Low-Rank Compression of Neural Network Weights by Null-Space Encouragement

Arda Ozdemir, Ozgur Soysal, Ege Doganay, Yigit Yildirim and Orhan Arikan
Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey
E-mail: arda.ozdemir@ug.bilkent.edu.tr, ozgur.soysal@ug.bilkent.edu.tr, ege.doganay@ug.bilkent.edu.tr,
yigit.yildirim@bilkent.edu.tr, oarikan@ee.bilkent.edu.tr

***Abstract*—Deep neural networks are often heavily over-parametrized for their purposes, making them costly to store and run on resource-constrained devices. Therefore, compressing them into fewer parameters with minimal performance loss is essential. One popular way is the low-rank decomposition because high-dimensional data often lie near a lower-dimensional manifold, and the most important features can be captured by the dominant singular values of the weight matrices. In this work, we first compress pretrained models into their low-rank forms (SVD or CP). The compressed low-rank network becomes the student and the original over-parametrized network becomes the teacher. Next, we exploit the low-rank structure of the approximated weight matrices by projecting the mismatch of intermediate activations onto the null space of the low-rank weight matrices, ensuring that the student focuses on the teacher's task-relevant directions. Finally, we use knowledge distillation to align outputs of the student and the teacher. We validate our approach on tasks involving fully connected layers and/or convolutional layers, achieving up to 90% parameter and 80% FLOP reduction, with either $< 1\%$ accuracy drop or up to 1.68% gain on MNIST, CIFAR-10, and CIFAR-100 benchmarks. We also compare our method with previously proposed FitNets [1] and Gramian [2] to show that the proposed technique is better at maintaining higher accuracy. These results indicate that the proposed technique not only reduces model size but also acts as a regularizer, improving generalization.**

I. Introduction

Deep neural networks have shown revolutionary results in various domains from image recognition to natural language processing, emerging from the success of [3] and later of [4]. Despite their expressive power, modern architectures are often heavily over-parameterized, leading to large memory footprints and high computational costs that impede deployment on mobile and embedded devices.

To address this challenge, a variety of model compression techniques have been proposed, including pruning and quantization [5], knowledge distillation [6], and exploiting low-rank structures in weight tensors. However, existing low-rank approaches suffer from performance losses as they aim to mimic the teacher one-to-one. In this work, we introduce a unified framework that combines low-rank factorization with a novel *null-space projection* loss and standard distillation to produce compact yet high-performing students. Our contributions are threefold:

1) We propose to initialize student layers via truncated SVD by using approximate methods like [7] for fully-connected layers and CP decomposition for convolutional kernels, immediately reducing parameters by about 90%.
2) We propose to employ *null-space absorbing loss* in the supervised fine-tuning stage of the compression to encourage the student to conduct parameter updates such that the difference between resulting hidden activations of the student and the original activations of the teacher are kept within the kernels of the teacher weights. This ensures only task-relevant directions are transferred to the student rather than point-to-point matching.
3) We aim for accuracy-focused compression rather than merely parameter reduction and incorporate knowledge distillation to fine-tune the student, enabling about 80% FLOP reduction keeping about the same performance on MNIST, CIFAR-10 and CIFAR-100.

The remainder of the paper is organized as follows. Section II reviews related work on neural network compression. Section III details our proposed framework. Section IV presents our experimental results, and Section V concludes with future directions.

II. Related Work

In [6], *knowledge distillation* is introduced by training a compact student network from a larger teacher network. Specifically, the student minimizes the Kullback-Leibler (KL) divergence between its softened output probability distributions and the teacher's, which is controlled with a temperature hyperparameter. This foundational technique proved that compact models can approximate larger models with minimal performance loss, paving the way for more advanced distillation and compression methods such as in [8] where the Wasserstein distance (WD) based distillation is introduced. Their work remodels the one-to-one class matching of KL divergence by an optimal transport problem. In this work, we focus on distilling low-rank students from larger teachers.

