRF methods exhibit significant variations in terms of ES, parameter count, overall memory, FLOPs, parameter/FLOPs coverage and decomposition time.
- 2) For a fixed parameter count, different LRF methods produce varying FLOPs depending on the layer shape (and vice versa). Similarly, the resulting FM memory (and thus overall memory) can differ significantly between methods.
- 3) FC layers: SVD/QR is the best choice when low inference time or minimal memory footprint is required, as T3F incurs higher FLOPs and additional reshape layers [36]. However, T3F offers higher ES and lower parameter counts, although it offers no solutions at low compression ratios. T3F may be preferable for ML

TABLE V

EVALUATION IN TERMS OF EXTRACTION AND DECOMPOSITION TIME FOR THREE FC LAYERS.

Layer shape	Solution generation time (s)			Comp. ratio	Decomposition time (s)		
	SVD	QR	T3F		SVD	QR	T3F
(400,120)	0.1	0.1	57	85%	0.01	0.01	0.09
				60%	0.01	0.01	0.08
				25%	0.01	0.01	-
(512,512)	0.1	0.1	313	85%	0.11	0.03	0.09
				60%	0.1	0.04	0.24
				25%	0.13	0.03	-
(512,256)	0.1	0.1	98	85%	0.05	0.02	0.1
				60%	0.04	0.02	0.08
				25%	0.04	0.02	-

engineers, as its multiple solutions per compression level enable greater potential for accuracy improvement.

- 4) Conv Layers: All three methods yield similar results in terms of parameter count and FLOPs. However, CP is less suitable for ML engineers as it provides only one solution per compression ratio, and thus there is less potential for accuracy improvement. Additionally, CP is not ideal when decomposition time is an issue.

V. PROPOSED METHODOLOGY AND FRAMEWORK

In this Section, the proposed methodology and framework is provided. In Subsection V-A, we describe the proposed framework when applying a single decomposition method for the Conv layers and a single decomposition method for the FC layers. In Subsection V-B, we introduce a post-processing step that applies hybrid decomposition, where different decomposition methods are employed in different CNN layers, further enhancing compression efficiency.

A. Proposed Methodology and Framework

The main steps of the proposed methodology/framework are shown in the left part of Fig. 8. Each step is further explained below. Initially, the user specifies the input model and the objective function, i.e., minimizing FLOPs, parameter count, or overall memory. Next, the process shown in the top-left box of Fig. 8 is repeated using a fixed LRF method for the Conv layers and a different fixed method for the FC layers.

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 allows for more efficient convergence to an efficient solution. In this study, we choose to factorize 90% of the layers (exclude the small layers with respect to the objective function). This approach balances reduced computational cost (e.g., number of operations) and memory footprint with preserved model accuracy.

Step1. Employ Maximum Compression (rank-1): Since the aim of the proposed methodology is to minimize the objective function, we begin by selecting the optimal baseline: a model in which all layers have a rank of one. This initial configuration consistently yields the lowest objective value, as it results in the smallest number of parameters and FLOPs among all factorized model variants.

Step2. Calibration: To compensate for the accuracy loss caused by factorization, each 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

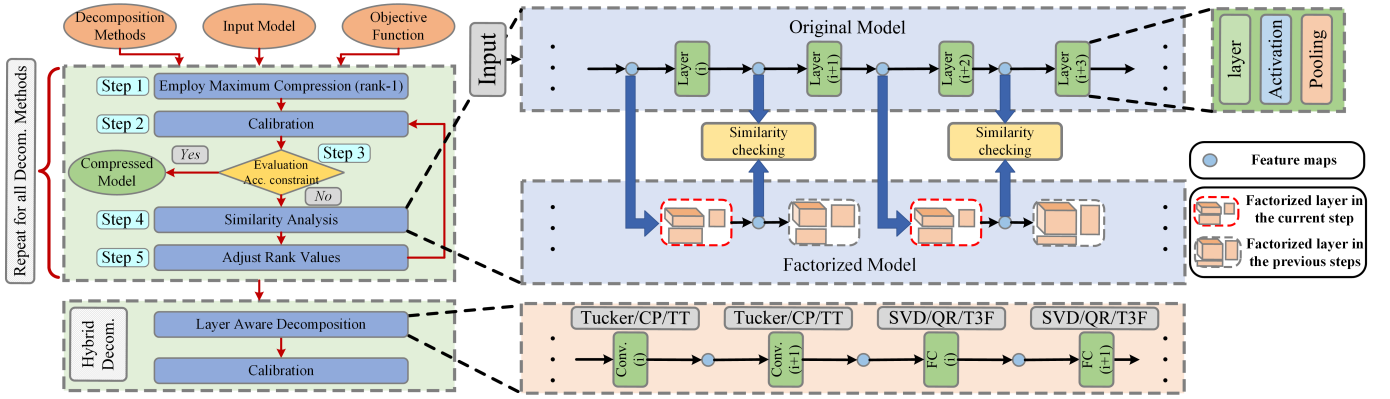


Fig. 8. Proposed Methodology and Framework

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. In the results Section VII we show that the proposed one-shot fine-tuning approach is on average 8× faster compared to iterative fine-tuning.

