Received 23 December 2024, accepted 6 January 2025, date of publication 23 January 2025, date of current version 29 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3533419

RESEARCH ARTICLE

Convolutional Neural Network Compression via Dynamic Parameter Rank Pruning

MANISH SHARMA[1], (Student Member, IEEE), JAMISON HEARD[2], (Member, IEEE), ELI SABER[1,2], (Senior Member, IEEE), AND PANAGIOTIS MARKOPOULOS[3,4], (Senior Member, IEEE)

[1]Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology, Rochester, NY 14623, USA
[2]Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA
[3]Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78204, USA
[4]Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78204, USA

Corresponding author: Panagiotis Markopoulos (panagiotis.markopoulos@utsa.edu)

This work was supported in part by the National Geospatial-Intelligence Agency (NGA) under academic grant HM0476-19-1-2014 (public release approval NGA-U-2024-01783). The work of Panagiotis Markopoulos was supported in part by the Air Force Office of Scientific Research (AFOSR) under award FA9550-20-1-0039 and in part by NSF under award 2332744.

ABSTRACT While Convolutional Neural Networks (CNNs) excel at learning complex latent-space representations, their over-parameterization can lead to overfitting and reduced performance, particularly with limited data. This, alongside their high computational and memory demands, limits the applicability of CNNs for edge deployment and applications where computational resources are constrained. Low-rank matrix approximation has emerged as a promising approach to reduce CNN parameters, but existing methods often require pre-determined ranks or involve complex post-training adjustments, leading to challenges in rank selection, performance loss, and limited practicality in resource-constrained environments. This underscores the need for an adaptive compression method that integrates into the training process, dynamically adjusting model complexity based on data and task requirements. To address this, we propose an efficient training method for CNN compression via dynamic parameter rank pruning. Our approach integrates efficient matrix factorization and novel regularization techniques, forming a robust framework for dynamic rank pruning and model compression. By using Singular Value Decomposition (SVD) to model low-rank convolutional filters and dense weight matrices, and training the SVD factors with back-propagation in an end-to-end manner, we achieve model compression. We evaluate our method on modern CNNs, including ResNet-18, ResNet-20, and ResNet-32, using datasets like CIFAR-10, CIFAR-100, and ImageNet (2012). Our experiments demonstrate that the proposed method can reduce model parameters by up to 50% and improve classification accuracy by up to 2% over baseline models, making CNNs more feasible for practical applications.

INDEX TERMS Convolutional neural network, dynamic rank selection, image classification, low-rank factorization, model compression, model pruning.

I. INTRODUCTION

The versatility of deep Convolutional Neural Networks (CNNs) is well-documented, finding applications in various areas, such as computer vision [1], [2], [3], remote sensing [4], [5], [6], [7], medical diagnosis [8], and autonomous driving [9], among others. CNNs are favored due to their ability to automatically extract features, promote sparsity and weight sharing, and for their end-to-end trainability. However, as CNNs are increasingly utilized to tackle complex problems, their underlying models have become more sophisticated, employing a large number of trainable parameters in the form of convolutional filters and fully-connected weight matrices [2]. While these large-scale models are viable

The associate editor coordinating the review of this manuscript and approving it for publication was Jeon Gwanggil.

in computer vision applications with abundant training data and computational resources, they pose challenges in environments with limited training examples or computational resources, such as remote sensing and edge computing [10]. Furthermore, traditional CNN architectures are static and cannot adapt their complexity during training to match the intricacies of the data, leading to potential overfitting on simple tasks, which not only results in inefficient learning but also hinders their deployment in scenarios where resource efficiency is paramount.

Several model compression techniques have been proposed to address this issue, including knowledge distillation [11], [12], [13], [14], [15], quantization [16], [17], [18], [19], [20], pruning [17], [21], [22], and special convolution operations [23], [24], [25]. However, the majority of these methods target pretrained models and do not integrate compression into the training process, which prevents the exploitation of potential efficiencies during learning. This lack of integration can lead to performance degradation after compression and typically necessitates retraining. Consequently, methods that can adaptively compress models during training are highly desirable, reducing redundancy without compromising performance or requiring extensive post-training modifications. In contrast, low-rank factorization methods offer a promising model compression approach [6], [26], [27] as they approximate weight matrices/convolutional filters with low-rank matrix/tensor factors, yielding efficient model compression [10], [28], [29], [30], [31]. Nonetheless, the successful deployment of such a low-rank factorized model necessitates meticulous rank selection tailored to the baseline model architecture and the complexity of the data or task at hand, a process that is often challenging and infeasible before training. Our motivation stems from these challenges; we aim to develop a dynamic compression method that eliminates the need for predetermined ranks and post-training interventions. By enabling adaptive rank determination during training, we strive to make CNNs more efficient and practical for deployment in diverse, resource-constrained environments.

Low-rank matrix factorization approaches can, in general, be divided into three categories: (i) post-training low-rank factorization followed by pruning and fine-tuning [28], [29], [30], [31]; (ii) low-rank factorization prior to training with a fixed architecture [6], [32]; and (iii) models factorized prior to training with an adaptable architecture approach [33], [34], [35], [36]. The third category has recently attracted interest due to its ability to leverage redundancies in trainable parameters during training, thus saving computational resources. However, methods in this category often have limited applicability in terms of the type of layer they act on, require post-training interventions and fine-tuning or retraining, and can lead to improper convergence and performance deterioration.

To address the above, our research introduces the following key contributions to the field of CNN compression:

1) *Dynamic Parameter Rank Pruning (DPRP)*: We introduce DPRP, a novel training method that integrates compression directly into the CNN training pipeline. It dynamically adapts the rank of parameter layers over the course of training based on data and task complexity, leveraging Singular Value Decomposition (SVD) with innovative parameter matrix reshaping to model the convolutional filters and dense weight matrices. Unlike traditional methods, DPRP does not require pre-determined ranks or post-training adjustments, allowing for adaptive and real-time compression during the training process.
2) *Innovative Regularization Techniques*: We have developed a framework that combines efficient matrix factorization with novel regularization techniques. These regularizations enforce SVD conditions during training, promoting orthogonality, hierarchical sorting of singular values, and sparsity in minor singular values. This combination of techniques is designed to enhance the efficiency of rank pruning, resulting in effective model compression while preserving or even enhancing performance.
3) *Efficiency Across Various Datasets and Models*: Our method demonstrates significant effectiveness in compressing various baseline models across a range of training datasets. It maintains or even enhances model performance, showcasing its adaptability and utility in diverse application scenarios.

The remainder of this paper is organized as follows. Section II offers a comprehensive literature review on network compression. Our proposed method is presented in Section III, followed by extensive experimental studies in Section IV. Subsequent Sections V and VI present discussions and concluding remarks, respectively.

II. RELATED WORK

A. KNOWLEDGE DISTILLATION

In the literature, numerous techniques have been proposed to address the model compression. A prominent approach is knowledge distillation, where a large, accurate model (the teacher) guides a smaller model (the student) by an appropriate transfer of knowledge [11]. Although this technique improves the efficiency of the student model by leveraging the rich representations learned by the teacher model, most current methods focus on distilling knowledge after the teacher model has been trained [12], [13], [14], [15], potentially missing opportunities for compression during the training process itself.

B. QUANTIZATION

Quantization, another model compression technique, reduces the precision of network parameters and activations to decrease memory footprint and accelerate computations [16], [17], [18]. However, these techniques struggle to balance quantization-induced loss while maintaining sufficient model capacity; and most methods focus on post-training quantization [19], [20], leaving the potential for exploring in-training quantization that allows for simultaneous compression. Recent advancements have addressed this issue through quantization-aware training (QAT) and dynamic precision quantization. References [37] and [38] propose frameworks for QAT, preserving model accuracy while improving computational efficiency. Additionally, mixed-precision QAT has been explored to enhance computational efficiency and reduce energy consumption in photonic neural networks [39].

C. PRUNING

Pruning techniques have also been employed for model compression by identifying and removing redundant or less important parameters [17]. However, these techniques usually involve an iterative process of pruning and subsequent fine-tuning, which can be computationally expensive [40], [41], [42]. Despite the majority of pruning methods being implemented post-training [21], [22], [43], some recent approaches have considered pruning during the training phase, predominantly concentrating on enforcing sparsity or binary weights [44], [45].

D. EFFICIENT CONVOLUTIONAL LAYERS

There has been interest in specially designed convolutional layers [46], such as depth-wise separable convolutions, for their potential to reduce model complexity. These layers aim to factorize standard convolutions into separate depth-wise and point-wise convolutions, decreasing the number of parameters and operations. However, current studies primarily focus on replacing standard convolutions in predefined architectures [23], [24], [25], leaving unexplored research space for adaptive and dynamic integration of such layers during training.

E. LOW-RANK METHODS

Low-rank factorization approaches play a vital role in model compression by reducing the architecture and size of the factorized model [47], [48], [49], [50]. Depending on the operational characteristics of low-rank matrix factorization, these methods can, in general, be divided into three categories.

1) COMPRESSION POST-TRAINING

The first category involves post-training low-rank factorization followed by pruning and fine-tuning [28], [29], [30], [31], [51]. Similar to other model compression techniques, these methods do not prioritize model compression during training, leading to a performance decline after pruning. Extensive retraining is required to restore model performance.

2) COMPRESSION PRIOR-TRAINING

The second category is defined by low-rank factorization before training with a fixed architecture [6], [32]. In this approach, the low-rank factors are trained during the training phase, making these methods more resilient to performance degradation after pruning, and thus requiring less retraining for fine-tuning. However, determining the appropriate ranks for factorization in both these methods requires considerable effort/time and multiple iterations. Moreover, enforcing a uniform compression rate across all network layers is inefficient, as different layers exhibit varying degrees of redundancies and susceptibility to compression. This uniform low-rank strategy often leads to deteriorated performance. There are methods that emerge as a mixture of the above two approaches, [52], utilizing training with full-rank decomposition while maintaining SVD conditions in the process followed by post-training singular values pruning and fine-tuning to recover the degraded performance.

3) COMPRESSION WHILE-TRAINING

Recently, attention has shifted towards the third category of low-rank factorization, which involve factorizing models before training with an adaptable architecture approach [33], [34], [35], [36]. In this approach, models are generally factorized initially with full rank. During the training process, the factors are gradually transformed into low-rank structures. These methods exploit redundancies in trainable parameters during training, eliminating the need for post-training fine-tuning thereby saving effort, time, and computational resources. To this effect, one study [33] applied this approach to speech recognition, wherein only the fully-connected layers were factorized with actual model compression conducted post-training.

