LEARNING LOW-RANK DEEP NEURAL NETWORKS VIA SINGULAR VECTOR ORTHOGONALITY REGULARIZATION AND SINGULAR VALUE SPARSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern deep neural networks (DNNs) require high memory consumption and large computational loads. In order to deploy DNN algorithms efficiently on edge or mobile devices, a series of DNN compression algorithms have been explored, including the line of works on factorization methods. Factorization methods approximate the weight matrix of a DNN layer with multiplication of two or multiple low-rank matrices. However, it is hard to measure the ranks of DNN layers during the training process. Previous works mainly induce low-rank through implicit approximations or via costly singular value decomposition (SVD) process on every training step. The former approach usually induces a high accuracy loss while the latter prevents DNN factorization from efficiently reaching a high compression rate. In this work, we propose *SVD training*, which first applies SVD to decompose DNN's layers and then performs training on the full-rank decomposed weights. To improve the training quality and convergence, we add orthogonality regularization to the singular vectors, which ensure the valid form of SVD and avoid gradient vanishing/exploding. Low-rank is encouraged by applying sparsity-inducing regularizers on the singular values of each layer. Singular value pruning is applied at the end to reach a low-rank model. We empirically show that *SVD training* can significantly reduce the rank of DNN layers and achieve higher reduction on computation load under the same accuracy, comparing to not only previous factorization methods but also state-of-the-art filter pruning methods.

1 INTRODUCTION

The booming development in deep learning models and applications has enabled beyond human performance in tasks like large-scale image classification (Krizhevsky et al., 2012; He et al., 2016; Hu et al., 2018; Huang et al., 2017), object detection (Redmon et al., 2016; Liu et al., 2016; He et al., 2017), and semantic segmentation (Long et al., 2015; Chen et al., 2017). Such high performance, however, comes with a high price of large memory consumption and computation load. For example, a ResNet-50 model needs approximately 4G floating-point operations (FLOPs) to classify a color image of 224×224 pixels. The computation load can easily expand to tens or even hundreds of GFLOPs for detection or segmentation models using state-of-the-art (SOTA) DNNs as backbones (Canziani et al., 2016). This is a major challenge that prevents the deployment of modern DNN models on resource-constrained platforms, such as phones, smart sensors, and drones.

Model compression techniques for DNN models have been extensively studied. Some successful methods include element-wise pruning (Han et al., 2015; Liu et al., 2015; Zhang et al., 2018), structural pruning (Wen et al., 2016; Luo et al., 2017; Li et al., 2019), quantization (Liu et al., 2018; Wang et al., 2019), and factorization (Jaderberg et al., 2014; Zhang et al., 2015; Yang et al., 2015; Xu et al., 2018). Among these methods, quantization and element-wise pruning can effectively reduce model's memory consumption, but require specific hardware to realize efficient computation. Structural pruning reduces the computation load by removing redundant filters or channels. However, the complicated structures adopted in some modern DNNs (i.e., ResNet or DenseNet) enforce strict constraints on the input/output dimension of certain layers. This requires additional filter grouping during the pruning and filter rearranging after the pruning to make the pruned structure valid (Wen et al., 2017a; Ding et al., 2019). Factorization method approximates the weight matrix of

a layer with a multiplication of two or more low-rank matrices. It by nature keeps the input/output dimension of a layer unchanged, and therefore the resulted decomposed network can be supported by any common DNN computation architectures, without additional grouping and post-processing.

The previous investigation show that it is feasible to approximate the weight matrices of a pretrained DNN model with the multiplication of low-rank matrices, but it may greatly degrade the performance (Jaderberg et al., 2014; Zhang et al., 2015; Moczulski et al., 2015). Some other methods attempt to manipulate the "directions" of filters to implicitly reduce the rank of weight matrices (Wen et al., 2017b; Li et al., 2019). However, the difficulties in training and the implicitness of rank representation prevent these methods from reaching a high compression rate. Nuclear norm regularizer (Xu et al., 2018) has been used to directly reduce the rank of weight matrices. Optimizing the nuclear norm requires back propagation through singular value decomposition (SVD). Applying such a numerical process on every training step is inefficient and unstable.

Our work aims to explicitly achieve a low-rank DNN network during the training without applying SVD on every step. In particular, we propose *SVD training* by training the weight matrix of each layer in the form of its full-rank SVD. The weight matrix is decomposed into the matrices of left-singular vectors, singular values and right-singular vectors, and the training is done on the decomposed variables. Furthermore, two techniques are proposed to induce low-rank while maintaining high performance during the SVD training: (1) *Singular vector orthogonality regularization* which keeps the singular vector matrices close to unitary through the training. It mitigates gradient vanishing/exploding during the training, and provide a valid form of SVD to guarantee the effective rank reduction. (2) *Singular value sparsification* which applies sparsity-inducing regularizers on the singular values during the training to induce low-rank. The low-rank model is finally achieved through singular value pruning. We evaluate the individual contribution of each technique as well as the overall performance when putting them together via ablation studies. Results show that the proposed method constantly beats SOTA factorization and structural pruning methods on various tasks and model structures.