In [1], the authors address the problem of training students that are thinner and deeper than the teachers by introducing hidden-layer *hints*. They aim to optimize the training of the student by guiding it towards intermediate representations. First, activation of a teacher network layer and the corresponding student layer are selected. Then, the student is trained up to that layer to minimize the ℓ_2 distance between the activations with a small regressor to align their dimensions. Next, the student weights are jointly trained with the knowledge distillation and the classification losses. In [2], this hidden layer guidance is performed by matching cross-layer Gramian matrices, extracting feature correlations across different layers, and combining with KL divergence-based distillation. Like [1],

979-8-3315-7206-8/25/$31.00

[2], we leverage hidden features, but we project activation mismatches into the teacher's discarded subspace to focus on learning task-related directions.

Many works take advantage of the low-rank nature of weight tensors, exploiting the manifold hypothesis, to compress large networks. [9] initializes a low-rank student by reshaping convolutional kernels into matrices and applying truncated SVD to them, and fully connected layers. Likewise, [10], uses singular values and right singular vectors from convolutional layers to serve as hints to align the teacher and the student in principal directions. In [11], CP decomposition is used to approximate convolutional layers, before a fine-tuning phase. In [12], layer tensors are unfolded along input/output channels, their ranks are estimated via global analytic VBMF to capture significant features, and those ranks then guide Tucker-2 decomposition, whose low-rank factors replace the originals for training. [13] adds a structured-sparse component to each low-rank weight tensor by solving a convex optimization problem that minimizes post-ReLU reconstruction error. These studies show that low-rank approximations yield significant reductions in floating point operations (FLOPs) and parameters. In this work, we exploit the nature of these low-rank approximations. Similar to [9], a low-rank student is initialized via decompositions of the layers of the teacher, then the student is fine-tuned in such a manner that the hidden activations do not exactly match the teacher's per layer, but the difference between two activations is in the null space of that layer.

III. Null Space Absorbing Network (NSA-Net)

In this section, proposed NSA-Net distillation is described with its mathematical background. The main idea of NSA-Net is to fine-tune the decomposition matrices such that any change made to an activation is *absorbed* in the null space of the next layer. To initialize the student, NSA-Net employs low-rank decompositions. Yet, initializing the student layers randomly would then cause the resulting layers to be full-rank due to the Rank-Nullity theorem [14]; thus reducing possible compression. To circumvent this, we further propose to initialize the student weights as low-rank decompositions of the teacher's weights. We then supplement these low-rank decompositions with a fine-tuning phase.

A. Student Initialization

Layers of a pretrained teacher model are approximated with fewer parameters by applying truncated SVD to fully connected layers and CP decomposition to convolutional layers. Due to the complexity of SVD, iterative methods such as [7] can be employed. This approximation is expected to cause a performance decrease. The decomposition rank needs to be chosen carefully. Depending on this choice, the student can catch and even surpass the teacher's performance with a fine-tuning phase. Below, details of these decompositions are given. See Section III-B for details on the said fine-tuning phase.

For fully connected layers, SVD choice is based on the widely cited manifold hypothesis, which argues that a set of high-dimensional data usually lies on a lower-dimensional manifold. For neural networks, this implies weights admit low-rank structures. The truncated SVD choice is then justified by the Eckart-Young theorem. Truncated SVD of a matrix $A \in \mathbb{R}^{m\times n}$, denoted by A_r $r < \min(m,n)$, is given by

$$A_r = U_r \Sigma_r V_r^\top \text{ with } A = U\,\Sigma\,V^\top \ , \tag{1}$$

$$U_r = [u_1, \cdots, u_r], \Sigma_r = \text{diag}(\sigma_1, \cdots, \sigma_r), V_r = [v_1, \cdots, v_r] \ . \tag{2}$$

SVD fails to describe N-way tensors accurately. Thus, we resort to CP decomposition for convolutional layers. In a CNN, filter banks are viewed as 4th-order tensors as $\mathcal{X} \in \mathbb{R}^{C_{in}\times C_{out}\times k_1\times k_2}$ with C_{in} input channels, C_{out} output channels, and spatial filter size of $k_1 \times k_2$. The filters are 3D tensors in the form $\mathrm{F} \in \mathbb{R}^{C_{in}\times k_1\times k_2}$ that maps C_{in} input channels into one output channel. Two methods stand out for tensor decomposition named CP decomposition [15], [16] and Tucker decomposition [17]. In short, both decompose tensors into rank-1 terms, vectors, CP results in vectors of identical length while Tucker allows the freedom of choosing separate sizes for the terms. For simplicity and practicality, CP decomposition is utilized in this work. CP decomposition is essentially the combination of CANDECOMP by [15] and PARAFAC by [16]. The algorithm in 4-way tensors as $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times I_3\times I_4}$ can be summarized as follows.