Step3. Evaluation: After fine-tuning the factorized model, its performance is evaluated. This can be assessed in various ways, e.g., validation loss, validation accuracy, or a combination of metrics. In this study, we use validation accuracy to compare the factorized model against the original model. 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 solutions.

Step4. Similarity analysis: After fine-tuning, if the factorized model does not meet the input accuracy constraint, we must reduce the compression ratio by increasing the rank values. Since each layer impacts accuracy differently, we face a question: how to adjust each layer's ranks to maximize overall compression while maintaining accuracy close to the original model? Previous methods [19], [29] consider layers' weights individually without accounting for their interactions within the model. Another method [39] focuses on the similarity between the factorized and original weights; this approach is time-consuming, because it requires reconstructing the weights from the factorized components and also overlooks layer interactions and calibration effects. To address these issues, we propose a novel similarity-based strategy that compares feature maps, rather than weights, using cosine similarity as the metric.

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 with blue arrows in the right part of Fig. 8): 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 measure 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 highest effect on 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 adaptive method will leverage this information, applying a higher compression rate to the second layer. Finally, the cosine similarity between two vectors is calculated as:

$$\text{cosine similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} \quad (1)$$

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

Step5. Adjust Rank Values: After calculating the cosine similarity between the factorized layers and their original counterparts, we must determine whether to retain the factorized layer as it is or adjust their ranks based on a threshold.

Specifically, if the average similarity exceeds 0.92⁴, we keep the current rank and solution for this layer (i.e., 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 the previous work [11], a fixed step-size of 5% is utilized, where the compression ratio of the target layers was reduced by 5% in each iteration. However, in this study, in order to analyze the impact of this parameter on both the factorization time and the resulting compression ratio, three different step-size values: 2%, 5%, and 10% are employed. The results corresponding to these step-size values are presented and discussed in Section VII.

⁴ The threshold is set to 0.96 for the layers in the non-sequential parts of the model

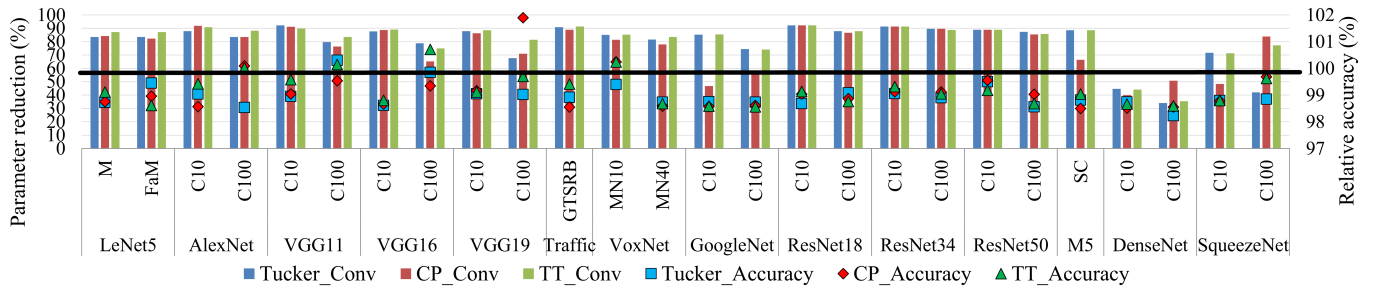


Fig. 9. Comparison with the non-factorized model for Conv layers, using $step-size=2$ and $max-sol=10$.

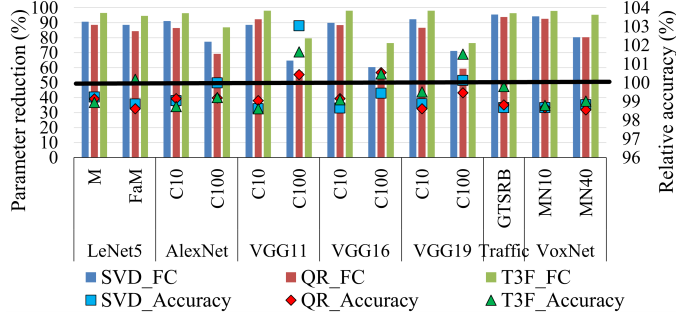


Fig. 10. Comparison with the non-factorized model for FC layers, using $step-size=2$ and $max-sol=10$.

The rationale for using a predefined compression step to increase the rank 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 the $step-size$ approach to select a subset of LRF solutions for each layer and prune the design space. The chosen $step-size$ value introduces a trade-off between achieving 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 TT, multiple LRF solutions may exist for a given compression ratio. In [11], three solutions were used for each compression ratio: i) the solution with the minimum FLOPs, ii) the solution with the maximum FLOPs, and iii) the solution with the same rank values across all dimensions. The first two solutions were selected based on the reasoning outlined in [10], while the third was chosen for its ability to achieve lower approximation error, as discussed in [34].

In this work, we extend the analysis by evaluating the effect of using different number of solutions (let $max-sol$) on the compression ratio and model accuracy. Specifically, we consider three different scenarios: i) a single solution with the minimum FLOPs, ii) three solutions as described above, and iii) ten solutions. The latter scenario includes the three predefined solutions and seven additional randomly selected ones. The results for these scenarios are presented and analyzed in the results section.

After selecting the next solution, we return to step 2 to recalibrate the factorized model. This process is repeated until a configuration that meets the accuracy constraints is found. 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.

