Trained Rank Pruning for Efficient Deep Neural Networks

Yuhui Xu[1], Yuxi Li[1], Shuai Zhang[2], Wei Wen[3], Botao Wang[2], Yingyong Qi[2], Yiran Chen[3], Weiyao Lin[1] and Hongkai Xiong[1]

[1]Shanghai Jiao Tong University, Email: {yuhuixu, lyxok1, wylin, xionghongkai}@sjtu.edu.cn
[2]Qualcomm AI Research, Email: {shuazhan, botaow, yingyong}@qti.qualcomm.com
[3]Duke University, Email: {wei.wen, yiran.chen}@duke.edu

Abstract

The performance of Deep Neural Networks (DNNs) keeps elevating in recent years with increasing network depth and width. To enable DNNs on edge devices like mobile phones, researchers proposed several network compression methods including pruning, quantization and factorization. Among the factorization-based approaches, low-rank approximation has been widely adopted because of its solid theoretical rationale and efficient implementations. Several previous works attempted to directly approximate a pre-trained model by low-rank decomposition; however, small approximation errors in parameters can ripple a large prediction loss. As a result, performance usually drops significantly and a sophisticated fine-tuning is required to recover accuracy. We argue that it is not optimal to separate low-rank approximation from training. Unlike previous works, this paper integrates low rank approximation and regularization into the training. We propose Trained Rank Pruning (TRP), which iterates low rank approximation and training. TRP maintains the capacity of original network while imposes low-rank constraints during training. A stochastic sub-gradient descent optimized nuclear regularization is utilized to further encourage low rank in TRP. The TRP trained network has low-rank structure in nature, and can be approximated with negligible performance loss, eliminating fine-tuning after low rank approximation. The methods are comprehensively evaluated on CIFAR-10 and ImageNet, outperforming previous compression methods using low rank approximation.

1. Introduction

Deep Neural Networks (DNNs) have shown remarkable success in many computer vision tasks such as image classification [13], object detection [27] and semantic segmentation [4]. With the availability of large databases and high performance computation resources, deeper and wider networks [29, 13, 36] are designed to achieve better performance. Despite the high performance in server-based DNNs powered by cutting-edge parallel computing hardware, most state-of-the-art network structures are not yet ready to be deployed on mobile devices due to limitation on computation ability, memory and power.

To address this problem, many network compression and acceleration methods have been proposed. Pruning based methods [12, 14, 22, 23] explore the sparsity in weights and filters. Quantization based methods [12, 38, 6, 26, 35] reduce bit-width of network parameters. Low-rank decomposition [8, 16, 31, 9, 34, 1] minimizes the channel-wise and spatial redundancy by decomposing original network into compact ones with low-rank layers. In addition, efficient architectures [28, 24] are cleverly designed to facilitate mobile deployment of deep neural networks. This paper proposes an innovative approach to obtain low-rank networks.

Low-rank networks can be trained directly from scratch. However, it is difficult to obtain satisfactory results for the following reasons: (1) *Low capacity:* compared with an original full rank network, the capacity of a low-rank network is small, which induces difficulties on performance optimization. Efficient architectures, such as MobileNet-V2 [28] and ShuffleNet-V2 [24], need to be carefully optimized during training. (2) *Deep structure:* low-rank decomposition typically doubles the number of layers in a network. The added layers make numerical optimization much more challenging because of exploding/vanishing gradients. (3) *Rank selection:* the rank of decomposed network is often heuristically chosen based on pre-trained networks. This may not the optimized rank for network trained from scratch.

Alternatively, several previous works attempted to [37, 9, 16] decompose pre-trained models in order to get an initial low-rank network [37, 31]. However, the heuristically

imposed low-rank could cause huge accuracy loss so that re-training is needed to recover the performance of the original network. Some attempts were made to use sparsity regularization [34, 5] to constrain the network into low-rank space. Though sparsity regularization reduces the error caused by decomposition to a certain extent, performance degrades sharply when speed up rate increases.

In this paper, we propose a new scheme, referred to as Trained Rank Pruning (TRP), for training low-rank networks. We embed the low-rank decomposition into the training process to gradually push the weight distribution of a well functioning network into a low-rank form, where all parameters of the original network are kept and optimized to maintain the original capacity. We also propose a stochastic sub-gradient descent optimized nuclear regularization that further constrains the weights in low-rank space and boosts TRP. In addition, ranks are auto-selected during TRP training. An illustration of our proposed solution is shown in Fig. 1.

Overall, our contributions can be summarized as:

1. A new training scheme, referred to as TRP, which explicitly embeds the low-rank decomposition into the training process;
2. A nuclear norm regularized, stochastic sub-gradient descent method to boost the TRP;
3. Improved inference acceleration when combined with channel and spatial decompositions. Code is available at https://github.com/yuhuixu1993/Trained-Rank-Pruning.

2. Related Works

A lot of works have been proposed to accelerate the inference process of deep neural networks or to guide efficient network design. Briefly, these works could be categorized into three main categories: pruning, quantization, and low-rank decomposition.