a) Initialization: First choosing a factor R, unfolding the tensor in one mode such as $X_1 \in \mathbb{R}^{I_1\times I_2 I_3 I_4}$ and initializing four matrices denoted by $A_i \in \mathbb{R}^{I_i\times R}$, $i = 1,2,3,4$. The Gramian matrices are formed as follows.

$$G_i = (A_j^\top A_j) \circ (A_k^\top A_k) \circ (A_l^\top A_l) \ , \tag{3}$$

where $\circ$ denotes element-wise multiplication (Hadamard product) and $j = 1,2,3,4 \neq i$. Another matrix, K, is formed as follows.

$$K_i = A_j \odot A_k \odot A_l \ , \tag{4}$$

where $\odot$ denotes column-wise Kronecker product.

b) ALS Updates: The original matrix A_i is updated as follows.

$$A_i \leftarrow X_i K_i (G_i)^\dagger \ , \tag{5}$$

where $\dagger$ denotes pseudo-inverse. Let a_r^i denote the columns of the matrix A_i, $r = 1,2,\ldots,R$. The column vectors are updated to maintain numerical stability as such.

$$a_r^i \leftarrow \frac{a_r^i}{\lambda_r}, \ \lambda_r = ||a_r^i||_2 \ . \tag{6}$$

c) Termination: The procedure is repeated for all i. The new tensor $\hat{\mathcal{X}}$ is computed as follows.

$$\hat{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r a_r^1 \otimes a_r^2 \otimes a_r^3 \otimes a_r^4 \ , \tag{7}$$

where $\otimes$ denotes tensor product. The process is repeated until a specified limit of convergence is met as $\left|\left|\mathcal{X} - \hat{\mathcal{X}}\right|\right|_F < \epsilon$ or the maximum number of iterations is reached. The procedure is also known as alternating least squares (ALS) method. The decomposition will be in the center of compressing convolutional

layers because at the end of the algorithm, the parameter number is reduced from $C_{in}C_{out}k_1k_2$ to $R(C_{in}+C_{out}+k_1+k_2)$, yielding a compression ratio in the control of the user.

After the teacher is fully trained, its layers are approximated with fewer parameters by applying the above decompositions. Then, the student is fine-tuned so that its performance is brought to the level of the teacher.

B. Supervised Fine-tuning Phase

To alleviate performance loss caused by approximate decompositions, a fine-tuning phase is necessary. To this end, NSA-Net adopts *null space loss* so that changes to activations of the student are absorbed in the null space, preserving the input-output function of the teacher.

The core of our model depends on absorbing hidden activation differences in the null-spaces (kernels) of the corresponding layer. For any linear transformation $T : \mathbb{R}^n \to \mathbb{R}^m$, its kernel is,

$$\ker(T) = \{x \in \mathbb{R}^n \big| Tx = 0\} \ . \tag{8}$$

Dually, the orthogonal complement of the null-space is the row-space of T. Every vector $x \in \mathbb{R}^n$ has a unique orthogonal decomposition in the form of

$$x = x_{\|} + x_{\perp} \text{ where } x_{\|} \in \text{Row}(T), x_{\perp} \in \ker(T) \ . \tag{9}$$