Within image classification, [53] suggested the use of rank-adaptive evolution on a low-rank manifold for training and compression of networks. This approach, interestingly, avoids the need for full weight representation but it was limited to matrix-valued layers only. In another attempt, Tucker-2 decomposition was used to factorize convolutional layers with regularization gates and funnel function to determine suitable ranks [35]. However, model compression was implemented post-training followed by a fine-tuning stage that incorporated the evaluation of computational costs relative to the original baseline model, layer swapping, and training of the resultant network from scratch. Another study proposed a budget-aware Tucker-2 compression approach taking model size constraints into account [36]. Imposing stringent constraints on model capacity during the training phase showed an improper convergence in the rank and accordingly in the number of trainable parameters over the course of training across different layers. With a new training strategy that alternates between low-rank approximation and standard training after a set number of optimization iterations, Tensor Rank Pruning (TRP) [34] exploits both space-wise [54] and channel-wise [55] correlations to decompose convolutional filters. Unlike the approach of training from scratch, this method is employed during training.

However, these investigations highlight a need for a truly dynamic model compression method that trains from scratch and adaptively determines rank based on data and task complexity, without post-training fine-tuning or interventions. While the aforementioned approaches emphasize adaptivity and efficiency, recent methods have also begun to explore complementary objectives like adversarial robustness. For example, a robust low-rank training algorithm [56] enforces approximate orthonormal constraints on factorized weight matrices in fully-connected layers, preserving adversarial robustness and accuracy. While this robust approach focuses on stability and resilience, our method prioritizes dynamic rank adaptation across both fully-connected and convolutional layers, eliminating the need for post-training interventions. By integrating compression into the learning pipeline and adjusting ranks on-the-fly, we directly address the gap identified in prior works, ensuring both efficiency and performance.

In doing so, our approach directly addresses this gap, operating from scratch and adaptively adjusting ranks for both fully-connected and convolutional layers, thus fully integrating compression into the training pipeline and ensuring both efficiency and performance.

Furthermore, based on the type of factorization, low-rank factorization methods, in general, can be categorized into matrix and tensor methods [33], [35], [36], [57]. While some tensor-based low-rank factorization methods provide a wider scope for compression [30], [58], [59], [60], they often require the determination of multiple ranks per layer in the network, making their appropriate selection a tedious task. Therefore, our proposed approach utilizes the SVD matrix factorization method.

III. PROPOSED METHOD

CNNs primarily consist of convolutional and fully-connected layers. In a convolutional layer, as shown in Fig. 1, trainable parameters reside in the convolutional filter. In a fully-connected layer, as shown in Fig. 2, trainable parameters are arranged in dense weight matrix. In this work, we demonstrate how SVD matrix factorization, coupled with proposed regularizations, can effectively model these elements of deep CNNs for dynamic compression via parameter rank updates during training. This, in turn, reduces redundancy and enhances performance, even when applied to optimized, efficient, standard and state-of-the-art deep CNNs.

A. NOTATION AND SVD PRELIMINARIES

Throughout this paper, we adhere to the following notation: scalar variables are represented by lowercase letters (e.g., x), vectors are indicated by boldface lowercase letters (e.g., $\mathbf{x}$), matrices are denoted by boldface uppercase letters (e.g., $\mathbf{X}$), and tensors are signified by underscored boldface uppercase letters (e.g., $\underline{\mathbf{X}}$). The identity matrix is symbolized by $\mathbf{I}$, and real numbers are signified by $\mathbb{R}$. To represent the entries of a vector, matrix, or tensor, we use the notation $[\cdot]_i$, where i denotes a set of indexes. $\mathbf{X}^T$ denotes the transpose of $\mathbf{X}$.

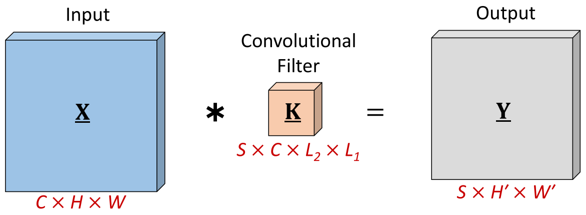


FIGURE 1. A typical convolutional layer.

We use $|x|$ to denote the absolute value of a scalar x. The ℓ_1 norm of a vector $\mathbf{x}$ is $||\mathbf{x}||_1 = \sum_i |[\mathbf{x}]_i|$, i.e., the sum of the absolute values of its elements. For a matrix $\mathbf{X}$, the ℓ_1 norm is the sum of absolute values of all its elements: $||\mathbf{X}||_1 = \sum_{i,j} |[\mathbf{X}]_{i,j}|$. The Frobenius norm of a matrix $\mathbf{X}$ is given by $||\mathbf{X}||_F = \sqrt{\sum_{i,j}[\mathbf{X}]_{i,j}^2}$, and the ℓ_2 norm (Euclidean norm) of a vector $\mathbf{x}$ is $||\mathbf{x}||_2 = \sqrt{\sum_i [\mathbf{x}]_i^2}$. Throughout this paper, these norms are used to quantify magnitudes and constraints on vectors and matrices in our low-rank factorization and compression framework.

Compact SVD, also referred to as SVD in this paper, is a powerful mathematical technique extensively utilized across various domains, including dimensionality reduction, data compression, and collaborative filtering [33], [61]. It decomposes a matrix into: the left singular vectors $\mathbf{U}$, the singular values (σ) in diagonal matrix Σ, and the transposed right singular vectors $\mathbf{V}^T$. In mathematical terms, given $\mathbf{A} \in \mathbb{R}^{m\times n}$ of rank r, the SVD factorization is expressed as $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{m\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, and $\mathbf{V}^T \in \mathbb{R}^{r\times n}$. SVD features several crucial properties such as orthogonality, whereby $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, i.e., $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, meaning their columns form an orthonormal basis. Moreover, σ are non-negative and are arranged in descending order, thereby enabling the identification of the most significant components in the matrix. The rank of the matrix can be discerned by examining the number of non-zero singular values, offering insights into the inherent structure and dimensionality of the original matrix.

B. FACTORIZED CONVOLUTIONAL AND FULLY-CONNECTED LAYER

1) CONVOLUTIONAL LAYER

Consider convolutional filter $\underline{\mathbf{K}} \in \mathbb{R}^{S\times C\times L_2\times L_1}$. It is a 4-way tensor comprising S 3-way kernels of pixel width L_1, pixel height L_2, and channel depth C. Each kernel convolves with an input image $\underline{\mathbf{X}} \in \mathbb{R}^{C\times H\times W}$, which is again a 3-way tensor of pixel width W, pixel height H, and channel depth C. The convolution is performed with padding parameters (p_1, s_1) and (p_2, s_1), controlling padding and stride along the width and height of $\underline{\mathbf{X}}$, respectively. The result of the convolution is a 3-way output tensor $\underline{\mathbf{Y}} = \underline{\mathbf{X}} * \underline{\mathbf{K}} \in \mathbb{R}^{S\times H'\times W'}$, where $W' = (W - L_1 + 2p_1)/s_1 + 1$ and $H' = (H - L_2 + 2p_2)/s_2 + 1$, as shown in Fig. 1. In the case of symmetric convolution, which is typically the case, $L_1 = L_2 = L$, $p_1 = p_2 = p$,

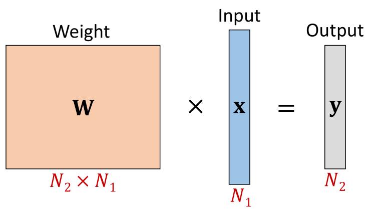


FIGURE 2. A typical fully-connected layer.

and $s_1 = s_2 = s$. The number of trainable parameters contained in a standard convolutional filter is $P_c = SCL_1L_2$.

To factorize a convolutional layer, we first consider reshaping of tensor $\underline{\mathbf{K}}$ into matrix $\mathbf{M} \in \mathbb{R}^{SC \times L_1L_2}$ so that

$$[\underline{\mathbf{K}}]_{s,c,l_2,l_1} = [\mathbf{M}]_{i,j}, \tag{1}$$

where $i = (s-1)C + c$ and $j = (l_2-1)L_1 + l_1$ with $s = 1, 2, \ldots, S$, $c = 1, 2, \ldots, C$, $l_2 = 1, 2, \ldots, L_2$, and $l_1 = 1, 2, \ldots, L_1$. Next, we consider that $\mathbf{M}$ is of rank $r \leq \min\{SC, L_1L_2\}$, attaining SVD $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T$, so that $[\mathbf{M}]_{i,j} = \sum_{g=1}^{r}[\mathbf{U}]_{i,g}[\Sigma]_{g,g}[\mathbf{V}]_{j,g}$, where $i = 1, 2, \ldots, SC$ and $j = 1, 2, \ldots, L_1L_2$. Thus, effectively, through the low-rank structure of $\mathbf{M}$, convolutional filter $\mathbf{K}$ is factorized as

$$[\underline{\mathbf{K}}]_{s,c,l_2,l_1} = \sum_{g=1}^{r}[\mathbf{U}]_{(s-1)C+c,g}[\Sigma]_{g,g}[\mathbf{V}]_{(l_2-1)L_1+l_1,g} \tag{2}$$

for every $s = 1, 2, \ldots, S$, $c = 1, 2, \ldots, C$, $l_2 = 1, 2, \ldots, L_2$, and $l_1 = 1, 2, \ldots, L_1$. The particular reshaping/matricization of $\underline{\mathbf{K}}$ to $\mathbf{M}$ was selected in order to reduce the number of trainable parameters and computational overhead. Instead of training the entries of $\underline{\mathbf{K}}$, we train the entries of its factors in $\mathbf{U} \in \mathbb{R}^{SC \times r}$, $\mathbf{V} \in \mathbb{R}^{L_1L_2 \times r}$, and $\Sigma \in \mathbb{R}^{r \times r}$. Thus, the number of trainable parameters in a factorized convolutional layer is given by $P_{fc} = r(SC + L_1L_2 + 1)$. Accordingly, the proposed factorization constitutes parameter compression when $P_{fc} \leq P_c$ or, equivalently,

$$r \leq \frac{SCL_1L_2}{SC + L_1L_2 + 1}. \tag{3}$$

The corresponding compression rate, as a function of r, is

$$R_{fc}(r) = 1 - \frac{P_{fc}}{P_c} = 1 - \frac{r(SC + L_1L_2 + 1)}{SCL_1L_2}. \tag{4}$$