Pruning Non-structured and structured sparsity are introduced by pruning. [12] proposes to prune unimportant connections between neural units with small weights in a pre-trained CNN, followed by network re-training to preserve the accuracy. [33] utilizes group Lasso strategy to learn the structure sparsity of networks including weights, channels, filters and layers. [22] adopts a similar strategy by explicitly imposing scaling factors on each channel to measure the importance of each connection and dropping those with small weights during training. In [14], the pruning problem is formulated as a data recovery problem. Pre-trained filters are re-weighted by minimizing a data recovery objective function. Channels with smaller weight are pruned. [23] heuristically selects filters using change of next layer's output as a criterion.

Quantization Weight quantization methods include training a quantized model from scratch [5, 6, 26] or converting a pre-trained model into quantized representation [38, 21, 11, 35]. The quantized weight representation includes binary value [26, 6] or hash buckets [5]. For training a quantized model from scratch, some works propose to use the gradient *w.r.t* the quantized weight to update the full-precision counterpart [6] or update the binarized weight with stochastic rounding scheme [10] during backward process. Note that our method here is inspired by the scheme of combining quantization with training process, *i.e.* we embed the low-rank decomposition into training process to explicitly guide the parameter to a low-rank form.

Low-rank decomposition Original models are decomposed into compact ones with more lightweight layers. [8] is one of earlier works to exploit low-rank approximation of convolution filters. The authors apply several tensor decomposition methods to decompose both convolution and fully connection layers. The overall speedup, however, is not notable. [16] considers both the spatial-wise and channel-wise redundancy and proposes decomposing a filter into two cascaded asymmetric filters. [37] further assumes the feature map lie in a low-rank subspace and decompose the convolution filter into $k \times k$ followed by 1×1 filters via SVD. [9] exploits the low-rank assumption of convolution filters and decompose a regular convolution into several depth-wise and point-wise convolution structures. Although these works achieved notable performance in network compression and acceleration, all of them are based on the low-rank assumption, which may limit the performance accuracy or result in large error when the assumption is not completely satisfied.

Alternatively, some other works [34, 1] implicitly utilize sparsity regularization to direct the neural network training process to learn a low-rank representation, and further decompose the network during network deployment. Our work is similar to this low-rank regularization method. However, rather than appending an implicit regularization during training, we impose an explicit sparsity constraint in our training process. We demonstrate that our approach pushes the weight distribution into a low-rank form quite effectively.

3. Methodology

3.1. Low-rank Approximation of Neural Networks

Formally, the convolution filters in a layer can be expressed by a tensor $W \in \mathbb{R}^{n \times c \times k_w \times k_h}$, where n and c are the number of filters and the input channels, k_h and k_w are the height and width of the filters. An input of the convolution layer $F_i \in \mathbb{R}^{c \times x \times y}$ generates an output as $F_o = W * F_i$. Channel-wise correlation [37] and spatial-wise correlation [16] [31] are considered to approximate convolution filters

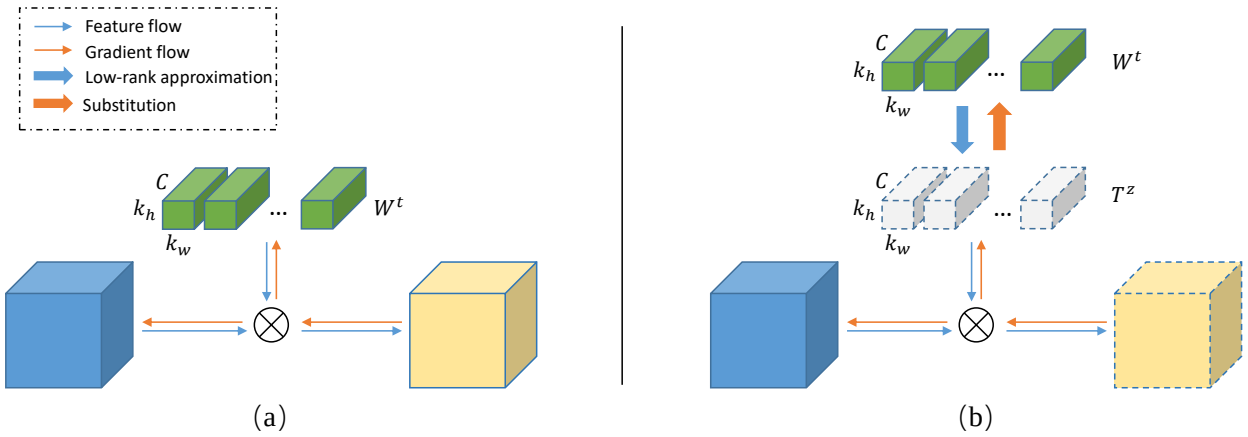


Figure 1. The training of TRP consisits of two parts as illustrated in (a) and (b), respectively. (a) one normal iteration with forward-backward broadcast and gradient update in filters. (b) one training iteration inserted by rank pruning, where the low-rank approximation is first applied on current filters before convolution. During backward propagation, the gradients are directly added on low-rank filters and the original weights are substituted by updated low-rank filters. (b) is adopted once every m iterations (*i.e.* when gradient update iteration $t = zm, z = 0, 1, 2, \cdots$), otherwise (a) is adopted.

into low-rank space. In this paper, we mainly focus on these two decomposition schemes. Channel-wise decomposition decomposes original tensor W into $U \in \mathbb{R}^{r\times c\times k_w\times k_h}$ and $M \in \mathbb{R}^{n\times r\times 1\times 1}$. Meanwhile, spatial-wise decomposition decomposes the tensor into $H \in \mathbb{R}^{r\times c\times 1\times k_h}$ and $V \in \mathbb{R}^{n\times r\times k_w\times 1}$. It is notable that both of the methods need to recover accuracy by re-training or data-driven strategy. Different from previous works that decompose pre-trained models, we propose a new scheme TRP to obtain a low-rank network.