Then, the output of that layer, $y \in \mathbb{R}^m$ is constructed solely from $x_{\|}$. Namely,

$$y = Tx = T(x_{\|} + x_{\perp}) = Tx_{\|} \ . \tag{10}$$

In the perspective of neural networks, the kernel component of the input has no effect on the transformation hence can be regarded as irrelevant information with no effect on the flow of the network. Using a similar idea to Fitnets ([1]), the student network is guided in a way in which, rather than matching hidden activations point-wise, we encourage the student's difference from the teacher to be absorbed in the null-spaces. In that way, students would only learn what would affect their final output. Since the student weights are low-rank forms of the teacher that are factored as $\hat{W}_i = U_i V_i^{\top}$, where $\hat{W}_i \in \mathbb{R}^{m\times n}$, $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ and $r < \min(m, n)$. The dimension of the kernel of any student layer will be larger, $\dim(\ker(\hat{W}_i)) = \min(m, n) - r$. Besides transferring only the important features from the teacher, penalizing the null component will be much more flexible for the training phase compared to point-wise matching hidden outputs. The following loss term is utilized for this purpose

$$\mathcal{L}_{\text{null}} = \sum_i \left|\left|\hat{W}_i(h_i - \hat{h_i})\right|\right|_2^2 \ . \tag{11}$$

where the summation is over all applicable student layers. In this way, the student will be guided to learn the semantic information from the hidden features of the student; only caring about features that affect the final output, i.e., those that reside in the row-space of the transformation and absorbing others. Letting $e_i \coloneqq (h_i - \hat{h}_i)$, we have

$$\mathcal{L}_{\text{null}} = \left|\left|\hat{W}_i e_i\right|\right|_2^2 = \left|\left|\hat{W}_i(e_i^{\|} + e_i^{\perp})\right|\right|_2^2 = \left|\left|\hat{W}_i e_i^{\|}\right|\right|_2^2 \tag{12}$$
$$\implies e_i^{\|} \to 0 \ ,$$

implying the parallel component is driven to 0. Thus, the loss flattens the teacher–student error onto the normal bundle, forcing all discrepancies to lie perpendicular to the row-space manifold. Note that the loss term is applied to both fully connected and convolutional layers.

Due to the noisy nature of gradient updates, the orthonormality of the left matrices will be altered. A regularization term is added to the loss to preserve orthonormality:

$$\mathcal{L}_{\text{orth}} = \left|\left|U_i^{\top} U_i - I\right|\right|_F^2 \ , \tag{13}$$

as $U^T U = U$ if and only if U is orthonormal. $\hat{W}_i$ can be represented with an infinite number of (U_i, V_i) pairs and (13) chooses the one with orthonormal U_i. It is also possible to use explicit normalization, such as QR factorization after each gradient update to guarantee orthonormality but slow down training for large models. Orthonormality will ensure a numerically stable row basis in U_i with all singular values equal to 1. Training and convergence will be faster because the gradients that flow through will not be amplified or attenuated.

The last term for the fine-tuning phase is knowledge distillation loss.

$$\mathcal{L}_{KD} = d(y, \hat{y}) \ , \tag{14}$$

where y and $\hat{y}$ are the outputs of the final linear layer of the teacher and student, respectively, $d(.,.)$ denotes a general metric. The possibilities are KL divergence or WD [6], [8]. KL divergence has proven to be simple and effective in many distillation scenarios. Despite its success, it can cause gradients to explode if the teacher distribution is too concentrated. Alternatively, WD is bounded and guarantees a smooth behavior for gradients. Unlike KL divergence, WD respects geometry where if neighboring classes have intrinsic similarities (e.g., class 1 is more alike 2 than class 5), it penalizes farther mismatches more but, not every task has that property. The 1-Wasserstein distance metric is shown below.

$$W(X_1, X_2) = \sum_x \left|F_{X_1}(x) - F_{X_2}(x)\right| \ , \tag{15}$$

where X_1, X_2 are distributions and $F_{X_1}(x), F_{X_2}(x)$ are their CDFs.