B. Hybrid Decomposition - Combining Multiple Methods

In this subsection, the proposed framework is enhanced with a post-processing step that combines the six LRF methods into a hybrid decomposition approach. This is feasible due to the generality and flexibility of the proposed framework.

Combining different LRF methods on a layer-by-layer basis within a single model offers several benefits, allowing each layer to be compressed using the most suitable decomposition strategy based on its structure and computational characteristics. Consequently, it leads to lower overall memory usage and fewer FLOPs.

For example, Tucker decomposition works well for layers with large feature maps, as it can capture the interactions between different dimensions effectively (compresses along multiple dimensions of the tensor). Similarly, CP decomposition can provide advantages in certain layers, although it comes with longer decomposition times due to convergence challenges. By analyzing these layer-specific characteristics, different LRF methods can be strategically combined within the same model to maximize compression while maintaining accuracy.

The hybrid decomposition post-processing step is explained hereafter. First, the proposed framework is applied separately using all three Conv decomposition methods and all three FC methods, resulting in nine combinations. Next, the best-performing method is identified and employed for each layer based on the target objective function. The resulting hybrid model is fine-tuned to recover any lost accuracy. If the calibration process does not achieve the required accuracy, we apply additional fine-tuning. The user can also exclude specific methods from the framework. For example, CP decomposition can be skipped due to its high decomposition time.

Finally, it is important to highlight that since LRF methods preserve the input and output feature map shapes, they maintain compatibility with adjacent layers, ensuring seamless integration across the network. Moreover, as the performance of each LRF method has already been evaluated for individual layers, the post-processing step involves systematically analyzing layer-wise results to select the most effective method for each layer. This structured selection process provides a practical pathway for optimally integrating different LRF methods, maximizing compression while maintaining accuracy and most importantly without introducing additional computational complexity.

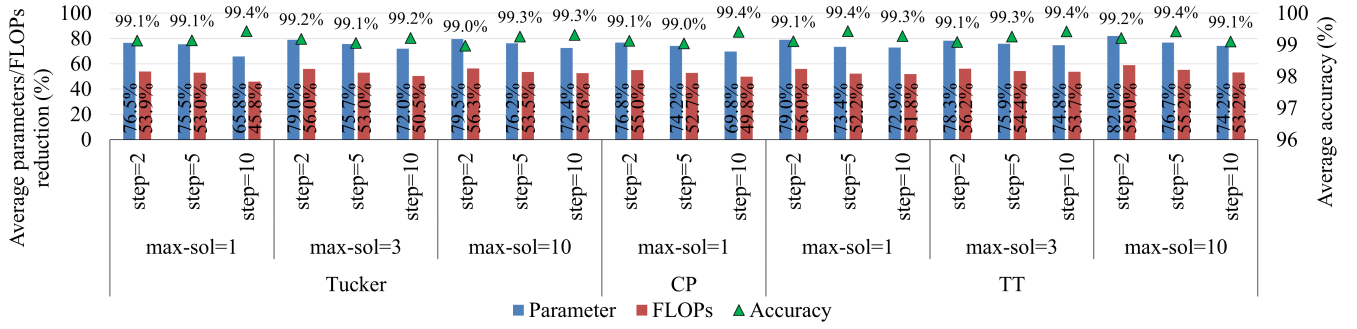


Fig. 11. Evaluating the impact of *step-size* and *max-sol* values on the proposed methodology for Conv layers

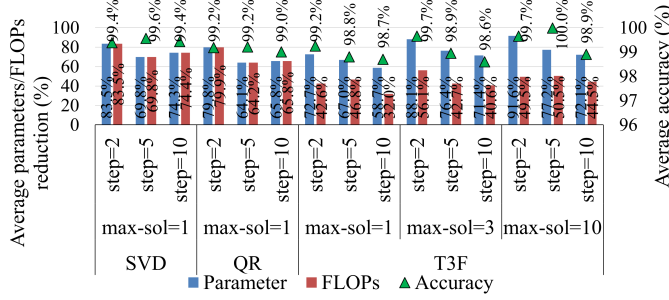


Fig. 12. Evaluating the impact of *step-size* and *max-sol* values on the proposed methodology for FC layers

VI. EXPERIMENTAL SETUP

To demonstrate the effectiveness of our approach, we evaluated it using 14 CNN models with different basic blocks i.e., i) sequential-blocks: LeNet5, AlexNet, VGG11/16/19, Traffic Sign, VoxNet (3D Conv), M5 (1D 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 (FaM), ModelNet10 (MN10), ModelNet40 (MN40), Speech Commands (SC), 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 $1e-4$. In addition, the ReduceLROnPlateau learning rate scheduler is used and 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. The validation accuracy is used to assess the performance of factorized models. In all cases, we set 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 the proposed methodology against the following approaches:

- 1) **Non-Factorized model.**
- 2) **Variational Bayesian Matrix Factorization (VBMF) [19]:** 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.

- 3) **Filter-Based Pruning (FBP) [7]:** 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) [7]). For a fair comparison, the metric that offers the highest compression ratio, while keeping the accuracy drop to less than 1.5%, is selected.