2 Related Works on low-rank DNNs

Approximating a weight matrix with the multiplication of low-rank matrices is a straightforward idea for compressing DNNs. Early works in this field focus on designing the matrix decomposition scheme, so that the operation of a pretrained network layer can be closely approximated with cascaded low-rank layers (Jaderberg et al., 2014; Zhang et al., 2015; Yang et al., 2015). Notably, Zhang et al. (2015) propose a channel-wise decomposition, which uses SVD to decompose a convolution layer with a kernel size $w \times h$ into two consecutive layers with kernel sizes $w \times h$ and 1×1, respectively. The computation reduction can be achieved by exploiting the channel-wise redundancy, e.g., channels with smaller singular values in both decomposed layers are removed. Similarly, Jaderberg et al. (2014) propose to decompose a convolution layer into two consecutive layers with less channels in between. They further utilize the spatial-wise redundancy to reduce the size of convolution kernels in the decomposed layers to $1 \times h$ and $w \times 1$, respectively. These methods provide a closed-form decomposition for each layer. However, the weights of the pretrained model may not be low-rank by nature, so the manually imposed low-rank after decomposition inevitably leads to high accuracy loss as the compression ratio increases (Xu et al., 2018).

Methods have been proposed to reduce the rank of weight matrices during training process in order to achieve low-rank decomposition with low accuracy loss. Wen et al. (2017b) induce low rank by applying an "attractive force" regularizer to increase the correlation of different filters in a certain layer. Ding et al. (2019) achieve a similar goal by optimizing with "centripetal SGD," which moves multiple filters towards a set of clustering centers. Both methods can reduce the rank of the weight matrices without performing actual low-rank decomposition during the training. However, the rank representations in these methods are implicit, so the regularization effects are weak and may lead to sharp performance decrease when seeking for a high speedup. On the other hand, Xu et al. (2018) explicitly estimate and reduce the rank throughout the training by adding Nuclear Norm (defined as the sum of all singular values) regularizer to the training objective. This method requires performing SVD to compute and optimize the Nuclear Norm of each layer on every optimization step. Since the complexity of the SVD operation is $\mathcal{O}(n^3)$ and the gradient computation through

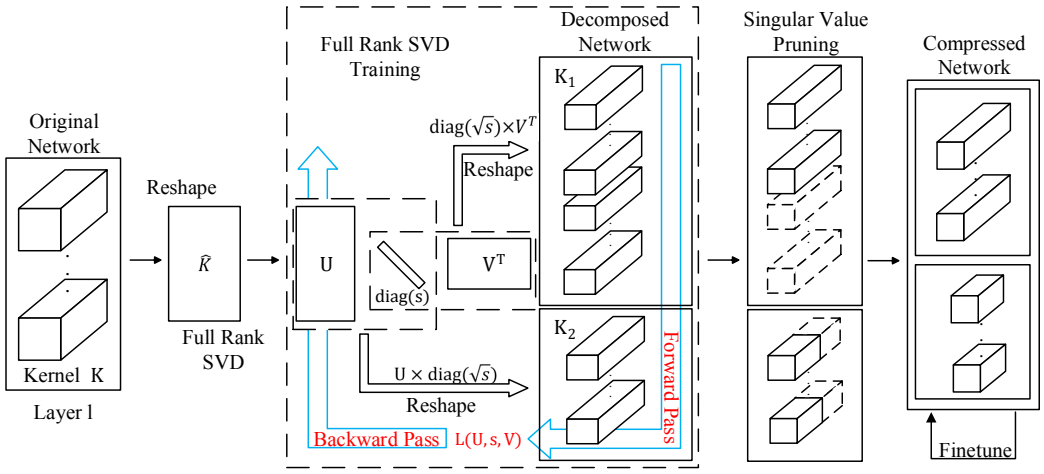

Figure 1: The training, compressing and finetuning pipeline of the proposed method.

SVD is not straightforward (Giles, 2008), performing SVD on every step is time consuming and leads to instability in the training process.