2) FULLY-CONNECTED LAYER

In the case of a fully-connected layer, a dense weight matrix $\mathbf{W} \in \mathbb{R}^{D_2 \times D_1}$ is multiplied with input $\mathbf{x} \in \mathbb{R}^{D_1}$ resulting in the output $\mathbf{y} = \mathbf{W}\mathbf{x} \in \mathbb{R}^{D_2}$. The number of trainable parameters in a standard fully-connected layer is given by $P_f = D_1D_2$. For a factorized fully-connected layer, $\mathbf{W}$ is considered to be of low rank $r \leq \min\{D_1, D_2\}$, admitting SVD $\mathbf{W} = \mathbf{U}\Sigma\mathbf{V}^T$, so that

$$[\mathbf{W}]_{d_2,d_1} = \sum_{g=1}^{r}[\mathbf{U}]_{d_2,g}[\Sigma]_{g,g}[\mathbf{V}]_{d_1,g}, \tag{5}$$

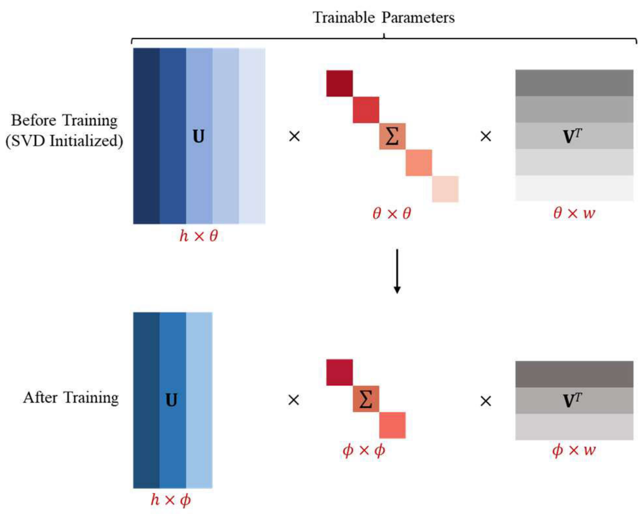


FIGURE 3. The variation in factor sizes, represented by the initial rank θ and the final rank ϕ.

for $d_2 = 1, 2, \ldots, D_2$ and $d_1 = 1, 2, \ldots, D_1$. That is, instead of learning $\mathbf{W}$, the proposed method learns the SVD factors in $\mathbf{U} \in \mathbb{R}^{D_2 \times r}$, $\mathbf{V} \in \mathbb{R}^{D_1 \times r}$, and $\Sigma \in \mathbb{R}^{r \times r}$. Accordingly, the number of trainable parameters in a factorized fully-connected layer is given by $P_{ff} = r(D_1 + D_2 + 1)$. For the factorization to accomplish compression we need $P_{ff} \leq P_f$ or, equivalently,

$$r \leq \frac{D_1D_2}{D_1 + D_2 + 1}. \tag{6}$$

The attained compression rate, as a function of r, is

$$R_{ff}(r) = 1 - \frac{P_{ff}}{P_f} = 1 - \frac{r(D_1 + D_2 + 1)}{D_1D_2}. \tag{7}$$

C. FACTOR INITIALIZATION AND TRAINING

Below we present the proposed training of the parameter factors of a layer, whether convolutional or fully-connected. For ease in notation, we denote $(h = SC, w = L_1L_2)$ if the layer is convolutional or $(h = D_2, w = D_1)$ if the layer is fully-connected. We begin the model training with SVD-factorized convolutional and fully-connected layers of full-rank $\theta = \min\{h, w\}$ with objective to reduce redundancy and converge to final rank ϕ, as illustrated in the Fig. 3. Using SVD-factorized weights as initialization for these layers avoids SVD convergence related issues. Further, starting with a full-rank initialization ensures that the model is similar to baseline in terms of number of trainable parameters and has sufficient capacity to learn the features in accordance with data complexity and task from the beginning.

The SVD structure (orthonormality of singular-vectors and sortment singular values) and preferred low rank are determined implicitly throughout training via intelligently designed loss functions. Accordingly, the total loss function considered for training is

$$L_{total} = L_{app} + \lambda_{str}L_{str} + \lambda_{comp}L_{comp}, \tag{8}$$

where λ_{str} and λ_{comp} are loss-weighing hyper-parameters. L_{app} is the loss function pertinent to the application at hand (e.g., image classification, object detection, segmentation). L_{str} is the loss responsible for maintaining the SVD structure of the parameter factorizations across the layers. Finally, L_{comp} is the function responsible for promoting parameter rank pruning and, thus, model compression. The significance of L_{total} lies in its ability to balance these three aspects—performance, structure, and compression—within a single optimization framework. By carefully selecting the hyper-parameters λ_{str} and λ_{comp} as discussed in Section IV-B, we can control the trade-offs between maintaining high task performance, enforcing the desired mathematical properties of the factorization, and achieving model compression. Next, we present the three proposed losses in detail.

1) APPLICATION LOSS

This loss is pertinent to the application at hand and can vary across CNN deployments. For the sake of numerical experimentation, in this paper we consider an entropy-based classification loss

$$L_{\text{app}} = -\frac{1}{n_c}\sum_{n=1}^{n_c} y_n \ln(\hat{y}_n), \tag{9}$$

where n_c denotes the number of classes in the classification task, y_n indicates the ground-truth, and $\hat{y}_n$ represents the prediction. It is worth noting that this loss term strives to improve classification performance on the training data, regardless of factor structure and compression, which will have to be regulated by the two loss terms presented below.

2) STRUCTURE LOSS

Next, we create a loss term that promotes SVD structure and, thus, facilitates adaptive rank and parameter compression. We recognize that there are two main components in the SVD structure: (i) orthonormality of the singular vectors and (ii) sortment of the singular values. Accordingly, we analyze L_{str} in two corresponding sub-terms: $L_{\text{str}} = \mu_{\text{orth}} L_{\text{orth}} + \mu_{\text{sort}} L_{\text{sort}}$, where μ_{orth} and μ_{sort} are hyper-parameter weights. Denoting by $\{\mathbf{U}_l, \Sigma_l, \mathbf{V}_l\}$ and r_l the SVD-factors and SVD-rank for layer l, respectively, we define

$$L_{\text{orth}} = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{r_l^2}\left(\|\mathbf{U}_l^T\mathbf{U}_l - \mathbf{I}\|_F + \|\mathbf{V}_l^T\mathbf{V}_l - \mathbf{I}\|_F\right), \tag{10}$$

where L is the total number of factorized layers in the network. This loss term promotes orthogonality to the left- and right-hand singular matrices, across all layers, with an emphasis normalized by each layer's rank.

Next, we design a loss term that promotes sortment of the singular values in $\{\Sigma_l\}_{l=1}^L$ so that dynamic truncation could result to optimal low-rank approximation, in accordance with the SVD principles. Specifically, L_{sort} strives to accomplish $[\sigma_l]_j \geq [\sigma_l]_{j+1} \geq 0 \ \forall \ j \in [1, r_l)$ and $\forall \ l \in \{1, 2, \ldots, L\}$. Let the set I_l contain the indices of all singular values of layer l that are out of desired order; that is, $I_l = \{j \in \{2, \ldots, r_l\} : [\sigma_l]_j > [\sigma_l]_{j-1}\}$. Accordingly, define the cardinality (number of entries) of I_l as $\gamma_l = |I_l|$. Also, let η_l denote the number of negative entries in σ_l. Moreover, define function $\chi : \mathbb{N} \to \mathbb{R}_0^+$ such that, for every $a \in \mathbb{N}$, $\chi(a) = 1/a$, if $a > 0$, and $\chi(a) = 0$, if $a = 0$. Then, we define the sorting loss term as

$$L_{\text{sort}} = \frac{1}{L}\sum_{l=1}^{L}\chi(\gamma_l)\sum_{j=1}^{r_l-1}\max\{0, [\sigma_l]_{j+1} - [\sigma_l]_j\} + \chi(\eta_l)\sum_{j=1}^{r_l-1}\max\{0, -[\sigma_l]_j\}. \tag{11}$$

The scaling terms $\chi(\gamma_l)$ and $\chi(\eta_l)$ are used so as to prevent layers with large γ_l and η_l, respectively, from dominating the loss. Overall, L_{sort} promotes that, across l, the entries of σ_l are non-negative and arranged in descending order.

3) COMPRESSION LOSS AND PRUNING STRATEGY

To facilitate dynamic compression we perform dynamic rank pruning. We denote by τ_l the reduced rank of layer l as the highest value of i for which $|[\sigma_l]_{i+1}| > \epsilon |[\sigma_l]_i| \ \forall \ i \in [1, r_l)$, for some pruning threshold $\epsilon \in (0, 1)$ (hyper-parameter). Then, we perform rank pruning by removing all singular values $\{[\sigma_l]_j\}_{j>\tau_l}$ (see Section III-D below). To make sure that this pruning comes with minimum approximation loss, we promote sparsity in $\{[\sigma_l]_j\}_{j>\tau_l}$ by means of the compression loss term:

$$L_{\text{comp}} = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{(r_l - \tau_l)||\sigma_l||_2}\sum_{i=\tau_l}^{r_l}|[\sigma_l]_i|. \tag{12}$$

In L_{comp}, we divide by $||\sigma_l||_2$ in order to prevent layers with relatively larger minimal singular values across layers to dominate the regularization term. Also, we divide by $r_l - \tau_l$ in order to avoid domination by layers with a relatively large number of singular values to be reduced. This arrangement promotes pruning of the minimal singular values, facilitating model compression through dynamic rank pruning in training.

D. MODEL COMPRESSION

While training, our DPRP method operates by continuously monitoring the singular values in each layer's factorized parameters. For a given layer l, we dynamically reduce the value of r_l to τ_l by removing $[\sigma_l]_{i=\tau_l+1}^{r_l}$. Accordingly, the corresponding trainable parameters are removed from $\mathbf{U}_l$ and $\mathbf{V}_l^T$. Since the removed singular values have been reduced throughout training, their influence on the final convolution filter is minimal. Thus, their removal does not significantly affect performance. If $\tau_l = r_l$, no trainable parameters are removed and the network continues training with the same number of trainable parameters as before. At the end of training $\tau_l = \phi_l$, resulting in a compact model, as illustrated in the bottom-half of Fig. 3.