All experiments were conducted using Tensorflow 2.15, Python 3.9.18, Tensorly 0.8.1 (for tensor decomposition), and Numpy 1.24.0 (for matrix decomposition). The experiments were carried out on a system with Ubuntu 22.04 OS and equipped with an Intel Xeon Silver 4309Y CPU at 2.80 GHz. It includes an NVIDIA A40 GPU and 256 GB RAM.

VII. EXPERIMENTAL RESULTS

Comparison with the non-factorized model. Fig. 9 and Fig. 10 present the evaluation results against the original, non-factorized model, for the Conv and FC layers, respectively. The first graph is normalized to the Conv part, the second to the FC part.

As shown in Fig. 9, the proposed methodology achieves an average parameter reduction of 79.5% in the Conv part (up to 92.3% for VGG11 and ResNet18 on the C10 dataset), 76.8% (up to 92.3% for ResNet18 on the C10 dataset), and 82% (up to 92.3% for ResNet18 on the C10 dataset) for Tucker, CP, and TT decompositions, respectively. As is evident from Fig. 9, in most cases, the three decomposition methods achieve similar compression ratios, with differences of less than 5%. However, the method yielding the highest compression ratio varies across models. In certain cases, significant differences in compression ratios are observed. Specifically, in GoogleNet on C10 and C100, M5 on the SC dataset, and SqueezeNet on C10, Tucker and TT decompositions outperform CP decomposition. Conversely, in DenseNet on C100, CP decomposition surpasses Tucker and TT decompositions. Additionally, in

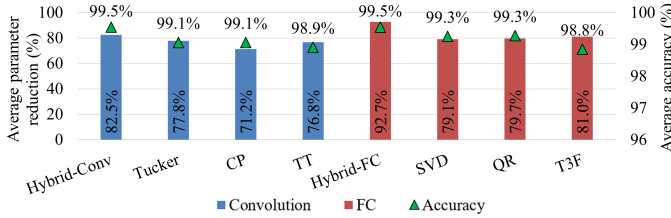


Fig. 13. Evaluation of hybrid decomposition when the objective function is parameter count

SqueezeNet on C100, TT and CP decompositions achieve higher compression ratios than Tucker decomposition.

Furthermore, in certain cases, the factorized models exhibit improved accuracy compared to their original counterparts. Specifically, this accuracy enhancement is observed in VGG11 on C100 dataset with Tucker decomposition; AlexNet and VGG19 on C100 dataset, as well as VoxNet on MN10 dataset, with CP decomposition; and VGG11 and VGG16 on C100 dataset, along with VoxNet on MN10 dataset, with TT decomposition.

For the FC layers in Fig. 10, we show the parameter reduction achieved using SVD, QR, and T3F decomposition methods, focusing only on the FC part of the model. Models without a substantial FC component, such as those with only a final classification layer, are excluded, as matrix decomposition is not meaningful in those cases. As Fig. 10 indicates, the proposed methodology achieves an average parameter reduction in FC part of 83.5% (up to 95.5% in Traffic on GTSRB dataset), 79.8% (up to 93.9% in Traffic on GTSRB dataset), and 91.6% (up to 98% in VGG11, VGG16, and VGG16 on C10 dataset) for SVD, QR, and T3F decompositions, respectively.

As illustrated in Fig. 10, unlike Conv methods where no single approach consistently outperforms the others, T3F decomposition consistently surpasses SVD and QR in FC layers across all cases, achieving up to 20% higher compression in certain instances. Additionally, with the exception of VGG11 on the C10 dataset, where QR decomposition yields slightly better results, SVD generally outperforms QR across all evaluated cases. Furthermore, like Conv methods in certain cases, the factorized models exhibit improved accuracy compared to their original counterparts.

Effect of Threshold Selection on Methodology Performance. As discussed in Section V-A, our methodology relies on two key parameters: *step-size* and *max-sol*. The *step-size* controls how the rank is increased when a layer fails to meet the similarity threshold, while *max-sol* sets the maximum number of solutions explored per step, since some decomposition methods yield multiple candidates. In this subsection, we assess the effect of these parameters, using *step-size* values of 2, 5, and 10, and *max-sol* values of 1, 3, and 10.

Fig. 11 presents the average parameter and FLOPs reductions (bars) across all studied models and datasets, along with the average relative accuracy (triangle marks), for all nine scenarios in the Conv layers using Tucker, CP, and TT decompositions. Note that for CP, there is only one configuration related to *max-sol*, as each LRF solution is determined by a single rank value. As observed in Fig. 11, for a fixed number of solutions per step, there is an inverse relationship between parameter/FLOPs reduction and *step-size*. Specifically, when

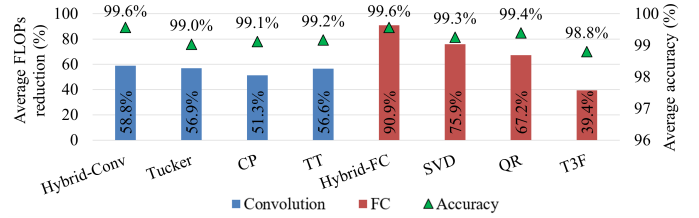


Fig. 14. Evaluation of hybrid decomposition when the objective function is FLOPs

the *step-size* is set to two, the highest compression ratio is achieved. This occurs because a smaller *step-size* enables a smoother rank increase, providing more intermediate solutions to explore, which increases the likelihood of achieving higher compression.