3.2. Trained Rank Pruning

Trained Rank Pruning (TRP) is inspired by the strategies of training quantized nets. One of the gradient update schemes to train quantized networks from scratch [20] is

$$w^{t+1} = Q(w^t - \alpha\nabla f(w^t)) \tag{1}$$

where $Q(\cdot)$ is the quantization function ,w^t denote the parameter in the t^{th} iteration. Parameters are quantized by $Q(\cdot)$ before updating the gradients.

Similarly, we propose a training scheme Trained Rank Pruning (TRP) in a periodical manner:

$$\begin{aligned} W^{t+1} &= \begin{cases} W^t - \alpha\nabla f(W^t) & t\%m \neq 0 \\ T^z - \alpha\nabla f(T^z) & t\%m = 0 \end{cases} \\ T^z &= \mathcal{D}(W^t), \quad z = t/m \end{aligned} \tag{2}$$

where $\mathcal{D}(\cdot)$ is a certain tensor low-rank approximation operator and α is the learning rate. t is the gradient update iterations and z is the iteration of operator $\mathcal{D}$, m is the corresponding period for low-rank approximation.

In the quantization method, if the gradient updates are smaller than the quantization step, the gradient information would be lost. However, it will not happen in TRP because the low-rank operator is applied on the weight tensor. Furthermore, we apply low-rank approximation every m SGD iterations. This saves training time for a large scale. As detailed in Fig. 1, for every m iterations, we conduct low-rank approximation on the original filters, while gradients are updated on the low-rank form filters. At other time the network is updated via normal SGD. Note that the detailed process of low-rank approximation on filters depends on the adopted low-rank techniques since our training scheme could be combined with any tensor-based low-rank strategies, and in our work we mainly focus on the low-rank techniques proposed in [16] and [37], both of which transform the 4-dimensional filters into 2D matrix at first and then apply the truncated singular value decomposition (TSVD). The SVD of matrix W^t can be written as:

$$W^t = \sum_{i=1}^{rank(W^t)} \sigma_i \cdot U_i \cdot (V_i)^T \tag{3}$$

where σ_i is the singular value of W^t and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{rank(W^t)}$. U_i and V_i are the singular vectors. The parameterized TSVD($W^t; e$) is to find the smallest integer k that subjects to

$$\sum_{j=k+1}^{rank(W^t)} (\sigma_j)^2 \leq \quad e \sum_{i=1}^{rank(W^t)} (\sigma_i)^2 \tag{4}$$

where e is a pre-defined hyper-parameter as the energy ratio threshold, $e \in (0, 1)$. In our work, after truncating the last $n - k$ singular values, we transform the low-rank 2D

matrix back to 4D tensor in order to keep network structure unchanged during training. Compared with directly training low-rank structures from scratch, the proposed TRP has following advantages.

(1) Different from updating the decomposed form of parameters independently in existing works [37, 16], we fuse the decomposed parameters into original 4D shapes to update gradients, which means the capacity is preserved as the original network during the training of the low-rank network.

(2) Furthermore, since the gradient update is conducted based on the original network structure, there will be no exploding and vanishing gradients problems caused by additional layers.

(3) Rank of each layer is auto selected during the training process. We would provide theoretic analysis about the variation of the rank in section 3.4.

3.3. Nuclear Norm Regularization

Nuclear norm is widely used in matrix completion problems [3, 15]. Recently, it is introduced to constrain the network into low-rank space during the training process [1].

$$\min \left\{ f\left(x; w\right) + \lambda \sum_{l=1}^{L} ||W_l||_* \right\} \tag{5}$$

where $f(\cdot)$ is the objective loss function, nuclear norm $||W_l||_*$ is defined as $||W_l||_* = \sum_{i=1}^{rank(W_l)} \sigma_l^i$, with σ_l^i the singular values of W_l. λ is a hyper-parameter setting the influence of the nuclear norm. In [1] the proximity operator is applied in each layer independently to solve Eq. (5). However, the proximity operator is split from the training process and doesn't consider the influence within layers.

In this paper, we utilize stochastic sub-gradient descent [2] to optimize nuclear norm regularization in the training process. Let $W = U\Sigma V^T$ be the SVD of W and let U_{tru}, V_{tru} be U, V truncated to the first $rank(W)$ columns or rows, then $U_{tru}V_{tru}^T$ is the sub-gradient of $||W||_*$ [32]. Thus, the sub-gradient of Eq. (5) in a layer is

$$\nabla f + \lambda U_{tru} V_{tru}^T \tag{6}$$

The nuclear norm and loss function are optimized simultaneously during the training of the networks and can further be combined with the proposed TRP.

3.4. Theoretic Analysis

In this section, we analyze the rank convergence of TRP from the perspective of matrix perturbation theory [30]. We prove that rank in TRP is monotonously decreasing, *i.e.,* the model gradually converges to a more compact model.