To explicitly achieve a low-rank network without performing decomposition on each training step, Tai et al. (2015) propose to directly train the network from scratch in the low-rank decomposed form. Since decomposition typically doubles the number of layers in the network, directly training the decomposed network may be difficult due to gradient vanishing or exploding. Tai et al. (2015) tackle this problem by adding batch normalization (Ioffe & Szegedy, 2015) between decomposed layers. However, adding batch normalization breaks the theoretical guarantee that the decomposed layers can approximate the operation of the original layer and will largely hurt the inference efficiency. Moreover, the low-rank decomposed training scheme used in this line of works requires setting the rank of each layer before the training (Tai et al., 2015; Ioffe & Szegedy, 2015). The manually chosen low rank may not lead to the optimal compression and will make the optimization harder as lower rank implies lower model capacity (Xu et al., 2018).

3 Proposed Method

Building upon previous works, we combine the ideas of decomposed training and trained low-rank in this work. As shown in Figure 1, the model will first be trained in a decomposed form through the full-rank SVD training, then undergoes singular value pruning for rank reduction, and finally be finetuned for further accuracy recovery. As we will explain in Section 3.1, the model will be trained in the form of the spatial-wise (Jaderberg et al., 2014) or channel-wise decomposition (Zhang et al., 2015) to avoid the time consuming SVD. Unlike the training procedure proposed by Tai et al. (2015), we will train the decomposed model in its full-rank to preserve the model capacity. During the SVD training, we apply orthogonality regularization to the singular vector matrices and sparsity-inducing regularizers to the singular values of each layer, the details of which will be discussed in Section 3.2 and 3.3, respectively. Section 3.4 will elaborate the full objective of the SVD training and the overall model compression pipeline. This method is able to achieve optimal compression rate by inducing low-rank through training without the need for performing decomposition on every training step.

3.1 SVD training of deep neural networks

In this work, we propose to train the neural network in its low-rank decomposition form, where each layer is decomposed into two consecutive layers via SVN, without introducing additional operations in between. For a fully connected layer, the weight $\boldsymbol{W}$ is a 2-D matrix with dimension $\boldsymbol{W} \in \mathbb{R}^{m \times n}$. Following the form of SVD, $\boldsymbol{W}$ can be directly decomposed into three variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{s}$ as $\boldsymbol{U}\text{diag}(\boldsymbol{s})\boldsymbol{V}^T$, with dimension $\boldsymbol{U} \in \mathbb{R}^{m \times r}$, $\boldsymbol{V} \in \mathbb{R}^{n \times r}$ and $\boldsymbol{s} \in \mathbb{R}^r$. Both $\boldsymbol{U}$ and $\boldsymbol{V}$ shall be unitary matrices. In the full-rank setting where $r = \min(m, n)$, $\boldsymbol{W}$ can be exactly reconstructed as

$\boldsymbol{W} = \boldsymbol{U}\text{diag}(\boldsymbol{s})\boldsymbol{V}^T$. For a neural network, this is equivalent to decomposing a layer with weight $\boldsymbol{W}$ into two consecutive layers with weight $\boldsymbol{W}_1 = \boldsymbol{U}\text{diag}(\sqrt{\boldsymbol{s}})$ and $\boldsymbol{W}_2 = \text{diag}(\sqrt{\boldsymbol{s}})\boldsymbol{V}^T$ respectively.

For a convolution layer, the kernel $\mathbf{K}$ can be represented as a 4-D tensor with dimension $\mathbf{K} \in \mathbb{R}^{n\times c\times w\times h}$. Here n, c, w, h represent the numbers of filters, the number of input channels, the width and the height of the filter respectively. This work mainly focuses on the *channel-wise decomposition* method (Zhang et al., 2015) and the *spatial-wise decomposition* method (Jaderberg et al., 2014) to decompose the convolution layer, as these methods have shown their effectiveness in previous CNN decomposition research. For channel-wise decomposition, $\mathbf{K}$ is first reshaped to a 2-D matrix $\hat{\boldsymbol{K}} \in \mathbb{R}^{n\times cwh}$. $\hat{\boldsymbol{K}}$ is then decomposed with SVD into $\boldsymbol{U} \in \mathbb{R}^{n\times r}$, $\boldsymbol{V} \in \mathbb{R}^{cwh\times r}$ and $\boldsymbol{s} \in \mathbb{R}^r$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices and $r = \min(n, cwh)$. The original convolution layer is therefore decomposed into two consecutive layers with kernels $\mathbf{K}_1 \in \mathbb{R}^{r\times c\times w\times h}$ reshaped from $\text{diag}(\sqrt{\boldsymbol{s}})\boldsymbol{V}^T$ and $\mathbf{K}_2 \in \mathbb{R}^{n\times r\times 1\times 1}$ reshaped from $\boldsymbol{U}\text{diag}(\sqrt{\boldsymbol{s}})$. Spatial-wise decomposition shares a similar process as the channel-wise decomposition. The major difference is that $\mathbf{K}$ is now reshaped to $\hat{\boldsymbol{K}} \in \mathbb{R}^{nw\times ch}$ and then decomposed into $\boldsymbol{U} \in \mathbb{R}^{nw\times r}$, $\boldsymbol{V} \in \mathbb{R}^{ch\times r}$, and $\boldsymbol{s} \in \mathbb{R}^r$ with $r = \min(nw, ch)$. The resulting decomposed layers would have kernels $\mathbf{K}_1 \in \mathbb{R}^{r\times c\times 1\times h}$ and $\mathbf{K}_2 \in \mathbb{R}^{n\times r\times w\times 1}$ respectively. Zhang et al. (2015) and Jaderberg et al. (2014)'s works theoretically show that the decomposed layers can exactly replicate the function of the original convolution layer in the full-rank setting. Therefore training the decomposed model at full-rank should achieve a similar accuracy as training the original model.