Similarly, for a fixed *step-size*, the *max-sol* generally exhibits the same trend: higher compression is expected when selecting more solutions per step. For instance, when the *step-size* is two, selecting three solutions typically results in higher compression than selecting one solution. Likewise, when the *step-size* is 10, selecting 10 solutions tends to yield higher compression than selecting three solutions. However, two exceptions are observed in TT decomposition.

Fig. 12 presents a similar analysis, for FC layers. Similar to CP decomposition, in both SVD and QR always *max-sol*=1, as there is a single rank value. As shown in Fig. 11, there is an inverse relationship between *step-size* / *max-sol* and the achieved compression ratio. However, when considering FLOPs reduction instead of parameter reduction in T3F decomposition, this trend does not hold (Fig. 12). This deviation arises from the fact that, unlike SVD and QR, there is no linear relationship between parameter count and FLOPs in T3F [36].

Hybrid Decomposition. In this Subsection we evaluate the post-processing step that combines the six LRF methods. For clarity and focus, this subsection evaluates the hybrid method against Conv and FC methods separately, when the objective function is parameter reduction (Fig. 13) and FLOPs reduction (Fig. 14). The results are compared with the best performing configurations of other decomposition methods. As expected, the Hybrid approach outperforms the individual methods, achieving a parameter reduction improvement ranging from 4.7% to 11.3% in the Conv layers and from 11.7% to 13.6% in the FC layers. Furthermore, the hybrid model exhibits superior accuracy compared to other methods, demonstrating the effectiveness of this approach.

A similar analysis is presented in Fig. 14, when the objective is to maximize FLOPs reduction. As expected, the Hybrid model outperforms the other decomposition methods, achieving from 1.9% to 7.5% in Conv layer and from 15% to 51.5% in FC layers FLOPs reduction. Likewise for this objective, the hybrid model exhibits superior accuracy compared to other methods, demonstrating the effectiveness of this approach.

Comparison against VBMF and FBP. This subsection evaluates the proposed framework in Subsection V-A with VBMF and FBP. We used Tucker decomposition for Conv layers with *step-size*=5 and *max-sol*=3, and SVD for FC layers with a *step-size*=2. In all cases, the target layers, fine-tuning epochs, and accuracy drop constraints remain consistent across all studied methods. For FBP, three different filter pruning metrics are considered and the best-performing one is selected. For

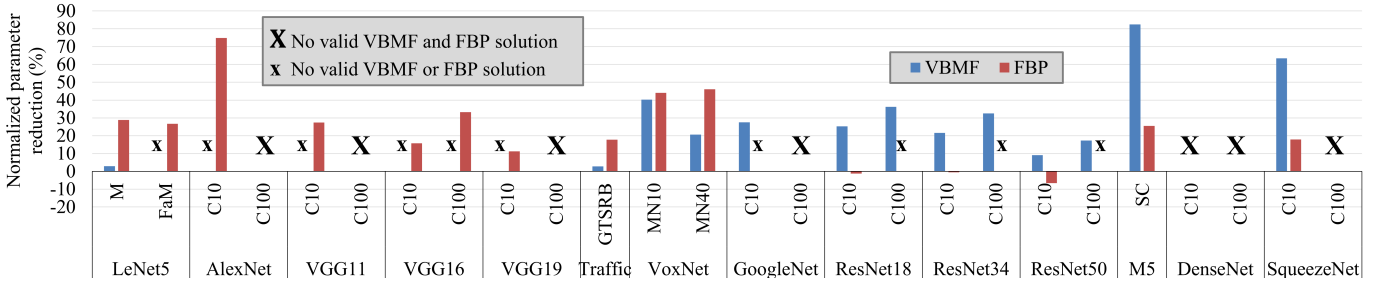


Fig. 15. Evaluation with VBMF and FBP. Tucker and SVD were used in all cases.

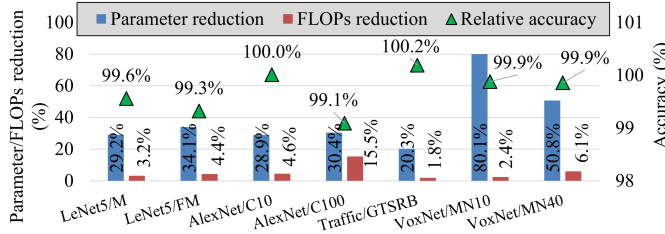


Fig. 16. Comparison of LRF vs. LRF+FBP. FBP is employed on the last Conv layer only.

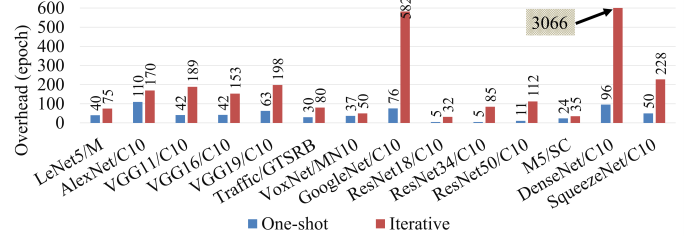


Fig. 17. Overhead of One-shot vs. iterative fine tuning.