To summarize, the key characteristics of our dynamic rank pruning are as follows:

- **Adaptivity:** The pruning process is not static; it adapts during training based on the evolution of the singular values, allowing the network to adjust its complexity in response to the data and network architecture.
- **Layer-wise Flexibility:** Each layer can have a different rank reduction, reflecting the varying degrees of redundancy and importance across different layers.
- **Integrated into Training:** Unlike methods that apply pruning post-training, our approach integrates rank pruning into the training process, eliminating the need for separate pruning and fine-tuning stages.
- **Minimal Performance Impact:** By focusing on singular values that have been reduced during training, we ensure that pruning has minimal impact on the network's performance, as these singular values contribute less to the overall function of the layer.

E. COMPARISON WITH EXISTING FRAMEWORKS

Several low-rank factorization and compression strategies have been previously proposed, including post-training factorization [28], [29], [30], tensor-based decompositions [30], [36], [58], and methods that rely on a fixed or manually selected rank prior to training [6], [32]. A key theoretical distinction of our approach lies in the dynamic selection of rank factors during training, grounded in the properties of SVD. Unlike methods that predetermine the rank and then project weights onto a low-dimensional subspace, our approach leverages an adaptive pruning threshold based on the singular values' relative magnitudes. This ensures that the retained subspace is continuously updated in response to the evolving training dynamics, potentially leading to a closer approximation of the optimal low-rank subspace at convergence.

From a complexity standpoint, methods that compress models post-training often incur additional computational overhead due to multiple approximation and fine-tuning steps. In contrast, our framework integrates compression into the training process itself, theoretically reducing the need for costly post-processing. Moreover, while some factorization-based methods assume static architectures or fixed ranks that may not align well with data complexity, our dynamic approach imposes fewer structural assumptions. Instead, it uses the descending order of singular values as a natural mechanism to identify and prune redundant parameters. This theoretically positions our method to generalize across a wide range of architectures and training conditions. The flexibility afforded by rank adaptation during training may lead to improved parameter efficiency, particularly when compared to frameworks that operate under fixed low-rank constraints.

In summary, the theoretical novelty of our approach rests on integrating adaptive rank selection seamlessly with ongoing optimization. By doing so, we exploit the structure of SVD and the hierarchical ordering of singular values to maintain a compact and expressive model representation. This contrasts with existing methods, which often treat compression as a separate, post-hoc procedure or rely on static assumptions that may not reflect the underlying data or task complexity.

IV. EXPERIMENTATION

In this section, we detail the experimental datasets, baseline models, evaluation metrics, experimental configurations, and results obtained for the proposed method in comparison to baselines and other comparative approaches for the image classification applications.

A. DATASETS, BASELINE MODELS, AND EVALUATION METRICS

Our image classification experiments utilize three common computer vision datasets: CIFAR-10 [62], CIFAR-100 [62], and ImageNet (2012) [63], consisting of 10, 100, and 1000 classes, respectively. CIFAR-10 and CIFAR-100 datasets both contain 50K training and 10K testing images of 32×32 resolution. For both datasets, samples are uniformly distributed across classes in the train and test sets. The ImageNet dataset, on the other hand, contains approximately 1.2M training images, 50K validation images, and 150K testing images with an average resolution of 469×387. Due to the absence of ground-truth for the test set, the validation set is utilized for testing. Standard transformations and augmentations techniques are employed to increase data variation in an online manner and provide a larger diverse dataset while training [2], [64].

Baseline models for the CIFAR-10 and CIFAR-100 datasets utilize ResNet-20 and ResNet-32 networks, respectively. On the contrary, the ImageNet dataset employs ResNet-18 network as its baseline models [2]. ResNet-20 and ResNet-32 are generally considered smaller networks suitable for CIFAR-10 and CIFAR-100 datasets.

We employ Top-1 and Top-5 accuracies as our primary evaluation metrics for classification performance. Top-1 accuracy is the percentage of times the model correctly predicts the highest ranked class, whereas Top-5 accuracy is the percentage of times the top 5 predictions of the model include the correct class. In addition, MMAC (Mega Multiply-Accumulate operations per second) and GMAC (Giga Multiply-Accumulate operations per second) are used to gauge a model computational complexity, with smaller MMAC/GMAC values denoting faster models.

For comparative methods, in case of code unavailability, results are directly sourced from the corresponding publications. Since we train our baseline model from scratch similar to methods [34], [35], [65], so, our baseline accuracy differs from the comparative method that utilize Torchvision pre-trained weights [36], [59] for baseline accuracy. Thus, for a fair comparison, if the baseline accuracy in the source, A'_{source}, differs from our calculated baseline accuracy, A'_{ours}, resulting from use of pre-trained weights or the randomness in model initialization and other non-deterministic uncertainties, we adopt a scaling method as done in [35] to adjust the comparative accuracy A_{source},

TABLE 1. Values of hyper-parameters λ and ϵ for best performance using different datasets and baseline models.

Dataset	Model	λ_{comp}	ϵ
CIFAR-10	ResNet-20	0.1	0.1
CIFAR-10	ResNet-32	0.5	0.001
CIFAR-100	ResNet-20	0.1	0.1
CIFAR-100	ResNet-32	1.0	0.001
ImageNet	ResNet-18	0.5	0.001

resulting in the scaled accuracy

$$A_{\text{scaled}} = \frac{A'_{\text{ours}}}{A'_{\text{source}}} A_{\text{source}}. \tag{13}$$

B. EXPERIMENTAL CONFIGURATION

We undergo training for ResNet-20, ResNet-32 and ResNet-18 until convergence is observed in the train-test losses. This was accomplished with over 300 epochs for ResNet-20 and ResNet-18, and 150 epochs for ResNet-18. Each network is trained with a batch size of 256 images. The training follows the method detailed in [2] which utilizes the stochastic gradient descent optimizer with a momentum of 0.9, a weight decay of $1e-4$, and an initial learning rate of 0.1. We incorporate a commonly used reduce-on-plateau strategy applied to the classification loss. This strategy involves reducing the learning rate by a factor of 0.1 when the loss does not decrease within a patience interval of 10 epochs, allowing the training to continue with the reduced learning rate.

In factorized models, we employ two distinct sets of hyper-parameters: i) fixed and ii) tunable. The fixed hyper-parameter set $\{\lambda_{\text{str}}, \mu_{\text{orth}}, \mu_{\text{sort}}\}$, remains constant across different baseline models and experimental datasets. In contrast, the tunable hyper-parameters vary to adjust model complexity depending on the baseline model and dataset. The fixed hyper-parameters, used for relative scaling of L_{str} and other loss components, enforce SVD conditions such as orthogonality, as discussed in Section III-A. These were empirically set to $\lambda_{\text{str}} = 1$, $\mu_{\text{orth}} = 1000$, and $\mu_{\text{sort}} = 1$. The high value of μ_{orth} prioritizes the minimization of L_{orth} among other loss terms during training. Additionally, the values of tunable hyper-parameters λ_{comp} and ϵ are determined empirically, based on the specific dataset and baseline model, as shown in Table 1. Hyper-parameter λ_{comp} enforces sparsity in the least singular values and ϵ controls the relative threshold value for pruning the least singular values. Thus, higher values of λ_{comp} accelerate the process of sparsity enforcement in the least singular values, leading to faster compression. Similarly, higher values of ϵ lower the threshold for the difference between the least singular value and the second least singular value, which makes the pruning process more aggressive. This means that the rate of compression is directly influenced by these parameters. Specifically, higher λ_{comp} and ϵ values generally result in a higher compression rate. However, this must be balanced against potential impacts on model accuracy, which necessitates empirical tuning based on the dataset and baseline model. For example, in the case of the CIFAR-10 dataset with the ResNet-20 model, we set $\lambda_{\text{comp}} = 0.1$ and $\epsilon = 0.1$ to achieve a balance between effective compression and maintaining high accuracy. These values were found to provide substantial compression without significant degradation in performance. Similarly, for the CIFAR-100 dataset with the ResNet-32 model, we increased $\lambda_{\text{comp}} = 1.0$ and set $\epsilon = 0.001$ to account for the increased complexity of the dataset and the deeper network architecture. This adjustment ensures sufficient compression while preserving the model's ability to learn the more complex features required for the CIFAR-100 dataset. We acknowledge that the expected compression rate may also depend on the size of the kernels. Larger kernels typically have more parameters and can offer greater opportunities for compression. This is because larger kernels have more singular values that can be pruned without significantly affecting model performance. Our results, as shown in Table 2 and Table 3, demonstrate that our method achieves significant compression while maintaining or even improving accuracy, indicating the effectiveness of our approach across different kernel sizes.

TABLE 2. Comparison of methods on ResNet-20 and ResNet-32 using CIFAR-10, showing Top-1 accuracy and parameter compression. Best results for each evaluation metric are highlighted in bold text.

Method	ResNet-20		ResNet-32	
	Top-1 (%)	Compression (%)	Top-1 (%)	Compression (%)
Baseline	90.98	0.00	92.47	0.00
Std. Tucker [30], [36]	87.15	61.54	87.67	80.39
PSTR-M [59]	89.04	**85.29**	90.57	**82.76**
PSTR-S [59]	90.53	60.00	91.41	62.96
BATUDE [36]	90.75	61.54	92.15	64.29
Proposed 1	90.99	30.66	92.25	52.78
Proposed 2	**92.16**	5.79	**93.03**	24.96

C. RESULTS

1) PERFORMANCE ANALYSIS ON CIFAR-10 DATASET

We compare our proposed method with a baseline and several contemporary methods using the CIFAR-10 dataset on ResNet-20 and ResNet-32 networks. The results are tabulated in Table 2, focusing on Top-1 classification accuracy and the degree of compression in the number of trainable parameters. We do not include FLOPs comparison in the table due to lack of FLOPs information for the comparative method. Two different λ_{comp} and ϵ configurations of the proposed method are presented, namely, proposed 1 and proposed 2. For the ResNet-20 based models, we use $\lambda_{\text{comp}} = 0.5$ and $\epsilon = 0.01$ for proposed 1, and proposed 2 uses λ_{comp} and ϵ values listed in Table 1. For the ResNet-32 based models, we use $\lambda_{\text{comp}} = 1$ and $\epsilon = 0.001$ for proposed 1, and again proposed 2 uses the values from Table 1.