During the SVD training, for each layer we use the variables from the decomposition, i.e., $\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}$, instead of the original kernel $\mathbf{K}$ or weight $\boldsymbol{W}$ as the trainable variables in the network. The forward pass will be executed by converting the $\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}$ into a form of the two consecutive layers as demonstrated above, and the back propagation and optimization will be done directly with respect to the $\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}$ of each layer. In this way, we can access the singular value $\boldsymbol{s}$ directly without performing the time-consuming SVD on each step.

Note that $\boldsymbol{U}$ and $\boldsymbol{V}$ need to be orthogonal for efficient rank reduction via SVD, but this is not naturally induced by the decomposed training process. Also, Training with the decomposed variables may aggravate gradient vanishing/exploding during the optimization. Therefore we add orthogonality regularization to $\boldsymbol{U}$ and $\boldsymbol{V}$ to tackle the aforementioned problems, as discussed in Section 3.2. Rank reduction is induced by adding sparsity-inducing regularizers to the $\boldsymbol{s}$ of each layer, which will be discussed in Section 3.3.

3.2 Orthogonality regularization on singular vectors

In a standard SVD procedure, the resulted $\boldsymbol{U}$ and $\boldsymbol{V}$ should be orthogonal by construction, which provides theoretical guarantee for the low-rank approximation. However, $\boldsymbol{U}$ and $\boldsymbol{V}$ in each layer are treated as free trainable variables in the decomposed training process, so the orthogonality may not hold. Without the orthogonal property, it is unsafe to prune $\boldsymbol{s}$ even if it reaches a small value, because the corresponding singular vectors in $\boldsymbol{U}$ and $\boldsymbol{V}$ may have high energy and induce a large difference to the result. To make the form of SVD valid and enable effective rank reduction via singular value pruning, we introduce an orthogonality regularization loss to $\boldsymbol{U}$ and $\boldsymbol{V}$ as:

$$L_o(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{r^2}(||\boldsymbol{U}^T\boldsymbol{U} - \boldsymbol{I}||_F^2 + ||\boldsymbol{V}^T\boldsymbol{V} - \boldsymbol{I}||_F^2), \tag{1}$$

where $||\cdot||_F$ is the Frobenius norm of matrix and r is the rank of $\boldsymbol{U}$ and $\boldsymbol{V}$. Note that the ranks of $\boldsymbol{U}$ and $\boldsymbol{V}$ are same given their definition in the decomposed training procedure. Adding the orthogonality loss in Equation (1) to the total loss function forces $\boldsymbol{U}$s and $\boldsymbol{V}$s of all the layers close to be orthogonal matrices.

Meanwhile, one layer in the original network is converted to two consecutive layers in the decomposed training, therefore doubles the number of layers. As aforementioned by Ioffe & Szegedy (2015), this may worsen the problem of exploding or vanishing gradient during the optimization, degrading the performance of the achieved model. Since the proposed orthogonality loss can keep all the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ to have the L^2 norms close to 1, adding it to the decomposed training objective can effectively mitigate the exploding or vanishing gradient, therefore achieving high accuracy. The accuracy gain brought by training with the orthogonality loss will be discussed in our ablation study in Section 4.1.

3.3 SPARSITY-INDUCING REGULARIZERS ON SINGULAR VALUES

With orthogonal singular vector matrices, reducing the rank of the decomposed network is equivalent to making the singular value vector $\boldsymbol{s}$ of each layer sparse. Although the sparsity of a vector is directly represented by its L^0 norm, it is hard to optimize the norm through gradient-based methods. Inspired by the recent works in DNN pruning (Liu et al., 2015; Wen et al., 2016), we use differentiable sparsity-inducing regularizer to make more elements in $\boldsymbol{s}$ closer to zero, and apply post-train pruning to make the singular value vector sparse.