VBMF, we use Tucker decomposition for Conv layers and SVD for FC layers. Figure 15 presents the relative parameter reduction achieved by our method over VBMF and FBP. Due to space constraints, only parameter reduction is shown, as FLOPs reduction exhibit similar trends. A capital “X” indicates the cases where both VBMF and FBP fail to meet accuracy constraints, while a lowercase “x” marks cases where at least one fails.

As seen in Fig. 15, VBMF and FBP fail to meet accuracy constraint in 13 and 11 out of 26 cases, respectively, demonstrating the robustness of our method. In all remaining cases, our approach consistently achieves higher compression ratios than VBMF, with up to 60% additional compression (e.g., SqueezeNet on CIFAR-10). This advantage stems from our methodology’s ability to account for inter-layer dependencies, unlike VBMF, which compresses each layer independently.

A similar trend is observed when comparing our approach against FBP. The fundamental distinction is that LRF approximates layer weights, whereas FBP remove network components (e.g., filters). Our approach outperforms FBP by 12% to 75% in most cases. However, for ResNet18/34/50 on CIFAR-10, FBP achieves up to 8% higher compression; this is because FBP inherently benefits from implicitly compressing the next layer. Notably, in these cases, our method still reduces parameters by over 85% relative to the original model (Fig. 9).

Compatibility with Other Compression Techniques: Applying FBP on Top of LRF. A key strength of our approach is its compatibility with additional compression methods, such as FBP. As shown in Fig. 16, applying FBP on top of the proposed LRF method yields further compression benefits. In this experiment, FBP is applied only to the last Conv layer, which indirectly reduces the parameter count of the first FC layer (typically one of the largest) by decreasing its input channels. Fig. 16 shows the additional parameter and FLOPs reduction achieved by the combined method, normalized to the LRF-only baseline. This integration results in an extra 21%–80% reduction in memory usage.

Fine-Tuning Time Evaluation This Subsection evaluates the one-shot fine-tuning strategy used in our methodology. Unlike other approaches that re-train the model for each configuration [32] or fine-tune after every layer is factorized [41], our method performs a single fine-tuning pass after all targeted layers are factorized, avoiding costly iterative calibration or retraining. To assess the efficiency of this one-shot approach, we modified our framework to perform iterative fine-tuning, where each layer is factorized and immediately fine-tuned. For this experiment, we used Tucker decomposition with a step-size of 5 and max-sol of 3 for Conv layers, and SVD with a step-size of 2 for FC layers.

Fig. 17 compares the overhead time of the proposed method (one-shot) and iterative fine-tuning method. The overhead time accounts for the overall execution time required to construct the factorized model, such as decomposing the weights or calculating the similarity between the factorized and original models. This overhead time is normalized to the duration of a single training epoch for each model. Only one dataset is shown for brevity.

Fig. 17 clearly demonstrates that the one-shot method consistently achieves significantly lower overhead in all evaluated scenarios. Specifically, the reduction in overhead reaches up to 31.9× (DenseNet model on C10 dataset), with an average reduction of approximately 8× across the entire models. Furthermore, the gap in overhead becomes more pronounced as the complexity and size of the models increase. For larger and deeper models, characterized by more layers, the iterative fine-tuning process becomes increasingly computationally expensive due to repeated calibration over every single layer. In contrast, the one-shot method avoids this iterative burden by applying a targeted update in a single step, making it more scalable and efficient for modern, large-scale architectures. These results indicate that the proposed one-shot method becomes increasingly effective as model size and complexity grow, highlighting the suitability of the proposed method for efficiently handling the fine-tuning of large and deep models.

VIII. CONCLUSION

This paper presents an end-to-end DSE methodology and framework that formulates LRF as a multi-objective optimization problem. Unlike existing approaches, our method introduces a more efficient rank selection strategy based on feature map similarity (rather than weight similarity), significantly reduces fine-tuning time, supports all CNN layer types and six LRF techniques, and is compatible with additional compression methods such as FBP. Notably, it also supports hybrid decomposition for the first time, allowing different LRF methods to be applied per layer to enhance overall compression. Additionally, we present the first in-depth analysis of six LRF methods, uncovering several key insights. For example, we observe that LRF solutions are unevenly distributed across the Memory–FLOPs space. Moreover, for a fixed parameter count, different methods yield varying FLOPs and overall memory, depending on the layer shape (and vice versa). Furthermore, we highlight that no single LRF method is universally ideal, as each involves distinct trade-offs.

Our framework outperforms state-of-the-art techniques like VBMF and FBP in compression efficiency while preserving accuracy. Moreover, we demonstrate that LRF can be effectively combined with FBP for further gains. Future work will focus on refining similarity thresholds and extending the framework to other architectures, such as transformers.