Our observations reveal that both configurations of the proposed method provide the highest Top-1 accuracy for ResNet-20 and ResNet-32 at 90.99% and 92.16%, and at 92.25% and 93.03% respectively, while simultaneously

TABLE 3. Comparison of methods on ResNet-20 and ResNet-32 using CIFAR-100, showing Top-1 accuracy and parameter compression. Best results for each evaluation metric are highlighted in bold text.

Method	ResNet-20		ResNet-32	
	Top-1 (%)	Compression (%)	Top-1 (%)	Compression (%)
Baseline	65.46	0.00	68.12	0.00
Std. Tucker [36], [30]	57.58	60.00	59.05	60.00
PSTR-M [59]	63.68	**78.72**	66.79	**80.77**
PSTR-S [59]	66.19	56.52	68.07	58.33
BATUDE [36]	66.73	64.29	68.97	61.54
Proposed 1	65.66	21.55	68.75	42.11
Proposed 2	**67.36**	5.66	**69.96**	35.70

reducing the number of trainable parameters by 30.66% and 5.79% for ResNet-20, and by 52.78% and 24.96% for ResNet-32, in comparison to the baseline. This indicates that our proposed method configurations are more parameter-efficient relative to the baseline ResNet-20 and ResNet-32 models, even if there is a slight degradation in performance for the proposed 1 configurations. One important observation to note is that unlike other comparative methods, these efficiencies are achieved without the necessity for post-training fine-tuning/retraining, which significantly reduces post-training processing time and effort. Even though methods such as Std. Tucker [30], [36], PSTR-M [59], and BATUDE [36] display higher parameter compression rates, they come at the expense of lower accuracy scores. This implies a trade-off between model efficiency and performance. The specific compression and accuracy values of the proposed methods suggest a more balanced approach in dealing with this trade-off.

2) PERFORMANCE ANALYSIS ON CIFAR-100 DATASET

Next, we extend our experimental results to the CIFAR-100 dataset, as depicted in Table 3. The CIFAR-100 dataset, in contrast to CIFAR-10, offers fewer images per class, thus presenting a scenario for image classification in a resource-constrained environment. We do not include FLOPs comparison in the table due to lack of FLOPs information for the comparative method.

Again, two distinct configurations of our proposed method, denoted as proposed 1 and proposed 2, are presented for comparison. For the ResNet-20 models, we use $\lambda_{\text{comp}} = 0.5$ and $\epsilon = 0.01$ for proposed 1, whereas proposed 2 employs λ_{comp} and ϵ values specified in Table 1. For the ResNet-32-based models, we adopt $\lambda_{\text{comp}} = 1$ and $\epsilon = 0.1$ for proposed 1, and again, proposed 2 uses the values from Table 1.

The results demonstrate that both proposed 1 and proposed 2 configurations yield the highest Top-1 accuracy for ResNet-20, at 65.66% and 67.36%, respectively, while simultaneously achieving a parameter compression of 21.55% and 5.66%, respectively, compared to the baseline. For ResNet-32, the proposed 2 configuration gives the highest Top-1 accuracy at 69.96% while achieving a parameter compression of 35.70% in comparison to the baseline. Proposed 1

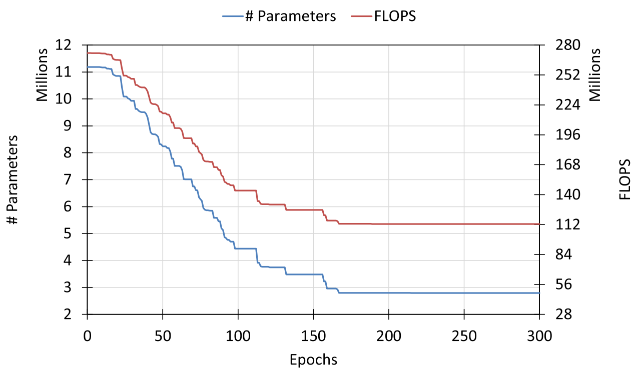


FIGURE 4. The variation in the number of trainable parameters and FLOPS across epochs for the ResNet-20 proposed model using CIFAR-10 dataset.

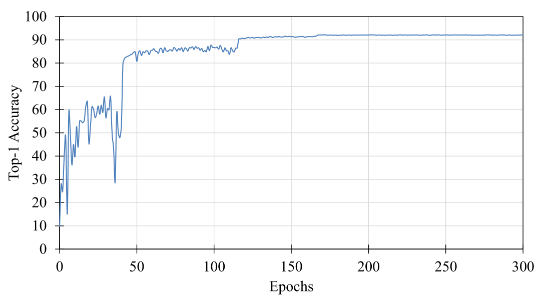


FIGURE 5. The evolution of the Top-1 accuracy on test set with training epochs for the ResNet-20 proposed model using CIFAR-10 dataset.

configuration outperforms the baseline and most of the comparative methods (with the exception of BATUDE [36]) in Top-1 accuracy at 68.75% while simultaneously reducing the number of trainable parameters by 42.11%.

Although the compression rates of the proposed configurations on CIFAR-100 are lower than some of the comparative methods, such as PSTR-M [59], its higher accuracy highlights an important trade-off between compression rates and classification performance. A higher compression rate does not always equate to better classification performance. Notably, the proposed 1 configuration achieves an improvement of 1.90% over the baseline on ResNet-20 and 1.84% on ResNet-32, while significantly reducing the number of trainable parameters (by 5.66% and 35.70% respectively compared to the baseline). Similarly to CIFAR-10, these improvements are achieved without the need for post-training fine-tuning or retraining, thus saving significant post-training processing time and effort. These results confirm the effectiveness of the proposed method for image classification tasks, especially in resource-constrained environments.

3) REDUNDANCY ANALYSIS

Fig. 4 illustrates the variations in the number of trainable parameters and MMAC over the course of training epochs for the ResNet-20 network, utilizing our proposed method on the CIFAR-10 dataset. The plot reveals an initial linear and monotonic decrease in both the number of trainable parameters and MMAC, persisting until approximately the

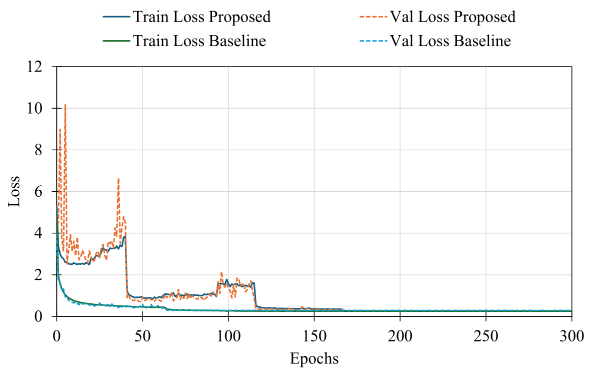


FIGURE 6. Training and validation loss curves for the proposed method and the baseline model for the ResNet-20 network using CIFAR-10 dataset. Both approaches show stable convergence without significant overfitting.

100th epoch. Subsequently, a plateau is observed, indicating convergence. This pattern suggests that the original network possessed redundant parameters that were effectively pruned by our proposed method during training. Consequently, a more efficient model was created, improving upon the original architecture, and adapting to the complexity of the data and the task at hand.

The rank reduction process varies across different layers, with each layer potentially undergoing rank reduction at different stages of training. This heterogeneity makes it challenging to directly compare accuracy against rank reduction throughout the training process. However, we provide an analysis of accuracy trends in relation to the training epochs, as shown in Fig. 5. This figure demonstrates that, despite the dynamic rank pruning process, the model's accuracy steadily improves, achieving a high level of performance early in the training process and maintaining it throughout. The combination of Fig. 4 and Fig. 5 illustrates how the dynamic rank pruning method impacts accuracy indirectly by compressing model without compromising performance, highlighting the model's ability to adapt effectively to the data and task at hand. To further ensure that these observed performance gains are not a byproduct of overfitting, we examined the training dynamics by comparing the training and validation losses over time.

In Fig. 6, we present the training and validation loss curves for both the proposed method and the baseline model. Although the proposed method initially displays a higher variance in the validation loss, it steadily converges, maintaining a close match between training and validation performance as the epochs progress. The baseline model similarly shows alignment between training and validation losses after the initial training phase. These observations suggest that neither approach suffers from significant overfitting, further reaffirming the robustness and generalization capability of our compression strategy. Having established that the model's compression-driven improvements do not lead to overfitting, we now turn our attention to understanding how these gains manifest at a more granular, layer-wise level. Specifically, we investigate the rank redundancy across

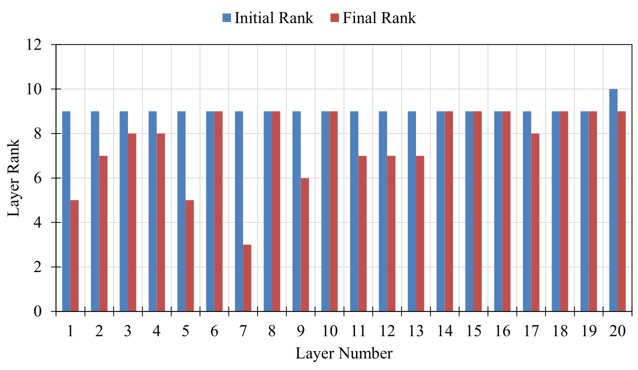


FIGURE 7. Initial and final rank comparison for the ResNet-20 proposed model using the CIFAR-10 dataset. A smaller rank indicates a more compact layer with relatively fewer trainable parameters.

TABLE 4. Comparison between regularizations using ResNet-20 baseline network on CIFAR-10 dataset. Best results for each evaluation metric are highlighted in bold text.

Method	Regularization	Top-1 (%)	Compression (%)	MMAC
Baseline		90.98	0.00	**41.01**
Factorized		**92.32**	−0.67	43.20
Factorized	L1	92.11	0.19	43.18
Factorized	L2	92.08	−0.65	43.20
Factorized	Funnel [35]	91.29	−0.62	43.20
Factorized	Proposed	92.16	**5.79**	43.04

different layers to gain insights into where and how the compression most effectively takes place.

To gauge the degree of rank redundancy across layers in the baseline network, we juxtapose (see Fig. 7) the initial and final ranks of the ResNet-20 network using our proposed method on the CIFAR-10 training dataset. The initial rank determines the learning capacity of a factorized layer. If the data and task at hand are relatively simple, a higher initial rank for a layer may lead to overfitting, indicating existing redundancy and potential for compression in the layer. Our observations uncover varying degrees of redundancy, most notably in the early to intermediate layers of the network, which generally are responsible for learning low- to intermediate-level features. For such layers, starting with a low initial rank does not deteriorate the performance and leads to a more compressed network. However, this is not true for other layers in the network that utilize full rank for feature learning.