For the choice of the sparsity-inducing regularizer, the L^1 norm has been commonly applied in feature selection (Tibshirani, 1996) and DNN pruning (Wen et al., 2016). The L^1 regularizer takes the form of $L^1(\boldsymbol{s}) = \sum_i |s_i|$, which is both almost everywhere differentiable and convex, making it friendly for optimization. Moreover, applying L^1 regularizer on the singular value $\boldsymbol{s}$ is equivalent to regularizing with the nuclear norm of the original weight matrix, which is a popular approximation of the rank of a matrix (Xu et al., 2018).

However, the L^1 norm is proportional to the scaling of parameters, i.e., $||\alpha W||_1 = \alpha ||W||_1$, with a non-negative constant α. Therefore, minimizing the L^1 norm of $\boldsymbol{s}$ will shrink all the singular values simultaneously. In such a situation, some singular values that are close to zero after training may still contain a large portion of the matrix's energy. Pruning such singular values may undermine the performance of the neural network.

To mitigate the proportional scaling problem of the L^1 regularizer, previous works in compressed sensing have been using Hoyer regularizer to induce sparsity in solving non-negative matrix factorization (Hoyer, 2004) and blind deconvolution (Krishnan et al., 2011), where the Hoyer regularizer shows superior performance comparing to other methods. The Hoyer regularizer is formulated as

$$L^H(\boldsymbol{s}) = \frac{||\boldsymbol{s}||_1}{||\boldsymbol{s}||_2} = \frac{\sum_i |s_i|}{\sqrt{\sum_i s_i^2}}, \tag{2}$$

which is the ratio of the L^1 norm and the L^2 norm of a vector (Krishnan et al., 2011). It can be easily seen that the Hoyer regularizer is almost everywhere differentiable and scale-invariant. The differentiable property implies that the Hoyer regularizer can be easily optimized as part of the objective function. The scale-invariant property shows that if we apply the Hoyer regularizer to $\boldsymbol{s}$, the total energy will be retained as the singular values getting sparser. Therefore most of the energy will be kept within the top singular values while the rest getting close to zero. This makes Hoyer regularizer attractive in our training process. The effectiveness of the L^1 regularizer and the L^H regularizer is explored and compared in Section 4.2.

3.4 OVERALL OBJECTIVE AND TRAINING PROCEDURE

With the analysis above, we propose the overall objective function of the decomposed training as:

$$L(\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}) = L_T(\text{diag}(\sqrt{|\boldsymbol{s}|})\boldsymbol{V}^T, \boldsymbol{U}\text{diag}(\sqrt{|\boldsymbol{s}|})) + \lambda_o \sum_{l=1}^{D} L_o(\boldsymbol{U}_l, \boldsymbol{V}_l) + \lambda_s \sum_{l=1}^{D} L_s(\boldsymbol{s}_l). \tag{3}$$

Here L_T is the training loss computed on the model with decomposed layers. L_o denotes the orthogonality loss provided in Equation (1), which is calculated on the singular vector matrices $\boldsymbol{U}_l$ and $\boldsymbol{V}_l$ of layer l and added up over all D layers. L_s is the sparsity-inducing regularization loss, applying to the vector of singular values $\boldsymbol{s}_l$ of each layer. We explore the use of both the L^1 regularizer and the L^H regularizer in Equation (2) as L_s in this work. λ_s and λ_o are the decay parameters for the sparsity-inducing regularization loss and the orthogonality loss respectively, which are hyperparameters of the proposed training process. λ_o can be chosen as a large positive number to enforce the orthogonality of singular vectors, and λ_s can be modified to explore the tradeoff between accuracy and FLOPs of the achieved low-rank model.

As shown in Figure 1, the low-rank decomposed network will be achieved through a three-stage process of *full-rank SVD training*, *singular value pruning* and *low-rank finetuning*. First we train a full-rank decomposed network using the objective function in Equation (3). Training at full rank enables the decomposed model to easily reach the performance of the original model, as there is no capacity loss during the full-rank decomposition. With the help of the sparsity-inducing regularizer,

Table 1: Comparison of top 1 accuracy on CIFAR-10 of ResNet models after decomposed training, with or without orthogonality loss. [-Ch] means using channel-wise decomposition and [-Sp] means using spatial-wise decomposition.