Let A be an $m \times n$ matrix, without loss of generality, $m \geq n$. Then there are unitary matrices U and V such that

$$U^T A V = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} \tag{7}$$

where $\Sigma = diag\left(\sigma_1, \cdots, \sigma_n\right)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. σ_i is the diagonal matrix composed by all singular values of A.

Let $\widetilde{A} = A + E$ be a perturbation of A, and E is the noise matrix and then we have

$$\widetilde{U}^T \widetilde{A} \widetilde{V} = \begin{pmatrix} \tilde{\Sigma} \\ 0 \end{pmatrix} \tag{8}$$

where $\widetilde{\Sigma} = diag\left(\widetilde{\sigma}_1, \cdots, \widetilde{\sigma}_n\right)$ and $\widetilde{\sigma}_1 \geq \widetilde{\sigma}_2 \geq \cdots \geq \widetilde{\sigma}_n$. $\widetilde{\sigma}_i$ are the singular values of $\widetilde{A}$. The basic perturbation bounds for the singular values of a matrix are given by

Theorem 1. *Mirsky's theorem [25]:*

$$\sqrt{\sum_i |\widetilde{\sigma}_i - \sigma_i|^2} \leq ||E||_F \tag{9}$$

where $||\cdot||_F$ is the Frobenius norm. Then the following corollary can be inferred from Theorem 1,

Corollary 1. *Let B be any $m \times n$ matrix of rank not greater than k, i.e. the singular values of B can be denoted by $\varphi_1 \geq \cdots \geq \varphi_k \geq 0$ and $\varphi_{k+1} = \cdots = \varphi_n = 0$. Then*

$$||B - A||_F \geq \sqrt{\sum_{i=1}^{n} |\varphi_i - \sigma_i|^2} \geq \sqrt{\sum_{j=k+1}^{n} \sigma_j^2} \tag{10}$$

Below, we will analyze the training procedure of the proposed TRP. Note that W below are all transformed into 2D matrix. In terms of Eq. (2), the training process between two successive TSVD operations can be rewritten as Eq. (11)

$$\begin{aligned} W^t &= T^z = TSVD(W^t; e) \\ W^{t+m} &= T^z - \alpha \Sigma_{i=0}^{m-1} \nabla f(W^{t+i}) \\ T^{z+1} &= TSVD(W^{t+m}; e) \end{aligned} \tag{11}$$

where W^t is the weight matrix in the t-th iteration. T^z is the weight matrix after applying TSVD over W^t. $\nabla f(W^{t+i})$ is the gradient back-propagated during the $(t+i)$-th iteration. $e \in (0, 1)$ is the predefined energy threshold. We treat the gradient update as a matrix perturbation process.

Theorem 2. *Assume that $||\alpha \nabla f||_F$ has an upper bound G, if $G < \frac{\sqrt{e}}{m} ||W^{t+m}||_F$, then $rank(T^z) \geq rank(T^{z+1})$.*

Proof. We denote σ_j^t and σ_j^{t+m} as the singular values of W^t and W^{t+m} respectively. Then at the t-th iteration, given the

energy ratio threshold e, the TSVD operation tries to find the singular value index $k \in [0, n-1]$ such that :

$$\begin{aligned} \sum_{j=k+1}^{n} \left(\sigma_j^t\right)^2 &< e||W^t||_F^2 \\ \sum_{j=k}^{n} \left(\sigma_j^t\right)^2 &\geq e||W^t||_F^2 \end{aligned} \tag{12}$$

We consider the energy proportion of the last $n-k$ singular values in W^{t+m}. In terms of Eq. (12), T^z is a k rank matrix, i.e, the last $n-k$ singular values of T^z are equal to 0. According to Corollary 1, we can derive that:

$$\begin{aligned} ||W^{t+m} - T^z||_F &= ||\alpha \sum_{i=0}^{m-1} \nabla f^{t+i}||_F \\ &\geq \sqrt{\sum_{j=k+1}^{n} \left(\sigma_j^{t+m}\right)^2} \end{aligned} \tag{13}$$

Given the assumption $G < \frac{\sqrt{e}}{m}||W^{t+m}||_F$, we can get:

$$\begin{aligned} \frac{\sqrt{\sum_{j=k+1}^{n} \left(\sigma_j^{t+m}\right)^2}}{||W^{t+m}||_F} &\leq \frac{||\alpha \sum_{i=0}^{m-1} \nabla f^{t+i}||_F}{||W^{t+m}||_F} \\ &\leq \frac{\sum_{i=0}^{m-1} ||\alpha \nabla f^{t+i}||_F}{||W^{t+m}||_F} \\ &\leq \frac{mG}{||W^{t+m}||_F} < \sqrt{e} \end{aligned} \tag{14}$$

Eq. (14) indicates that since the perturbations of singular values are bounded by the parameter gradients, if we properly select the TSVD energy ratio threshold e, we could guarantee that if $n-k$ singular values are pruned by previous TSVD iteration, then before the next TSVD, the energy obtained from noise (*i.e.* gradient in CNN scenario) for the last $n-k$ singular values is still less than the pre-defined energy threshold e. Therefore TSVD should keep the number of pruned singular value or drop more to achieve the criterion in Eq. (12), consequently a weight matrix with lower rank or same rank is obtained, *i.e.* $Rank(T^z) \geq Rank(T^{z+1})$. We further confirm our analysis about the variation of rank distribution in Section 4. □