The proposed method inherently considers the synergistic effects of compression among layers. As each layer undergoes rank reduction dynamically during training, the interdependencies between layers are implicitly accounted for. This dynamic adjustment ensures that the overall network architecture adapts cohesively, maintaining a balance between compression and performance. By allowing different layers to adjust their ranks at different stages, the method ensures that the compression in one layer synergistically complements the compression in other layers. This results in a globally optimized network that leverages layer-wise redundancy without compromising the overall learning capacity and accuracy of the model.

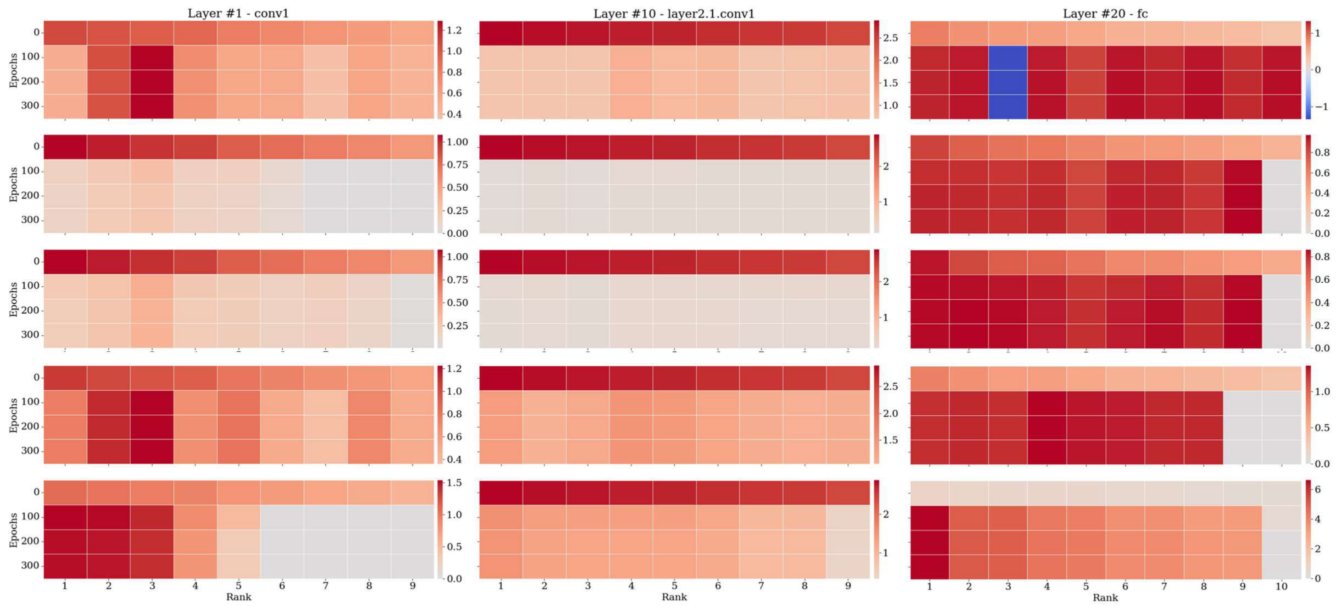


FIGURE 8. Comparison of rank variation (x-axis) across training epochs (primary y-axis) with a color bar (secondary y-axis) representing the values of singular terms examined at three distinct layers of a ResNet-20 network (from left to right: the initial (#1) layer, the intermediate (#10) layer, and the final (#20) layer) trained on the CIFAR-10 dataset under diverse regularization conditions (from top to bottom: no regularization, L1, L2, funnel, and our proposed regularization).

These insights pave the way for the design of comparatively leaner networks with fewer trainable parameters per layer.

4) ABLATION STUDY WITH DIFFERENT REGULARIZATIONS

Within this factorization framework, various regularization techniques, such as L1, L2, and funnel [35], can be employed for network compression in place of the proposed losses. Similar to (8), the general expression for the total loss is of the form

$$L_{\text{total}} = L_{\text{app}} + \lambda_{\text{reg}} L_{\text{reg}}, \tag{14}$$

where λ_{reg} is the regularization hyper-parameter. For L1 regularization,

$$L_{\text{reg}} = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{r_l} ||\sigma_l||_1. \tag{15}$$

For L2 regularization,

$$L_{\text{reg}} = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{r_l} ||\sigma_l||_2. \tag{16}$$

For funnel regularization,

$$L_{\text{reg}} = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{r_l} \sum_{i=1}^{r_l} \frac{|[\sigma_l]_i|}{|[\sigma_l]_i| + \delta}, \tag{17}$$

for some low positive value for δ. Table 4 summarizes the outcomes of an ablation study that explores the use of different regularization methods as mentioned in (15), (16), (17) to dynamically facilitate model compression during the training process. The CIFAR-10 dataset on the ResNet-20 baseline network serves as the foundation for this analysis, and each method is evaluated in terms of Top-1 classification accuracy, compression (i.e. reduction in the number of trainable parameters), and MMAC. While L1, L2, and funnel regularizations have been employed in prior research for model compression during the post-training phases [35], we have instead incorporated them into our proposed dynamic compression framework during the training process for a more equitable comparison using $\lambda_{\text{reg}} = 0.1$ and $\epsilon = 0.001$ as was done in [35]. The factorized method without any regularization is our full-rank factorized baseline model.

Our observations indicate that the factorized model, when combined with the proposed regularization, achieves an accuracy of 92.16%. This exceeds the baseline and all other regularization methods except the factorized method without any regularization. The latter, while yielding the highest accuracy of 92.32% (0.16% higher than our proposed method), does so at the expense of an increase in trainable parameters. These results suggest that our proposed regularization technique provides a competitive performance, delivering near-optimal accuracy whilst promoting model compression.

5) VISUALIZATION OF RANK VARIATION WITH DIFFERENT REGULARIZATIONS

Fig. 8 provides additional evidence substantiating our findings. This figure contrasts the rank variation (x-axis) across training epochs (primary y-axis) with a color bar (secondary y-axis) representing the intensity of singular values. We examine this at three distinct layers of the ResNet-20 network: the initial (#1) layer, the intermediate (#10) layer, and the final (#20) layer. We also study the network performance under diverse regularization conditions: no regularization,

TABLE 5. Comparison of Top-1 accuracy, Top-5 accuracy, and GMAC across different methods using a ResNet-18 network and the ImageNet dataset. Best results for each evaluation metric are highlighted in bold text. Unavailable accuracy scores are indicated by '-'.

Method	Top-1 (%)	Top-5 (%)	GMAC
Baseline	69.54	88.92	1.82
SlimNet [35], [40], [66]	67.76	87.63	1.31
LCL [67]	65.91	86.63	1.19
CP-TPM [35], [58]	67.30	–	1.15
FPGM [68]	67.62	87.83	1.06
DCP [66], [69]	67.25	87.54	0.96
SFP [70]	66.39	87.08	1.06
FBS [66]	67.04	87.47	0.92
CGNN [71]	67.91	87.89	1.13
MUSCO [35], [36], [51]	68.72	88.62	0.75
TRP [34]	65.93	86.72	0.70
DSA [65]	68.43	88.20	1.06
Stable Low-rank [57]	68.85	88.77	**0.59**
Funnel [35]	68.82	–	0.90
BATUDE [36]	–	89.25	0.72
Proposed	**70.08**	**89.62**	1.85

L1, L2, funnel, and our proposed regularization. All tests are conducted on the CIFAR-10 dataset, with configurations initialized by SVD at epoch 0.

The first row illustrates a factorized model without any regularization, which departs from the SVD condition during training and exhibits random value fluctuations across all three layers. In contrast, models implementing L1 and L2 regularizations adhere to a more rigorous protocol, suppressing all values during each parameter update in a manner that could be described as 'greedy'. Yet, both regularizations lack a focused suppression scheme beneficial for pruning.

The L1 regularization, the most stringent of all, can lead to over-pruning of trainable parameters and subsequent performance degradation. Therefore, it demands cautious selection of pruning thresholds and scaling weights. Although L2 regularization penalizes large deviations from sparsity, its failure to suppress values beyond the pruning threshold undermines its suitability for the compression process. Funnel regularization strives for rank reduction through a steep loss slope for minimal values. However, it presumptuously anticipates the presence of small singular values across all layers, thus hindering its effectiveness.

It should be noted that the unregularized factorized method as well as all the above regularizations deviate from the SVD condition, inducing the learning of correlated features and sub-optimal exploration of redundancies in trainable parameters. In contrast, our proposed regularization method actively encourages adherence to the SVD condition throughout training, exhibiting well-managed rank variations. This method concentrates these variations, prompting sparsity in the least-valued rightmost values, which are dynamically removed during the training phase itself. The focus on SVD conditions during training fosters the learning of uncorrelated parameters, which in turn allows for an optimal exploration of redundancies in trainable parameters. Notably, our proposed regularization deviates from other methods by employing a pruning threshold in relative terms rather than absolute ones. This approach promotes the removal of less significant parameters based on the relative values of singular values sorted in descending order. Consequently, pruning of such less important parameters results in little to no deterioration in performance.

6) PERFORMANCE ANALYSIS ON IMAGENET DATASET

In Table 5, we compare the proposed method with the baseline and various other methods, using the ResNet-18 network and ImageNet dataset. We specifically evaluate the Top-1 and Top-5 accuracy, and GMAC. Although our proposed method achieves compression for the ResNet-18 model, it is not presented in the table due to the lack of comparative data from other methods. Notably, our proposed method achieves the highest Top-1 and Top-5 accuracy of 70.08% and 89.62%, respectively, making it the only method to exceed the baseline performance in terms of Top-1 accuracy. Although our method does not achieve the lowest GMAC, it remains computationally similar to the baseline with a GMAC of 1.85. This indicates a strategic balance between maintaining high accuracy and managing computational complexity, a critical consideration in scenarios where accuracy cannot be compromised for computational efficiency. Our method's unique ability to dynamically determine the factorization rank per layer in an end-to-end trainable manner, based on the training dataset, significantly contributes to this balance. This feature, absent in other comparative methods, facilitates a more adaptable and efficient learning process, enhancing accuracy without excessively increasing computational demands. The GMAC value for our method is slightly higher than the baseline due to the additional operations needed to convert SVD factors to convolutional filters and dense matrix weights. However, our approach accomplishes model compression during training, thereby eliminating the need for post-training operations required by other comparative methods. This feature results in significant savings in post-training rank determination and processing times, enhancing the overall efficiency of our method despite the slight computational overhead. Additionally, a closer analysis of the table reveals the delicate balance between GMAC and Top-1 accuracy. Methods with lower GMAC, such as BATUDE [36] and funnel [35], do not necessarily guarantee superior Top-1 or Top-5 accuracy. This result highlights the effectiveness of our proposed method, which provides the highest Top-1 accuracy while maintaining a computational GMAC nearly identical to the baseline. The proposed method's performance underscores the advantage of its novel, end-to-end trainable approach and the benefits of dynamic compression during the training phase.