Model	λ_o	Accuracy (%)	Model	λ_o	Accuracy (%)
ResNet-56	N/A	93.14	ResNet-110	N/A	93.62
ResNet-56-Ch	1.0	93.28	ResNet-110-Ch	1.0	93.58
ResNet-56-Ch	0.0	91.28	ResNet-110-Ch	0.0	91.83
ResNet-56-Sp	1.0	93.36	ResNet-110-Sp	1.0	93.93
ResNet-56-Sp	0.0	90.70	ResNet-110-Sp	0.0	91.86

most of the singular values will be close to zero after the full-rank training process. Inspired by Xu et al. (2018)'s work, we prune the singular values using an energy based threshold. For each layer we find a set $\mathbb{K}$ with the largest number of singular values subject to:

$$\sum_{j \in \mathbb{K}} s_j^2 \leq e \sum_{i=1}^{r} s_i^2, \tag{4}$$

where $e \in [0, 1]$ is a predefined energy threshold. We use the same threshold for all the layers in our experiments. When e is small enough, the singular values in set $\mathbb{K}$ and the corresponding singular vectors can be removed safely with negligible performance loss. The pruning step will dramatically reduce the rank of the decomposed layers. For a convolution layer with kernel $\mathbf{K} \in \mathbb{R}^{n \times c \times w \times h}$, if we can reduce the rank of the decomposed layers to r, the number of FLOPs for the convolution will be reduced by $\frac{(n+chw)r}{nchw}$ or $\frac{(nw+ch)r}{nchw}$ when channel-wise or spatial-wise decomposition is applied, respectively. The resulted low-rank model will then be finetuned with λ_s set to zero for further performance recovery.

4 Experiment Results

In this section, we first perform ablation studies on the importance of the singular vector orthogonality regularization and the choice of singular value sparsity regularizers. The studies use ResNet models (He et al., 2016) on the CIFAR-10 dataset (Krizhevsky & Hinton, 2009). We then apply the proposed decomposed training method on various DNN models on the CIFAR-10 dataset and the ImageNet ILSVRC-2012 dataset (Russakovsky et al., 2015). Different hyperparameters are used to explore the accuracy-FLOPs trade-off induced by the proposed method. Our results constantly stay above the Pareto frontier of previous works.

4.1 Importance of the orthogonal constraints

Here we demonstrate the importance of adding the singular value orthogonality loss to the decomposed training process. We separately train two decomposed model with the same optimizer and hyperparameters, one with the orthogonality loss of $\lambda_o = 1.0$ and the other with $\lambda_o = 0$. No sparsity-inducing regularizer is applied to the singular values in this set of experiments. The experiments are conducted on ResNet-56 and ResNet-110 models, both trained under channel-wise decomposition and spatial-wise decomposition. The CIFAR-10 dataset is used for training and testing. As shown in Table 1, the orthogonality loss enables the decomposed model to achieve similar or even better accuracy comparing to that of the original full model. On the contrary, training the decomposed model without the orthogonality loss will cause around 2% accuracy loss.

4.2 Comparison of decomposition methods and sparsity-inducing regularizers

With the proposed decomposed training method, there are two main factors related to the final compression rate and the performance of the compressed model: the decomposition method for the DNN layers and the choice of sparsity-inducing regularizers for the singular values. As mentioned in Section 3.1 and Section 3.3, we mainly consider the channel-wise and the spatial-wise decomposition

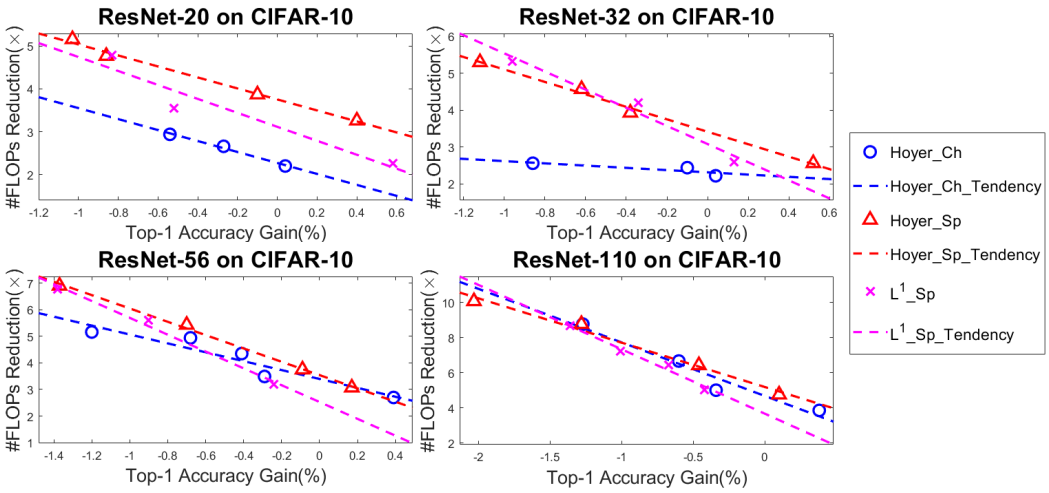

Figure 2: Effect of different decomposition methods and regularizers. Dash lines show the approximated tendency of the accuracy-compression tradeoff. See Appendix B for detailed data.