4. Experiments

4.1. Datasets and Baseline

We evaluate the performance of Trained Rank Pruning scheme on two common datasets, CIFAR-10 [17] and ImageNet [7] datasets. The CIFAR-10 dataset consists of colored natural images with 32×32 resolution and has totally 10 classes. The ImageNet dataset consists of 1000 classes of images for recognition task. The images are split into 1.2 million for training and 50,000 for validation. For both of the datasets, we adopt ResNet [13] as our baseline model since it is widely used not only in classification tasks but also other vision problems such as object detection and re-identification. As reported in [13], we use ResNet-20 and ResNet-32 for CIFAR-10 and ResNet-18 for ImageNet as our baseline models. For evaluation metric, we adopt the standard top-1 accuracy on CIFAR-10 and top-5 accuracy on ImageNet. To measure the acceleration performance, we compute the FLOPs ratio between baseline and decomposed models to obtain the final speedup rate. Wall-clock CPU and GPU time is also compared.

4.2. Implementation Details

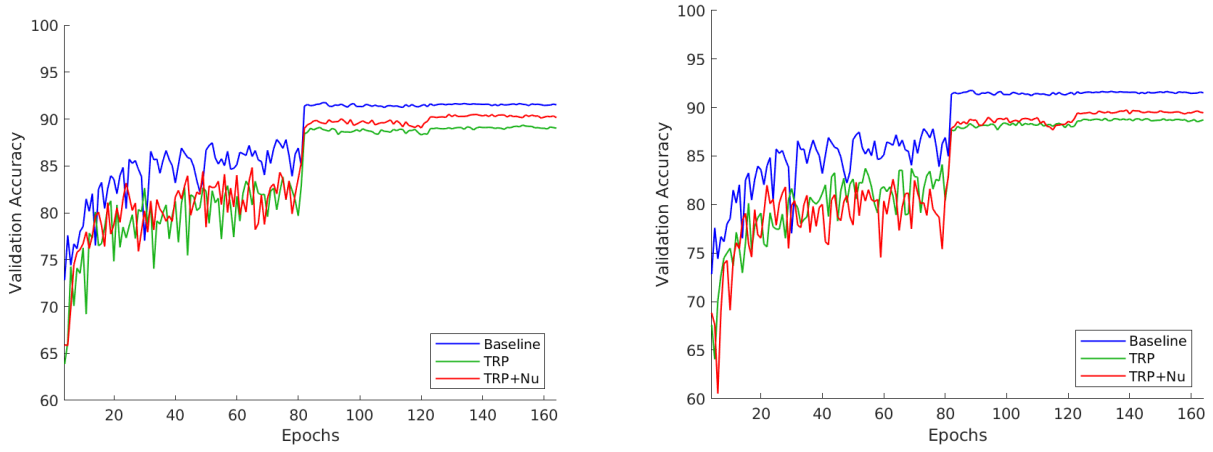
We implement our Trained Rank Pruning scheme on the PyTorch framework with an NVIDIA 1080 Ti GPU. For the training process on CIFAR-10, we start with base learning rate of 0.1 and degrade the value by a factor of 0.1 at the 82-th and 122-th epoch, totally we train our model for 164 epochs on this dataset. On the other hand, we adopt a slightly different training process for ImageNet, we directly finetune the model with Trained Rank Pruning scheme from the pre-trained baseline. On the ImageNet dataset, we also set 0.1 as our starting learning rate, and divide it by a factor of 10 every 10 epochs, we totally finetune the model for 35 epochs. For both of the dataset, we adopt SGD solver [18] to update gradient and set the weight decay value as 10^{-4} and momentum value as 0.9. The accuracy improvement enabled by data dependent decomposition vanishes after the fine-tuning process [31, 19]. For this reason, we simply adopt the retrained data independent decomposition as our basic methods.

4.3. Comparison with Training-from-scratch

We compare our methods with training decomposed low-rank networks from scratch and the results are shown in Table 1. In the spatial-wise decomposition, we train two decomposed models setting energy threshold e as 0.05 and 0.01 respectively with the proposed TRP+Nu ("Nu" indicates nuclear norm regularization). The trained models obtain 2.84× speed up with accuracy 90.52 and 4.69× speed up with accuracy 87.61. For training from scratch, we randomly initialize the decomposed models from TRP+Nu. The training process of both models does not converge.