Our experiments on the ImageNet dataset utilized the ResNet-18 network due to computational constraints. Despite being a moderately sized network with approximately 11M trainable parameters, ResNet-18 presents a significant increase in complexity compared to ResNet-20 and ResNet-32 with approximately 270K and 460K trainable parameters, respectively. The successful application of our method to ResNet-18 demonstrates its scalability and stability in deeper architectures. The consistent improvements in accuracy and parameter reduction across different network sizes suggest that our method is not only effective for small models but also holds promise for larger networks like ResNet-201. We anticipate that the dynamic parameter rank pruning and regularization techniques would further exploit the redundancies inherent in larger models, potentially leading to even greater compression rates without sacrificing performance.

7) MEMORY USAGE ANALYSIS

Memory usage is a critical factor when deploying deep learning models, especially in resource-constrained environments. We measured the GPU memory consumption of our proposed method during both training and inference and compared it to the baseline models. For the ResNet-20 and ResNet-32 models on the CIFAR-10 and CIFAR-100 datasets, we observed that the proposed method requires slightly more GPU memory during training compared to the baseline models. Specifically, the peak memory usage increased by approximately 8% to 12%. This increase is due to the additional storage needed for the SVD factors and the computations associated with our regularization techniques. However, during inference, the memory usage of the proposed method is comparable to or slightly less than that of the baseline models. The dynamic rank pruning reduces the number of parameters, leading to a smaller model size. As shown in Tables 2 and 3, our method achieves up to 52% reduction in model parameters, which translates to decreased memory requirements during inference. For the ResNet-18 experiments on the ImageNet dataset, we observed a similar trend. The training phase showed an increase in memory usage of about 10%, while the inference phase exhibited a slight decrease in memory consumption compared to the baseline model. Overall, while our method introduces a modest increase in memory usage during training due to the overhead of factorization and regularization, it offers memory savings during inference as a result of model compression. This trade-off is acceptable, especially considering the significant reduction in model size and the improvements in accuracy.

V. DISCUSSION

Our study presents a novel dynamic CNN compression training approach, factorization reshaping, and regularization techniques that have demonstrated exceptional performance in terms of Top-1 accuracy, Top-5 accuracy, model compression, and computational complexity (MMAC, GMAC). The primary focus of the proposed regularizations is to promote SVD condition during training that ensures the learning of uncorrelated parameters. Consequently, it encourages optimal exploration of redundancies in trainable parameters and fosters better generalization. By concentrating rank variations and promoting focused sparsity, our method allows for dynamic pruning of less significant parameters during the training phase. It is distinct from traditional pruning techniques in that it uses a relative threshold based on the sorted singular values instead of an absolute threshold.

Interestingly, this approach results in minimal performance degradation, if any. An essential element of our proposed method is its ability to dynamically determine the factorization rank per layer in an end-to-end trainable manner. This ability is novel compared to other techniques and contributes to significant savings in post-training rank determination and processing times. The contrast between our method and others in terms of computational complexity provides valuable insights into the trade-off between model efficiency and accuracy. Despite not achieving the lowest computational complexity, our method ensured a near-baseline computational complexity while posting the highest Top-1 accuracy. This delicate balance is a critical factor for practical deployments where computational resources may be limited, but high accuracy is necessary.

However, we recognize the potential trade-offs in our study. The formation of factorized convolutional filters from SVD factors is the main source of additional computational complexity. This aspect warrants further investigation and exploration to reduce computational overhead.

Further, the effectiveness of our method is influenced by the choice of hyper-parameters such as λ_{comp} and ϵ. Finding these optimal set of hyper-parameters requires empirical tuning, which can be dataset and architecture specific. Developing adaptive strategies for hyper-parameter selection during training could further enhance the robustness and applicability of our method.

While our study focuses on 2D image classification, the proposed method can be extended to 3D image classification tasks without significant modification. In 3D CNNs, convolutional filters are 5D tensors that capture spatial features across three dimensions (depth, height, width) along with input and output channels. However, these filters can be reshaped into 2D matrices similarly to our approach for 2D filters. By flattening the spatial dimensions and rearranging the tensor into a matrix of size $(C_{out} \times C_{in})$ by $(D \times H \times W)$, where C_{out} and C_{in} are the output and input channels, and D, H, W are the depth, height, and width of the filter, we can apply SVD for factorization. Using our DPRP and regularization techniques, we can dynamically adjust the rank of these reshaped matrices during training, effectively compressing the model while maintaining performance. This approach allows us to apply our method directly to 3D CNNs used in tasks such as volumetric medical image analysis and video classification, without the need for higher-order tensor decompositions.

VI. CONCLUSION

In this paper, we introduced a novel training method that compresses a CNN via DPRP, utilizing an innovative reshaping technique for SVD factorization alongside our proposed regularization techniques. Our method demonstrated superior performance across several key measures such as Top-1 accuracy, Top-5 accuracy, and model compression with competitive computational complexity. The regularization techniques presented a compelling approach to model compression during training via dynamic rank pruning while maintaining high performance in classification tasks. The success of the proposed approach lies in its focus on promoting the SVD condition during training, which facilitates the learning of uncorrelated parameters and dynamic pruning of less significant parameters. Our findings underscore the importance of carefully balancing model accuracy, network compression, and computational complexity. Even though achieving the lowest computational complexity is a common objective, our research highlighted the crucial nature of preserving or even improving model accuracy amidst network compression for real-world applications.

Looking forward, there are several avenues to expand our research. Exploring the applicability and performance of our method with different types of neural network architectures, such as transformers or recurrent networks, as well as tasks beyond image classification, like object detection and image segmentation, is a promising direction. Further investigation into determining different hyper-parameters dynamically during training could potentially enhance our technique accuracy and compression further. These exciting prospects suggest that our work lays a firm foundation for future research on model compression via dynamic rank determination.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NGA, DoD, or the U.S. Government.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[4] M. Dhanaraj, M. Sharma, T. Sarkar, S. Karnam, D. G. Chachlakis, R. Ptucha, P. P. Markopoulos, and E. Saber, "Vehicle detection from multi-modal aerial imagery using YOLOv3 with mid-level fusion," *Proc. SPIE*, vol. 11395, pp. 22–32, May 2020.

[5] M. Sharma, M. Dhanaraj, S. Karnam, D. G. Chachlakis, R. Ptucha, P. P. Markopoulos, and E. Saber, "YOLOrs: Object detection in multimodal remote sensing imagery," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 1497–1508, 2020.

[6] M. Sharma, P. P. Markopoulos, and E. Saber, "YOLOrs-lite: A lightweight CNN for real-time object detection in remote-sensing," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. IGARSS*, Jul. 2021, pp. 2604–2607.

[7] S. Singh, M. Sharma, J. Heard, J. D. Lew, E. Saber, and P. P. Markopoulos, "Multimodal aerial view object classification with disjoint unimodal feature extraction and fully-connected-layer fusion," *Proc. SPIE*, vol. 12522, Jun. 2023, Art. no. 1252206.

[8] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.

[9] L.-H. Wen and K.-H. Jo, "Deep learning-based perception systems for autonomous driving: A comprehensive survey," *Neurocomputing*, vol. 489, pp. 255–270, Jun. 2022.

[10] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 1–9.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[12] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–13.

[13] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 4133–4141.

[14] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9163–9171.

[15] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3967–3976.

[16] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014, *arXiv:1412.6115*.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*.

[19] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.

[20] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[21] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–17.

[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–6. [Online]. Available: https://openreview.net/forum?id=rJqFGTslg

[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016, *arXiv:1602.07360*.

[24] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[26] M. Sharma, P. P. Markopoulos, E. Saber, M. S. Asif, andA. Prater-Bennette, "Convolutional auto-encoder with tensor-train factorization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 198–206.

[27] R. Hyder, K. Shao, B. Hou, P. Markopoulos, A. Prater-Bennette, and M. S. Asif, "Incremental task learning with incremental rank updates," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland. Springer, 2022, pp. 566–582.

[28] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.

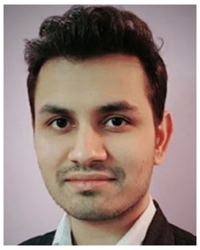
[29] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," 2014, *arXiv:1412.6553*.

[30] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," 2015, *arXiv:1511.06530*.

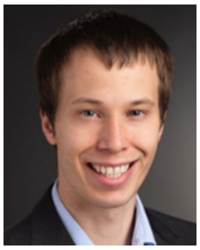
[31] T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov, "Ultimate tensorization: Compressing convolutional and FC layers alike," 2016, *arXiv:1611.03214*.

[32] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6655–6659.