All these components are combined into one augmented fine-tuning loss $\mathcal{L}$ in the form of,

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \alpha\mathcal{L}_{\text{null}} + \beta\mathcal{L}_{\text{KD}} + \lambda\mathcal{L}_{\text{orth}} \ . \tag{16}$$

$\mathcal{L}_{\text{sup}}$ is a generalized supervised loss term that can change from task to task (e.g., cross-entropy in classification, MSE in regression, etc.), enabling the student generalize better than an over-parametrized teacher. $d(\cdot, \cdot)$ is a generalized knowledge distillation loss function between output logits, which is KL divergence or Wasserstein distance in our case, encouraging the student to learn the teacher's mapping. Low-rank initialization allows the student to start from a well-performing point. Thus, the fine-tuning process utilizes a small adaptive learning rate that decays after several epochs, leading to convergence. The updated parameters of the model are the U_i, V_i and the CP

decomposed 1-D convolutional layer vectors. Consider a linear layer of the teacher model as follows.

$$h = \sigma(Wx + b) \ . \tag{17}$$

Where σ is an arbitrary activation function. For the student model, a forward pass is given by,

$$V^\top x = z \Rightarrow Uz + b = \alpha \Rightarrow h = \sigma(\alpha) \ . \tag{18}$$

Hence, the updates of the student are given as;

$$U \leftarrow U - \eta \frac{\partial \mathcal{L}}{\partial U}, V \leftarrow V - \eta \frac{\partial \mathcal{L}}{\partial V}, b \leftarrow b - \eta \frac{\partial \mathcal{L}}{\partial b} \ , \tag{19}$$

with the following gradients derived by backpropagating $\mathcal{L}$ through U and V:

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \frac{\partial \mathcal{L}}{\partial h} \circ \sigma'(\alpha) \ , \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial U} = \frac{\partial \mathcal{L}}{\partial \alpha} z^\top + \frac{\partial \mathcal{L}_{orth}}{\partial U} \ , \tag{21}$$

$$\frac{\partial \mathcal{L}}{\partial V} = x(U^\top \frac{\partial \mathcal{L}}{\partial \alpha})^\top \ , \tag{22}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial \alpha} \ . \tag{23}$$

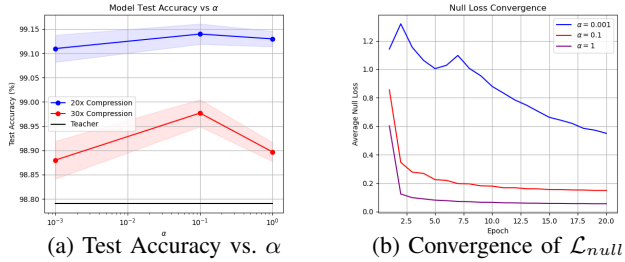
A similar derivation can be done for the convolutional kernels as well.

IV. Experiments

We evaluate NSA-Net across 3 benchmarks of increasing complexity. First, we replicate the LeNet-5 architecture on the MNIST digits dataset [3]. Next, we adopt a lightweight VGG-style network with stacked 3×3 convolutions on CIFAR-10 [18], trained for 10 epochs with *Adam* and 10^{-3} learning rate. Third, we test *AlexNet* on CIFAR-100 [4], trained for 50 epochs with *SGD*, 10^{-2} learning rate, 5×10^{-4} weight decay and 0.9 momentum. In each case, the original network serves as the teacher; we initialize a corresponding low-rank student via our factorization scheme and train it with (16). 4 teachers are used for distilling 5 students each for MNIST and CIFAR-10, 2 teachers and 3 students each are trained for *AlexNet* [4]. We sweep the student's rank to obtain compression factors ranging from 10× to 30×. We let NSA$_x$-Net denote the resulting student with rank x. We also test cascade type distillation, in which we first distill a rank x student from the teacher, then the rank x student becomes the teacher for a rank $y < x$ student. This method is denoted as NSA$_{x,y}$-Net. The training is carried out using batch sizes of 64 with *Adam* optimizer with an adaptive step learning rate 0.001 that has a step size of 2 and decay rate 0.7. The trained students are evaluated on classification accuracy, number of parameters, and FLOPs on each task. Every model is trained for the same number of epochs.