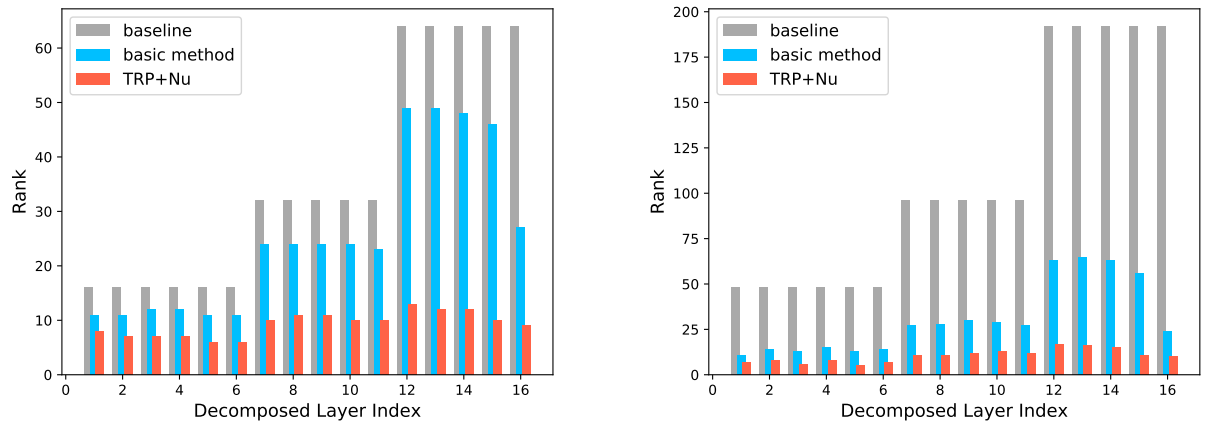
4.4. Comparison with Existing Methods

CIFAR-10. The results on CIFAR-10 are summarized in Table 2. We use ResNet-20 and ResNet-32 as our baseline models. Experiments on channel-wise decomposition and spatial-wise decomposition are both considered. The TSVD energy threshold in TRP and TRP+Nu is 0.02 and the nuclear norm weight λ is set as 0.0003. The rank of each de-



(a) Channel-wise decomposition (b) Spatial-wise decomposition

Figure 2. Validation accuracy curves of the proposed method. Undecomposed model training, TRP training and TRP combined with nuclear regularization are compared. In (a), the speed up rates of TRP and TRP with nuclear regularization are both $2.3\times$. In (b), the speed up rates are both $3.6\times$.



(a) Channel-wise decomposition (b) Spatial-wise decomposition

Figure 3. The rank in each decomposed convolution layer of ResNet-20. Baseline is the un-decomposed model. In channel-wise decomposition, the accuracy of basic method is 89.47%, and that of TRP+Nu is 89.36% respectively. In spatial-wise decomposition, the accuracy of basic method is 89.66%, and that of TRP+Nu is 89.72%.

Method	Top 1 (%)	Speed up
baseline	91.74	1.00×
from scratch	10.35 (failed)	4.69×
TRP+Nu	87.61	4.69×
from scratch	10.44 (failed)	2.84×
TRP+Nu	90.52	2.84×

Table 1. Comparison with training from scratch on CIFAR-10. Models in the same decomposed architecture from TRP+Nu are randomly initialized when training from scratch.

composed layer in ResNet-20 is shown in Fig. 3. The models trained by TRP+Nu have lower rank than those decomposed by basic methods in each layer. Fig. 3 also indicates that deeper layers are more redundant. As shown in Table 2, for both spatial-wise and channel-wise decomposition, the proposed TRP outperforms basic methods on ResNet-20 and ResNet-32. Results become even better when nuclear regularization is used. Fig. 2 is the training curves of the original model, TRP and TRP combined with nuclear regularization. With the same FLOPs, nuclear regularization boosts TRP method. In the spatial-wise decomposition, our results of TRP combined with nuclear regulariza-

Model	Top 1 (%)	Decomp.	Speed up
ResNet-20 (baseline)	91.74	/	1.00×
ResNet-20 ([37])	88.13	channel	1.41×
ResNet-20 (TRP)	90.12	channel	1.97×
ResNet-20 (TRP+Nu)	**90.50**	channel	**2.17**×
ResNet-20 ([16])	89.49	spatial	1.66×
ResNet-20 (TRP)	90.13	spatial	2.66×
ResNet-20 (TRP+Nu)	**90.62**	spatial	**2.84**×
ResNet-32 (baseline)	92.26	/	1.00×
ResNet-32 ([37])	89.50	channel	1.41×
ResNet-32 (TRP)	91.21	channel	1.81×
ResNet-32 (TRP+Nu)	**91.40**	channel	**2.20**×
ResNet-32 ([16])	89.97	spatial	1.68×
ResNet-32 (TRP)	90.85	spatial	2.86×
ResNet-32 (TRP+Nu)	**91.39**	spatial	**3.40**×

Table 2. Experiment results on CIFAR-10.

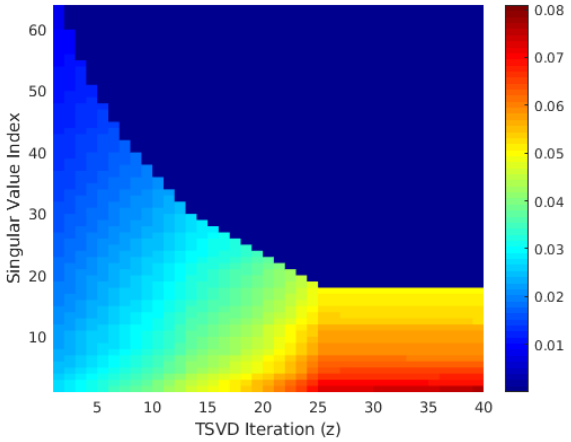


Figure 4. Visualization of rank selection along the training process, taken from the res3-1-2 convolution layer in ResNet-20 trained on CIFAR-10.

tion can even achieve 2× speed up rate than basic methods with accuracy kept the same. The results also show that, with the same accuracy, spatial-wise decomposition usually reduces more FLOPs than channel-wise decomposition.