[33] H. Chung, E. Chung, J. G. Park, and H.-Y. Jung, ''Parameter reduction for deep neural network based acoustic models using sparsity regularized factorization neurons,'' in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2019, pp. 1–5.
[34] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, Y. Qi, Y. Chen, W. Lin, and H. Xiong, ''TRP: Trained rank pruning for efficient deep neural networks,'' in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 977–983.
[35] B.-S. Chu and C.-R. Lee, ''Low-rank tensor decomposition for compression of convolutional neural networks using funnel regularization,'' 2021, *arXiv:2112.03690*.
[36] M. Yin, H. Phan, X. Zang, S. Liao, and B. Yuan, ''BATUDE: Budget-aware neural network compression based on tucker decomposition,'' in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 8, pp. 8874–8882.
[37] M. Kirtas, A. Oikonomou, N. Passalis, G. Mourgias-Alexandris, M. Moralis-Pegios, N. Pleros, and A. Tefas, ''Quantization-aware training for low precision photonic neural networks,'' *Neural Netw.*, vol. 155, pp. 561–573, Nov. 2022.
[38] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, ''Quantization and training of neural networks for efficient integer-arithmetic-only inference,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[39] M. Kirtas, N. Passalis, A. Oikonomou, M. Moralis-Pegios, G. Giamougiannis, A. Tsakyridis, G. Mourgias-Alexandris, N. Pleros, and A. Tefas, ''Mixed-precision quantization-aware training for photonic neural networks,'' *Neural Comput. Appl.*, vol. 35, no. 29, pp. 21361–21379, Oct. 2023.
[40] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, ''Learning efficient convolutional networks through network slimming,'' in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2736–2744.
[41] Z. Yang, Y. Zhai, Y. Xiang, J. Wu, J. Shi, and Y. Wu, ''Data-aware adaptive pruning model compression algorithm based on a group attention mechanism and reinforcement learning,'' *IEEE Access*, vol. 10, pp. 82396–82406, 2022.
[42] A. Amelio, G. Bonifazi, F. Cauteruccio, E. Corradini, M. Marchetti, D. Ursino, and L. Virgili, ''Representation and compression of residual neural networks through a multilayer network based approach,'' *Expert Syst. Appl.*, vol. 215, Apr. 2023, Art. no. 119391.
[43] D. Tian, S. Yamagiwa, and K. Wada, ''Heuristic compression method for CNN model applying quantization to a combination of structured and unstructured pruning techniques,'' *IEEE Access*, vol. 12, pp. 66680–66689, 2024.
[44] Y. Guo, A. Yao, and Y. Chen, ''Dynamic network surgery for efficient DNNs,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.
[45] Z. Huang and N. Wang, ''Data-driven sparse structure selection for deep neural networks,'' in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 304–320.
[46] P. Bian, Z. Zheng, and D. Zhang, ''Light-weight multi-channel aggregation network for image super-resolution,'' in *Proc. Chinese Conf. Pattern Recognit. Comput. Vis. (PRCV)*, Beijing, China. Cham, Switzerland: Springer, Nov. 2021, pp. 287–297.
[47] J. Kossaifi, A. Khanna, Z. Lipton, T. Furlanello, and A. Anandkumar, ''Tensor contraction layers for parsimonious deep nets,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 26–32.
[48] D. T. Tran, A. Iosifidis, and M. Gabbouj, ''Improving efficiency in convolutional neural networks with multilinear filters,'' *Neural Netw.*, vol. 105, pp. 328–339, Sep. 2018.
[49] J. Kossaifi, A. Bulat, G. Tzimiropoulos, and M. Pantic, ''T-net: Parametrizing fully convolutional nets with a single high-order tensor,'' in *Proc. IEEE/CVF Conf. Comput. Vis. pattern Recognit.*, Jun. 2019, pp. 7822–7831.
[50] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, T. Patti, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, ''Tensor methods in deep learning,'' in *Signal Processing and Machine Learning Theory*. Amsterdam, The Netherlands: Elsevier, 2024, ch. 15, pp. 1009–1048.
[51] J. Gusak, M. Kholiavchenko, E. Ponomarev, L. Markeeva, P. Blagoveschensky, A. Cichocki, and I. Oseledets, ''Automated multi-stage compression of neural networks,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, Oct. 2019, pp. 1–8.
[52] H. Yang, M. Tang, W. Wen, F. Yan, D. Hu, A. Li, H. Li, and Y. Chen, ''Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 678–679.
[53] J. Frankle and M. Carbin, ''The lottery ticket hypothesis: Finding sparse, trainable neural networks,'' in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–42.
[54] M. Jaderberg, A. Vedaldi, and A. Zisserman, ''Speeding up convolutional neural networks with low rank expansions,'' 2014, *arXiv:1405.3866*.
[55] X. Zhang, J. Zou, K. He, and J. Sun, ''Accelerating very deep convolutional networks for classification and detection,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, Oct. 2015.
[56] D. Savostianova, E. Zangrando, G. Ceruti, and F. Tudisco, ''Robust low-rank training via approximate orthonormal constraints,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–20.
[57] A.-H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavskỳ, V. Glukhov, I. Oseledets, and A. Cichocki, ''Stable low-rank tensor decomposition for compression of convolutional neural network,'' in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, 2020, pp. 522–539.
[58] M. Astrid and S.-I. Lee, ''CP-decomposition with tensor power method for convolutional neural networks compression,'' in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 115–118.
[59] N. Li, Y. Pan, Y. Chen, Z. Ding, D. Zhao, and Z. Xu, ''Heuristic rank selection with progressively searching tensor ring network,'' *Complex Intell. Syst.*, vol. 8, no. 2, pp. 1–15, Apr. 2021.
[60] E. Zangrando, S. Schotthöfer, G. Ceruti, J. Kusch, and F. Tudisco, ''Rank-adaptive spectral pruning of convolutional layers during training,'' 2023, *arXiv:2305.19059*.
[61] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2013.
[62] A. Krizhevsky and G. Hinton, ''Learning multiple layers of features from tiny images,'' Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[63] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, ''ImageNet large scale visual recognition challenge,'' *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.
[65] X. Ning, T. Zhao, W. Li, P. Lei, Y. Wang, and H. Yang, ''DSA: More efficient budgeted pruning via differentiable sparsity allocation,'' in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland. Springer, Jan. 2020, pp. 592–607.
[66] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C.-z. Xu, ''Dynamic channel pruning: Feature boosting and suppression,'' in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.
[67] X. Dong, J. Huang, Y. Yang, and S. Yan, ''More is less: A more complicated network with less inference complexity,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 5840–5848.
[68] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, ''Filter pruning via geometric median for deep convolutional neural networks acceleration,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4340–4349.
[69] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, ''Discrimination-aware channel pruning for deep neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, Dec. 2018, pp. 1–12.
[70] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, ''Soft filter pruning for accelerating deep convolutional neural networks,'' in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2234–2240.
[71] W. Hua, Y. Zhou, C. M. De Sa, Z. Zhang, and G. E. Suh, ''Channel gating neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

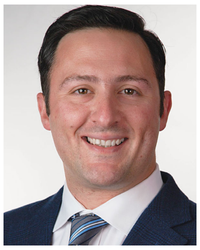


MANISH SHARMA (Student Member, IEEE) was born in Kaithal, Haryana, India, in 1994. He received the B.Sc. degree in physics from the University of Delhi, New Delhi, India, in 2014, and the M.Sc. degree in physics from Indian Institute of Technology Delhi, New Delhi, in 2016. He is currently pursuing the Ph.D. degree in imaging science with the Rochester Institute of Technology, Rochester, NY, USA. His research interests include computer vision, deep learning, tensor processing for neural networks, and multimodal data fusion.



JAMISON HEARD (Member, IEEE) received the Bachelor of Science degree in electrical engineering from the University of Evansville, Evansville, IN, USA, in 2013, and the Master of Science and Ph.D. degrees in electrical engineering from Vanderbilt University, Nashville, TN, USA, in 2016 and 2019, respectively. He is currently researching adaptive human–robot teams, human-aware reinforcement learning, human–robotic interaction, task recognition, and real-time human state assessment.

ELI SABER (Senior Member, IEEE) received the B.S. degree in electrical and computer engineering (ECE) from the University of Buffalo, in 1988, and the M.S. and Ph.D. degrees in ECE from the University of Rochester, in 1992 and 1996, respectively. From 1997 to until 2004, he was an Adjunct Faculty Member with the EME Department, RIT, and the ECE Department, University of Rochester. He is currently a Professor with the Electrical and Microelectronic Engineering (EME) Department, Kate Gleason College of Engineering, and an Extended Faculty Member with the Center for Imaging Science, College of Science, Rochester Institute of Technology. Prior to that, he was with Xerox Corporation, from 1988 to until 2004, in a variety of positions ending as a Product Development Scientist and the Manager of the Business Group Operations Unit. He holds over 100 peer-reviewed conference, journal and patent publications and is the co-author of a textbook titled "Advanced LINEAR Algebra for Engineers with MATLAB." He has also co-edited a special issue for *IEEE Signal Processing Magazine* on color imaging and *Journal of Electronic Imaging* on video surveillance/transportation imaging. His research interests include machine learning, digital image and video processing, remote sensing, and computer vision. He is a Senior Member of the Institute for Electrical and Electronic Engineers, a Former Member of the Image and Multi-Dimensional Digital Signal Processing Technical Committee, and a Former Member and the Past Chair of the IEEE Industry Technology Committee. He has served as an Associate Editor for IEEE Transaction on Image Processing and an Area Editor for *Journal of Electronic Imaging*. He has also served in various roles on several conference committees, including ICIP 2002, ICIP 2007, ICIP 2009, ICIP 2012, ICASSP 2017, and ICIP 2021.



PANAGIOTIS (PANOS P.) MARKOPOULOS (Senior Member, IEEE) was born in Athens, Greece, in 1986. He received the Diploma and M.S. degrees in electronic and computer engineering from the Technical University of Crete, Chania, Greece, in 2010 and 2012, respectively, and the Ph.D. degree in electrical engineering from The State University of New York at Buffalo, Buffalo, NY, USA, in 2015. He is currently an Associate Professor and Margie and a Bill Klesse Endowed Professor with the Department of Electrical and Computer Engineering and the Department of Computer Science, The University of Texas at San Antonio (UTSA), San Antonio, TX, USA. He is also the Founding Director of the UTSA Machine Learning Optimization and Signal Processing Laboratory (MELOS), a Core Faculty Member with the UTSA School of Data Science, and a Core Faculty Member with the MATRIX: The UTSA AI Consortium for Human Well-Being. Prior to joining UTSA, he was a tenured Associate Professor with the Rochester Institute of Technology (RIT), Rochester, NY, USA. In 2018, 2020, and 2021, he was a Summer Visiting Research Faculty with the U.S. Air Force Research Laboratory, Information Directorate, Rome, NY, USA. He has co-authored more than 70 journal and conference articles and three book chapters in his research interests, which include signal processing, machine learning, data analysis, numerical optimization, and artificial intelligence. He has been the Principal Investigator of multiple research projects funded by the U.S. National Science Foundation and the U.S. Department of Defense. He is a Senior Member of the IEEE Signal Processing, Computer, Communications, and Computational Intelligence Societies, and a member of the IEEE Signal Processing Society Education Board. In 2019, he was a recipient of the prestigious AFOSR Young Investigator Award. He is the Chair of the Content Production Committee. He has participated in the organization of multiple IEEE conferences, workshops, and special sessions.