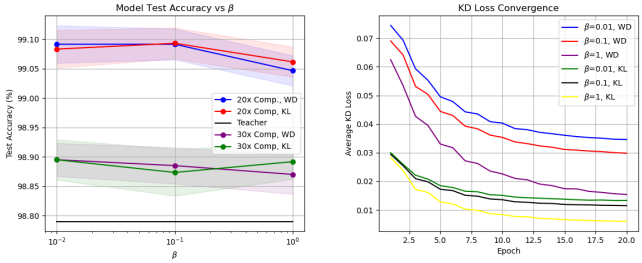
A. Hyperparameter Sensitivity

We begin with a hyperparameter search for α, β, λ to study how they affect student distillation. The teacher is LeNet-5 trained on the MNIST dataset, which obtains a test accuracy of 98.79% with 431080 parameters. Students with $20\times$ and $30\times$ compression ratios relative to the teacher are averaged over 3 seeds. Fig. 1a demonstrates how α choice affects the test accuracy, while Fig. 1b shows the fine-tuning loss curve for $\mathcal{L}_{\text{null}}$. From Fig. 1a, it is possible to deduce that accuracy is robust to α for the $20\times$ compressed model. For the smaller student, *too large* or *too small* α leads to visible performance loss, where the student either focuses too much on hidden features or does not focus at all, the loss curve with $\alpha = 0.001$ validates this. Next, we focus on the $\mathcal{L}_{KD}$ term. In Fig. 2, we compare WD and KL divergence as the knowledge distillation loss term with varying β. Fig. 2a shows test accuracies and Fig. 2b shows how $\mathcal{L}_{KD}$ evolves over epochs. NSA-Net remains relatively robust to β choice, the fastest and most stable convergence is achieved when $\beta = 0.1$, where the associated loss term reduces to half in fewer epochs. KL divergence and WD do not produce much distinctive results but it is visible that the model shows a sharper logit alignment with KL divergence. However, to avoid potential instabilities of the KL divergence, we adopt WD for the following experiments. In Fig. 3, we present the results for various λ. Fig. 3a shows the student performances for $\lambda \in [10^{-2}, 10^{0}]$. We observe that accuracy remains around 99% for both tested students. The main difference is seen in Fig. 3b, which shows the model mean condition number for various λ. We observe that condition numbers are very close to 1, and increasing λ makes the gap smaller. However, we can observe from Fig. 3d that increasing λ comes with slower convergence of students. Thus, we choose $\lambda = 0.1$ for the following experiments.



(a) Test Accuracy vs. α (b) Convergence of $\mathcal{L}_{null}$

Fig. 1: α Analysis ($\beta = 0.1$, $\lambda = 0.1$)

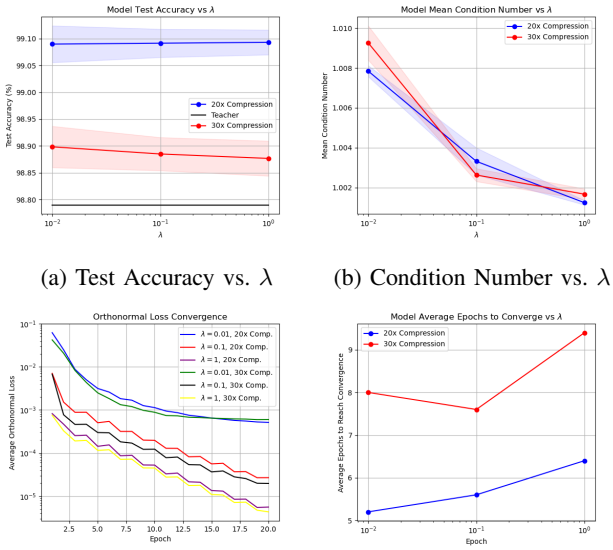


(a) Test Accuracy vs. β (b) Convergence of $\mathcal{L}_{KD}$

Fig. 2: β Analysis ($\alpha = 0.1$, $\lambda = 0.1$)

B. Accuracy Analysis

First, NSA-Net is evaluated on the MNIST [3] with the hyperparameter configuration found in the previous section. The teachers are initialized at 4 seeds, and 5 students are generated from each teacher. We compare NSA-Net with hidden-layer feature extraction-based approaches FitNets [1] and Gramian



(a) Test Accuracy vs. λ (b) Condition Number vs. λ

(c) Convergence of $\mathcal{L}_{orth}$ (d) Epochs to Converge vs. λ

Fig. 3: λ Analysis ($\alpha = 0.1$, $\beta = 0.1$)

[2]. FitNets [1] is designed for thinner and deeper students, matching chosen hidden features point-wise. Gramian-based approach [2] aligns the pairwise channel correlations between the teacher and the student feature maps via their Gramian matrices. For comparison, all student models are initialized with roughly the same parameter count. In Table I, we report the average test accuracy and 1 mean standard deviation for 4 initializations. FLOPs merge additions and multiplications into a single count.