Method	Top 5 (%)	Speed up
Baseline	88.54	1.00×
[37][1]	83.69	1.39×
[37]	84.44	1.41×
TRP1	86.48	1.81×
TRP2	85.63	2.20×
TRP2+Nu	85.52	2.50×

Table 3. Results of ResNet-18 on ImageNet by Channel-wise decomposition. The difference between TRP1 and TRP2 is that TRP1 does not decompose the 3*3 convolution layers with stride=2.

Method	Top 5 (%)	Speed up
Baseline	88.54	1.00×
[16]	83.72	2.00×
TRP1	86.74	2.60×
TRP2	85.83	3.20×
TRP2+Nu	85.30	3.68×

Table 4. Results of ResNet-18 on ImageNet by spatial-wise decomposition. The difference between TRP1 and TRP2 is that TRP1 does not decompose the 3*3 convolution layers with stride=2.

ImageNet. The results on ImageNet are shown in Table 3 and Table 4, which are channel-wise decomposition and spatial-wise decomposition respectively. We choose ResNet-18 as our baseline model. Results are compared with [37] and [16]. The difference between TRP1 and TRP2 is that TRP1 does not decompose the down-sampling 3×3 convolution layer with stride equals 2. By comparing TRP1 with TRP2, we can conclude that decomposing the down-sampling 3×3 convolution layers can increase the speed up rates. However, the accuracy drops a little. In both the channel-wise decomposition and the spatial-wise decomposition, the TSVD energy threshold e is set as 0.005. λ of nuclear norm regularization is 0.0003.

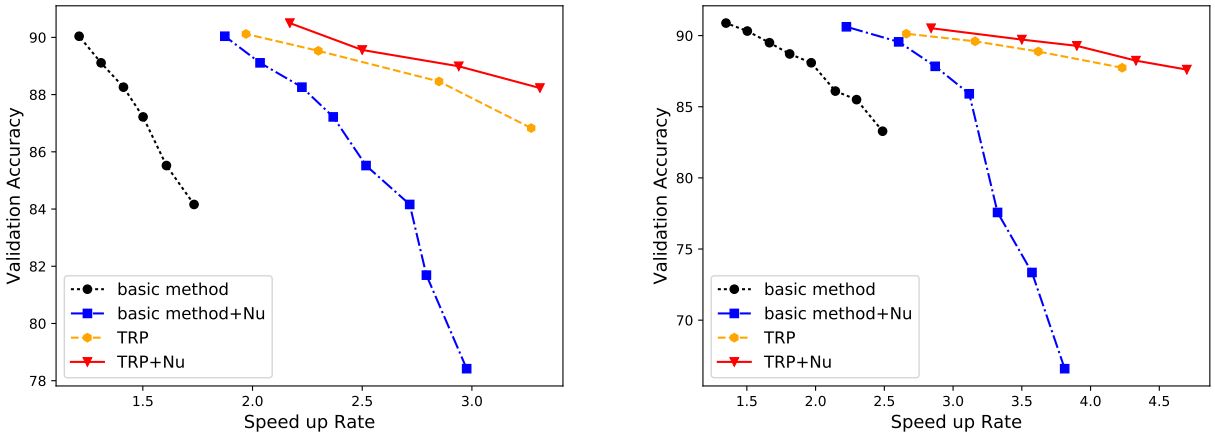
In the channel-wise decomposition, note that [37][1] is data driven method and [37] is data independent method with retraining. TRP2 obtains 2.2× speed up rate with 85.63 Top5 accuracy on ImageNet which outperforms both the data-driven and data independent methods by a large margin. Nuclear regularization can increase the speed up rates with the same accuracy.

In the spatial-wise decomposition, [16] is data independent method with retraining. For [16], the accuracy drops to 83.69 when speeds up rate is 2.0×. The proposed TRP can still achieve Top 5 accuracy 85.83 when the speed up rate is 3.2×. Nuclear regularization boosts the proposed TRP in the spatial decomposition.

4.5. Rank Variation

To analyze the variation of rank distribution during our training process, we further conduct an experiment by recording the energy ratio of singular values after each time of TSVD in the early stage of training. We evaluate the results on the CIFAR-10 dataset with ResNet-20 and extract the weight from the *res3-1-2* convolution layer with channel-wise decomposition as our low-rank scheme. After each time of TSVD, we compute and record the normalized energy ratio $ER(i)$ for each singular value σ_i in terms of

[1] the implementation of [9]



(a) Channel-wise decomposition

(b) Spatial-wise decomposition

Figure 5. Ablation study on ResNet-20. Basic methods are data-independent methods with retraining.

Eq. (15).

$$ER(i) = \frac{\sigma_i^2}{\sum_{j=1}^{rank(T^z)} \sigma_j^2} \tag{15}$$

we record for totally 40 times of TSVD with period $m = 20$, which is equal to 800 training iterations, and our energy theshold e is pre-defined as 0.05 according to the initial $||W^0||_F$ and weight gradient. Then we visualize our results in Fig. 4. During our training, we observe that the theoretic bound value $\max_t \frac{mG}{||W^t||_F} \approx 0.092 < \sqrt{e} \approx 0.223$, which indicates that our basic assumption in theorem 2 always holds for the initial training stage.

And this phenomenon is also reflected in Fig. 4, at the beginning, the energy distribution is almost uniform w.r.t each singular value, and the number of dropped singular values increases after each TSVD iteration since the energy from gradient is limited and the energy distribution becomes more dense among singular values with smaller index. Finally, the rank distribution converges to a certain point where the smallest energy ratio exactly reaches our threshold e and TSVD will not cut more singular values.