APPENDIX

Table VI shows the range of different metrics used in Fig. 6 and Fig. 7. Higher score means better. **Rank Configuration:** This metric indicates the degree of configurability in selecting decomposition ranks for each method. It reflects how many independent ranks can be chosen. **Best/Worst Param/FLOPs count:** Denotes the maximum and minimum parameter/FLOPs reduction achieved by each method. The "best" value corresponds to the configuration yielding the highest parameter/FLOPs reduction, while the "worst" indicates the least parameter/FLOPs reduction within feasible configurations. **Best/Worst overall mem:** Measures the total memory usage, accounting for both parameters and intermediate feature maps. The best case indicates whether overall memory usage improves (i.e., is reduced) compared to the original layer. The worst case captures how much the total memory footprint increases, or at best, whether there is no increase. **ES:** Reflects the size of the practical design space for LRF solutions. This metric counts the number of distinct decomposition configurations that are feasible for a given method. **Param/FLOPs coverage:** Quantifies the variability or range in parameter and FLOPs reduction across different configurations. It is defined as the difference between the best and worst reduction values. **Flexibility:** Captures the method's adaptability to different layer shapes and rank settings. Several flexibility modes are defined. *Shape + Ranks:* The method supports diverse decomposition shapes for a single layer structure and allows setting multiple ranks. *Per-dim ranks:* The method can selectively skip decomposing certain dimensions and assign independent ranks to others. *Multiple ranks:* All tensor dimensions are decomposed, but each can have a

distinct rank. *Fixed:* A single rank must be applied uniformly across all dimensions. *Rigid:* Applies only to matrix structures, with a single dimension and fixed rank. **Decomposition Time:** Indicates the computational cost (time) required to decompose a given weight tensor or matrix. This reflects the method's practical overhead during model compression or deployment.

REFERENCES

- [1] M. E. Mswahili and Y.-S. Jeong, "Transformer-based models for chemical SMILES representation: A comprehensive literature review," *Heliyon*, vol. 10, p. e39038, Oct. 2024.
- [2] 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*, vol. 10, p. 2470, Oct. 2021.
- [3] H. Hussain, P. S. Tamizharasan, and C. S. Rahul, "Design possibilities and challenges of DNN models: a review on the perspective of end devices," *Artificial Intelligence Review*, vol. 55, pp. 5109–5167, Oct. 2022.
- [4] 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," *IEEE Access*, vol. 10, pp. 109167–109186, 2022.
- [5] G. C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca, "Deep neural networks compression: A comparative survey and choice recommendations," *Neurocomputing*, vol. 520, pp. 152–170, Feb. 2023.
- [6] M. Gabor and R. Zdunek, "Compressing convolutional neural networks with hierarchical Tucker-2 decomposition," *Applied Soft Computing*, vol. 132, p. 109856, Jan. 2023.
- [7] K. Gkrispanis, N. Gkalelis, and V. Mezaris, "Filter-Pruning of Lightweight Face Detectors Using a Geometric Median Criterion," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, (Waikoloa, HI, USA), pp. 280–289, IEEE, Jan. 2024.
- [8] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, "A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, pp. 1–50, Dec. 2023.
- [9] I. Sarridis, C. Koutlis, G. Kordopatis-Zilos, I. Kompatsiaris, and S. Papadopoulos, "InDistill: Information flow-preserving knowledge distillation for model compression," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 9033–9042, Feb. 2025. ISSN: 2642-9381.
- [10] 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*, vol. 52, pp. 20–39, Apr. 2024.
- [11] M. Kokhazadeh, G. Keramidas, V. Kelefouras, and I. Stamoulis, "A CNN Compression Methodology for Layer-Wise Rank Selection Considering Inter-Layer Interactions," in *2025 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7, Mar. 2025. ISSN: 1558-1101.
- [12] Z. Bai, Y. Li, M. Woźniak, M. Zhou, and D. Li, "DecomVQANet: Decomposing visual question answering deep network via tensor decomposition and regression," *Pattern Recognition*, vol. 110, p. 107538, Feb. 2021.
- [13] G. Frusque, G. Michau, and O. Fink, "Canonical Polyadic Decomposition and Deep Learning for Machine Fault Detection," July 2021. arXiv:2107.09519.
- [14] D. Wang, G. Zhao, H. Chen, Z. Liu, L. Deng, and G. Li, "Nonlinear tensor train format for deep neural network compression," *Neural Networks*, vol. 144, pp. 320–333, Dec. 2021.
- [15] G. Cai, J. Li, X. Liu, Z. Chen, and H. Zhang, "Learning and Compressing: Low-Rank Matrix Factorization for Deep Neural Network Compression," *Applied Sciences*, vol. 13, p. 2704, Feb. 2023.
- [16] I. Ganev and R. Walters, "Model Compression via Symmetries of the Parameter Space," Oct. 2021.
- [17] A. Novikov, P. Izmailov, V. Khurlov, M. Figurnov, and I. Oseledets, "Tensor Train Decomposition on TensorFlow (T3F)," *Journal of Machine Learning Research*, vol. 21, no. 30, pp. 1–7, 2020.
- [18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "[TensorFlow]: A System for {Large-Scale} Machine Learning," pp. 265–283, 2016.

TABLE VI

MAPPING OF QUANTITATIVE THRESHOLDS TO QUALITATIVE LEVELS USED IN THE SPIDER GRAPHS FOR SIX DECOMPOSITION METHODS ACROSS MULTIPLE CHARACTERISTICS.