NSA_{10}-Net surpasses the teacher by an average difference of 0.20%, while reducing FLOPs by 70% and parameters by 20-$30\times$. Cascade distillations demonstrate that an intermediate student can serve as a teacher where they perform within the margins of the directly distilled students. Overall, NSA-Net compresses over-parameterized teachers into much smaller students that match or exceed their accuracy with dramatically fewer resources. We see our method outperforms in terms of accuracy in the MNIST dataset for both model sizes. The advantages over FitNets [1] and Gramian-based method [2] are that our absorbing loss allows more flexibility than direct matching and leverages low-rank structure. One disadvantage we have is the number of FLOPs, which is due to the truncated SVD nature of fully connected layers.

Secondly, we evaluate NSA-Net on a more complex dataset, CIFAR-10 [20]. We use a VGG-style teacher with two convolutional blocks (each with two 3x3 filters), increasing feature maps from $3 \to 32 \to 64$, followed by a two-layer MLP ($4096 \to 128 \to 10$). The experiments are conducted similarly to the MNIST case, and results are reported in Table II. From Table II, we observe that the rank 10 student obtains the highest accuracy among all tested algorithms and exceeds the teacher by a margin of 0.45% on average. NSA-$_{10}$-Net achieves the highest performance in its parameter count class. If we are to distill further, we observe that FitNets [1] obtain

TABLE I: Compression vs. Accuracy on MNIST [19]

Model	Test Accuracy	FLOPs	#Parameters
Teacher	$98.85\% \pm 0.11\%$	2.292M	431080
NSA_{10}-Net	$99.05\% \pm 0.07\%$	0.705M	21500
$\text{NSA}_{25,10}$-Net	$99.07\% \pm 0.07\%$	0.705M	21500
FitNets [1]	$98.69\% \pm 0.11\%$	0.479M	21598
Gramian [2]	$98.63\% \pm 0.08\%$	0.221M	21466
NSA_{6}-Net	$98.80\% \pm 0.10\%$	0.698M	14260
$\text{NSA}_{25,6}$-Net	$98.70\% \pm 0.14\%$	0.698M	14260
FitNets [1]	$98.41\% \pm 0.13\%$	0.134M	14260
Gramian [2]	$98.57\% \pm 0.11\%$	0.214M	14970

TABLE II: Compression vs. Accuracy on CIFAR-10 [20]

Model	Test Accuracy	FLOPs	#Parameters
Teacher	$70.92\% \pm 0.35\%$	25.003M	591274
NSA_{10}-Net	$71.37\% \pm 0.23\%$	3.943M	53566
$\text{NSA}_{25,10}$-Net	$70.75\% \pm 0.16\%$	3.943M	53566
FitNets [1]	$70.74\% \pm 0.75\%$	2.616M	53576
Gramian [2]	$67.12\% \pm 1.02\%$	3.801M	53582
NSA_{6}-Net	$69.89\% \pm 0.18\%$	3.926M	36670
$\text{NSA}_{25,6}$-Net	$68.79\% \pm 0.18\%$	3.926M	36670
FitNets [1]	$70.45\% \pm 0.58\%$	2.598M	35986
Gramian [2]	$63.24\% \pm 1.63\%$	3.785M	37110

the highest accuracy while our rank 6 student is within 1 standard deviation behind. Similar to the MNIST dataset, we observe that the number of FLOPs for our algorithm remains significantly higher than other methods.

Cascade distillations incur a visible performance loss vs. direct students, -0.62% on rank-10, -1.10% on rank-6. This shows that while intermediate students can teach further compressions, each stage introduces accuracy loss. Overall, although our teacher reaches only about 71% accuracy, NSA_{10}-Net still exceeds or matches its performance, demonstrating that NSA-Net acts as a powerful regularizer, refining teacher logits, even for under-parameterized models.