Model	GPU time (ms)	CPU time (ms)
Baseline	0.45	118.02
TRP+Nu(channel)	0.33	64.75
TRP+Nu(spatial)	0.31	49.88

Table 5. Actual running time per image of size 224×224. Baseline is full-rank ResNet-18 model.

4.6. Discussion

In order to show the effectiveness of different components of our method, we compare four training schemes, basic methods [37, 16], basic methods combined with nuclear norm regularization, TRP and TRP combined with nuclear norm regularization. The results are shown in Fig. 5. We can have following conclusions:

(1) *Nuclear norm regularization* After combining nuclear norm regularization, basic methods improve by a large margin. Besides, nuclear norm regularization also improves the performance of TRP both in the channel-wise decomposition and spatial-wise decomposition. Since Nuclear norm regularization constrains the filters into low rank space, the loss caused by TSVD is smaller than the basic methods.

(2) *Trained rank pruning* As depicted in Fig. 5, when the speed up rate increases, the performance of basic methods and basic methods combined with nuclear norm regularization degrades sharply. However, the proposed TRP degrades very slowly. This indicates that by reusing the capacity of the network, TRP can learn a better low-rank feature representations than basic methods. The gain of nuclear norm regularization on TRP is not as big as basic methods because TRP has already induced the parameters into low-rank space by embedding TSVD in training process.

4.7. Runtime Speed up of Decomposed Networks

We further evaluate the actual runtime speed up of the compressed Network as shown in Table 5. Our experiment is conducted on a platform with one Nvidia 1080Ti GPU and Xeon E5-2630 CPU. The models we used are the original Resnet-18 and decomposed models by TRP+Nu from Table 3 and Table 4. From the results, we observe that on CPU our TRP scheme achieves more salient acceleration performance. Overall the spatial decomposition combined with our TRP+Nu scheme has better performance. Because cuDNN is not friendly for 1×3 and 3×1 kernels, the actual speed up of spatial-wise decomposition is not as obvious as

the reduction of FLOPs.

5. Conclusion

In this paper, we propose a new scheme Trained Rank Pruning (TRP) for training low-rank networks. It leverages capacity and structure of the original network by embedding the low-rank approximation in the training process. Furthermore, we propose stochastic sub-gradient descent optimized nuclear norm regularization to boost the TRP. The proposed TRP can be incorporated with any low-rank decomposition method. On CIFAR-10 and ImageNet datasets, we have shown that our methods can outperform basic methods both in channel-wise decmposition and spatial-wise decomposition.

References

[1] J. M. Alvarez and M. Salzmann. Compression-aware training of deep networks. In *NIPS*, 2017. 1, 2, 4

[2] H. Avron, S. Kale, S. P. Kasiviswanathan, and V. Sindhwani. Efficient and practical stochastic subgradient descent for nuclear norm regularization. In *ICML*, 2012. 4

[3] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20:1956–1982, 2010. 4

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40:834–848, 2018. 1

[5] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *ICML*, 2015. 2

[6] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016. 1, 2

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009. 5

[8] E. Denton, W. Zaremba, J. Bruna, Y. Lecun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014. 1, 2

[9] J. Guo, Y. Li, W. Lin, Y. Chen, and J. Li. Network decoupling: From regular to depthwise separable convolutions. In *BMVC*, 2018. 1, 2, 7

[10] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. *Computer Science*, 2015. 2

[11] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. 2

[12] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 1, 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. 2016. 1, 5

[14] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 1, 2

[15] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He. Fast and accurate matrix completion via truncated nuclear norm regularization. *TPAMI*, 35:2117–2130, 2013. 4

[16] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014. 1, 2, 3, 4, 7, 8

[17] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science*, 2009. 5

[18] Y. Lecun, B. Boser, J. Denker, and D. Henderson. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 2014. 5

[19] C. Li and R. Shi. Constrained optimization based low-rank approximation of deep neural networks. In *ECCV*, 2018. 5

[20] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein. Training quantized nets: A deeper understanding. In *NIPS*, 2017. 3

[21] D. D. Lin, S. S. Talathi, and V. S. Annapureddy. Fixed point quantization of deep convolutional networks. *Computer Science*, 2016. 2

[22] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 1, 2

[23] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. *ICCV*, 2017. 1, 2

[24] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *arXiv preprint arXiv:1807.11164*, 2018. 1

[25] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59, 1960. 4

[26] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 1, 2

[27] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *TPAMI*, 39:1137–1149, 2015. 1

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, June 2018. 1

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1

[30] G. W. Stewart. Matrix perturbation theory. 1990. 4

[31] C. Tai, T. Xiao, Y. Zhang, X. Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015. 1, 2, 5

[32] G. A. Watson. Characterization of the subdifferential of some matrix norms. *Linear algebra and its applications*, 170:33–45, 1992. 4

[33] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016. 2

[34] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li. Coordinating filters for faster deep neural networks. In *ICCV*, 2017. 1, 2

[35] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong. Deep neural network compression with single and multiple level quantization. *CoRR*, abs/1803.03289, 2018. 1, 2

[36] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. 1

[37] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *TPAMI*, 38(10):1943–1955, 2016. 1, 2, 3, 4, 7, 8

[38] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 1, 2