method, with the L^1 and the Hoyer regularizer. In this section, we comprehensively explore the accuracy-compression rate tradeoff of ResNet models in various depths and under different configurations by changing the strength of the sparsity-inducing regularizer (λ_s in Equation (3)). From the results shown in Figure 2, we make the following observations:

Channel-wise decomposition works as well as spatial-wise decomposition in deeper neural networks Here we compare the accuracy-#FLOPs tradeoff tendency of the channel-wise decomposition (red) and the spatial-wise decomposition (blue), both with the Hoyer regularizer. The spatial-wise decomposition shows a large advantage comparing to the channel-wise decomposition in the experiments with a shallower network like ResNet-20 or ResNet-32. However, with a deeper network like ResNet-110, these two decomposition methods perform similarly. Spatial-wise decomposition can utilize both spatial-wise redundancy and channel-wise redundancy, while the channel-wise decomposition utilizes channel-wise redundancy only. The observations indicate that as networks get deeper, the channel-wise redundancy will become a dominant factor comparing to the spatial-wise redundancy. This corresponds to the fact that deeper layers in modern DNN typically have significantly more channels than shallower layers, resulting in significant channel-wise redundancy.

Hoyer achieves higher speedup under low accuracy loss comparing to the L^1 regularizer Here we compare the effect of the L^1 regularizer (magenta) and the Hoyer regularizer (red), both with spatial decomposition. As shown in Figure 2, the tradeoff tendency of the L^1 regularizer constantly demonstrates a larger slope than that of the Hoyer regularizer. Under low accuracy loss, the Hoyer regularizer achieves a higher compression rate comparing to that of the L^1 regularizer. However, if we are aiming for extremely high compression rate while allowing higher accuracy loss, the L^1 regularizer can have a better performance. One possible reason for the difference in tendency is that the L^1 regularizer will make all the singular values small through the training process, while the Hoyer regularizer will maintain the total energy of the singular values during the training, focusing more energy in larger singular values. Therefore more singular values can be removed from the decomposed model trained with the Hoyer regularizer without significantly hurting the performance of the model, resulting in higher compression rate at low accuracy loss. But it would be harder to keep most of the energy in a tiny amount of singular values than simply making everything closer to zero, therefore the L^1 regularizer may perform better in the case of extremely high speedup.

4.3 Comparing with previous works

We apply the proposed SVD training framework on the ResNet-20, ResNet-32, ResNet-56 and ResNet-110 models on the CIFAR-10 dataset as well as the ResNet-50 model on the ImageNet ILSVRC-2012 dataset to compare the accuracy-#FLOPs tradeoff with previous methods. The hyperparameter choices of these models can be found in Appendix A. Here we mainly compare our

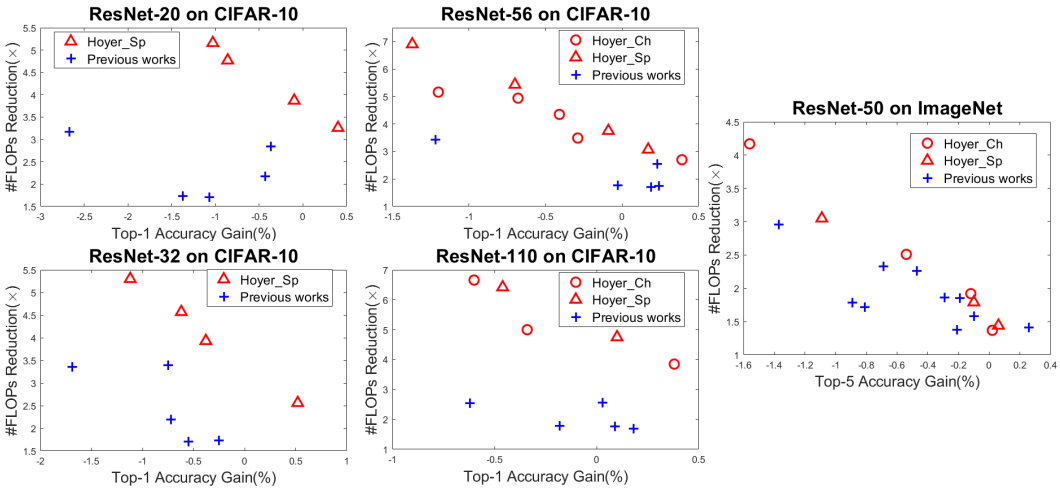

Figure 3: Comparison of accuracy-#FLOPs tradeoff against previous methods. Being closer to the top-right corner indicates a better tradeoff point. See Appendix B for detailed data.