Metric	5 = Excellent	4 = Good	3 = Moderate	2 = Low	1 = Poor
Rank Configurations	5+ ranks	4-5 ranks	2-3 ranks	1 rank + limits	Fixed
Best param count	>98%	94-98%	90-94%	80-90%	<80%
Worst param count	>20%	10-20%	6-10%	2-6%	<2%
Best FLOPs count	>98%	94-98%	90-94%	80-90%	<80%
Worst FLOPs count	>20%	10-20%	6-10%	2-6%	<2%
Best overall mem	improve >90%	60-90%	30-60%	improve <30%	increase memory
Worst overall mem	No extra memory	increase <25%	increase <25-75%	increase <75-150%	increase >150%
ES	Huge >10 ⁶	extra Large (10 ⁴ -10 ⁶)	Large (10 ³ -10 ⁴)	Medium (10 ² -10 ³)	Small (<100)
Param coverage	>98%	93-98%	85-93%	70-85%	<70%
FLOPs coverage	>98%	93-98%	85-93%	70-85%	<70%
Flexibility	Shape + ranks	Per-dim ranks	Multiple ranks	Fixed rank	Rigid
Decomposition Time	<5s	5-30s	30s-1min	1min-5min	>5min

- [19] Z. Ma, A. E. Teschendorff, A. Leijon, Y. Qiao, H. Zhang, and J. Guo, "Variational Bayesian Matrix Factorization for Bounded Support Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 876-889, Apr. 2015.
- [20] A. Novikov, D. Podoprikin, A. Osokin, and D. P. Vetrov, "Tensorizing Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.
- [21] S. Chen, J. Zhou, W. Sun, and L. Huang, "Joint matrix decomposition for deep convolutional neural networks compression," *Neurocomputing*, vol. 516, pp. 11-26, Jan. 2023.
- [22] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition," Apr. 2015. arXiv:1412.6553.
- [23] M. Astrid and S.-I. Lee, "CP-decomposition with Tensor Power Method for Convolutional Neural Networks compression," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 115-118, Feb. 2017. ISSN: 2375-9356.
- [24] A. Mai, L. Tran, L. Tran, and N. Trinh, "VGG deep neural network compression via SVD and CUR decomposition techniques," in *2020 7th NAFOSTED Conference on Information and Computer Science (NICS)*, pp. 118-123, Nov. 2020.
- [25] 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," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 522-539, Springer International Publishing, 2020.
- [26] 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," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3292-3296, May 2020. ISSN: 2379-190X.
- [27] C. Dai, H. Cheng, and X. Liu, "A Tucker Decomposition Based on Adaptive Genetic Algorithm for Efficient Deep Model Compression," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 507-512, Dec. 2020.
- [28] C. Yang and H. Liu, "Stable Low-Rank CP Decomposition for Compression of Convolutional Neural Networks Based on Sensitivity," *Applied Sciences*, vol. 14, p. 1491, Feb. 2024.
- [29] T. Kim, J. Lee, and Y. Choe, "Bayesian Optimization-Based Global Optimal Rank Selection for Compression of Convolutional Neural Networks," *IEEE Access*, vol. 8, pp. 17605-17618, 2020.
- [30] W. Ahmed, H. Hajimolahoseini, A. Wen, and Y. Liu, "Speeding up Resnet Architecture with Layers Targeted Low Rank Decomposition," Sept. 2023. arXiv:2309.12412.
- [31] M. Astrid, S.-I. Lee, and B.-S. Seo, "Rank selection of CP-decomposed convolutional layers with variational Bayesian matrix factorization," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pp. 347-350, Feb. 2018.
- [32] Y. Sui, M. Yin, Y. Gong, J. Xiao, H. Phan, and B. Yuan, "ELRT: Efficient Low-Rank Training for Compact Convolutional Neural Networks," Jan. 2024. arXiv:2401.10341.
- [33] 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*, vol. 161, pp. 165-177, Apr. 2023.
- [34] 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," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 10464-10472, June 2023.
- [35] M. Kokhazadeh, G. Keramidas, V. Kelefouras, and I. Stamoulis, "A Design Space Exploration Methodology for Enabling Tensor Train Decomposition in Edge Devices," in *Embedded Computer Systems: Architectures, Modeling, and Simulation* (A. Orailoglu, M. Reichenbach, and M. Jung, eds.), (Cham), pp. 173-186, Springer International Publishing, 2022.
- [36] M. Kokhazadeh, G. Keramidas, V. Kelefouras, and I. Stamoulis, "Denseflex: A Low Rank Factorization Methodology for Adaptable Dense Layers in DNNs," in *Proceedings of the 21st ACM International Conference on Computing Frontiers*, (Ischia Italy), pp. 21-31, ACM, May 2024.
- [37] Y. Liu and M. K. Ng, "Deep neural network compression by Tucker decomposition with nonlinear response," *Knowledge-Based Systems*, vol. 241, p. 108171, Apr. 2022.
- [38] J. T. Schuurmans, K. Batselier, and J. F. P. Kooij, "How Informative is the Approximation Error from Tensor Decomposition for Neural Network Compression?," Aug. 2023. arXiv:2305.05318.
- [39] Y.-J. Luo, Y.-S. Tai, M.-G. Lin, and A.-Y. A. Wu, "Similarity-Aware Fast Low-Rank Decomposition Framework for Vision Transformers," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5, May 2024. ISSN: 2158-1525.
- [40] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "TensorLy: Tensor Learning in Python," *Journal of Machine Learning Research*, vol. 20, no. 26, pp. 1-6, 2019.
- [41] M. Astrid and S.-I. Lee, "Deep compression of convolutional neural networks with low-rank approximation," *ETRI Journal*, vol. 40, pp. 421-434, Aug. 2018.