Next, we evaluate NSA-Net on CIFAR-100 [20] dataset and the *AlexNet* architecture [4]. Reduced SVD is calculated and then truncated to initialize the students for computational cost. Table III shows that the NSA is ahead of the teacher model. We compare only FitNets [1] as it performed better, resulting in more FLOPs than NSA-Net, as it contains more layers. NSA-Net obtains $10.82:1$ reduction with 53.74% accuracy while FitNets obtains $9.16:1$ reduction with 53.74% accuracy.

V. Conclusions

In this work, we proposed a novel teacher-to-student distillation method to reduce the number of parameters of a network without a significant decrease in accuracy. In the proposed NSA-Net method, students are initialized as low-rank approximations of the teacher's layers, then a supervised fine-tuning phase is conducted for minimal performance loss. A novel term in the fine-tuning loss is proposed so that changes made to the student parameters are absorbed in the kernels of the layers. NSA-Net enables the student to match teacher

TABLE III: Compression vs. Accuracy on CIFAR-100 [20]

Model	Test Accuracy	FLOPs	#Parameters
Teacher	$52.06\% \pm 0.34\%$	1.13G	58.69M
NSA-Net	$53.74\% \pm 0.53\%$	0.17G	5.40M
FitNets [1]	$51.16\% \pm 1.01\%$	0.37G	6.41M

performance or even surpass it, due to supervised loss. In the MNIST, CIFAR-10, and CIFAR-100 datasets, we have demonstrated that NSA-Net achieves better accuracy with a similar number of parameters compared to FitNets [1] and Gramian [2].

We have shown that NSA-Net is successful in general convolutional architectures. Future work may explore its application to other architectures, including recurrent models, transformer-based architectures, and graph networks, as well as constructing deeper students than their teachers.

REFERENCES

[1] A. Romero, N. Ballas, S. Ebrahimi Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *International Conference on Learning Representations*, 2015.

[2] H.-H. Chou, C.-T. Chiu, and Y.-P. Liao, "Deep neural network compression with knowledge distillation using cross-layer matrix, kl divergence and offline ensemble," in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2020, pp. 71–75.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Best Paper Award, 2016. [Online]. Available: https://arxiv.org/abs/1510.00149.

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

[7] J. Baglama and L. Reichel, "Augmented implicitly restarted lanczos bidiagonalization methods," *SIAM Journal on Scientific Computing*, vol. 27, no. 1, pp. 19–42, 2005. DOI: 10.1137/04060593X. eprint: https://doi.org/10.1137/04060593X. [Online]. Available: https://doi.org/10.1137/04060593X.

[8] J. Lv, H. Yang, and P. Li, *Wasserstein distance rivals kullback-leibler divergence for knowledge distillation*, 2024. arXiv: 2412.08139 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2412.08139.

[9] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 1269–1277.

[10] S. H. Lee, D. H. Kim, and B. C. Song, "Self-supervised knowledge distillation using singular value decomposition," in *European Conference on Computer Vision*, 2018.

[11] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," in *International Conference on Learning Representations (ICLR)*, 2015.

[12] 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," in *International Conference on Learning Representations (ICLR)*, 2016.

[13] Y. Ma, R. Chen, W. Li, *et al.*, " A Unified Approximation Framework for Compressing and Accelerating Deep Neural Networks," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, pp. 376–383.

[14] S. Banerjee and A. Roy, *Linear Algebra and Matrix Analysis for Statistics* (Chapman & Hall/CRC Texts in Statistical Science), 1st. Boca Raton, FL: Taylor & Francis Group / CRC Press, 2014, p. 582, ISBN: 9781420095388. DOI: 10.1201/b17040.

[15] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart–young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970. DOI: 10.1007/BF02310791.

[16] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis," UCLA Working Papers in Phonetics, Tech. Rep. 16, Dec. 1970, pp. 1–84.

[17] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009. DOI: 10.1137/07070111X. eprint: https://doi.org/10.1137/07070111X. [Online]. Available: https://doi.org/10.1137/07070111X.

[18] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1409.1556.

[19] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.