method with state-of-the-art low-rank compression methods like TRP (Xu et al., 2018) and C-SGD (Ding et al., 2019), as well as recent filter pruning methods like NISP (Yu et al., 2018), SFP (He et al., 2018) and CNN-FCF (Li et al., 2019). The results of different models are shown in Figure 3. As analyzed in Section 4.2, the spatial-wise decomposition methods achieves significantly higher compression rate than the channel-wise decomposition in shallower networks, while similar performance can be achieved when compressing a deeper model. Thus we compare the results of only the spatial-wise decomposition against previous works for ResNet-20 and ResNet-32. For other deeper networks, we report the results for both channel-wise and spatial-wise decomposition. As most of the previous works focus on compressing the model with a small accuracy loss, here we use the Hoyer regularizer for the singular values sparsity, as it can achieve a better compression rate than the L^1 norm under low accuracy loss (see Section 4.2). We use multiple strength for the Hoyer regularizer to explore the accuracy-#FLOPs tradeoff, in order to compare against previous works with different accuracy levels. As shown in Figure 3, our proposed method can constantly achieve higher FLOPs reduction with less accuracy loss comparing to previous methods on different models and datasets. These comparison results prove that the proposed SVD training and singular value pruning scheme can effectively compress modern deep neural networks through low-rank decomposition.

5 Conclusion

In this work, we propose the SVD training framework, which incorporates the full-rank decomposed training and singular value pruning to reach low-rank DNNs with minor accuracy loss. We apply SVD to decompose each DNN layer before the training and directly train with the decomposed singular vectors and singular values, so we can keep an explicit measure of layers' ranks without performing the SVD on each step. Orthogonality regularizers are applied to the singular vectors during the training to keep the decomposed layers in a valid SVD form. And sparsity-inducing regularizers are applied to the singular values to explicitly induce low-rank layers.

Thorough experiments are done to analyse each proposed technique. We demonstrate that the orthogonality regularization on singular vector matrices is crucial to the performance of the decomposed training process. For decomposition methods, we find that the spatial-wise method performs better than channel-wise in shallower networks while the performances are similar for deeper models. For the sparsity-inducing regularizer, we show that higher compression rate can be achieved by Hoyer regularizer comparing to that of the L^1 regularizer under low accuracy loss. We further apply the proposed method to various depth of ResNet models on both CIFAR-10 and ImageNet dataset, where we find the accuracy-#FLOPs tradeoff achieved by the proposed method constantly stays above the Pareto frontier of previous methods, including both factorization and structural pruning methods. These results prove that this work provides an effective way for learning low-rank deep neural networks.

REFERENCES

Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 834–848, 2017.

Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4943–4953, 2019.

M Giles. An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation. an extended version of a paper that appeared in the proceedings of ad2008. In *the 5th International Conference on Automatic Differentiation*, 2008.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR 2011*, pp. 233–240. IEEE, 2011.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Tuanhui Li, Baoyuan Wu, Yujiu Yang, Yanbo Fan, Yong Zhang, and Wei Liu. Compressing convolutional neural networks via factorized convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3977–3986, 2019.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 806–814, 2015.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 722–737, 2018.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. Acdc: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946*, 2015.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016.

Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. Learning intrinsic sparse structures within long short-term memory. *arXiv preprint arXiv:1709.05027*, 2017a.

Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 658–666, 2017b.

Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. Trained rank pruning for efficient deep neural networks. *arXiv preprint arXiv:1812.02402*, 2018.

Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1476–1483, 2015.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018.

Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 184–199, 2018.

Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015.

A EXPERIMENT SETUPS

Our experiments are done on the CIFAR-10 dataset (Krizhevsky & Hinton, 2009) and the ImageNet ILSVRC-2012 dataset (Russakovsky et al., 2015). We access both datasets via the API provided in the "TorchVision" Python package. As recommended in the PyTorch tutorial, we normalize the data and augment the data with random crop and random horizontal flip before the training. We use batch size 100 to train CIFAR-10 model and use 256 for the ImageNet model. For all the models on CIFAR-10, both the full-rank SVD training and the low-rank finetuning are trained for 164 epochs. The learning rate is set to 0.001 initially and decayed by 0.1 at epoch 81 and 122. For models on ImageNet, the full-rank SVD training is trained for 90 epochs, with initial learning rate 0.1 and learning rate decayed by 0.1 every 30 epochs. The finetuning is done for 60 epochs, starting at learning rate 0.01 and decay by 0.1 at epoch 30. We use pretrained full-rank decomposed model (trained with the orthogonality regularizer but without sparsity-inducing regularizer) to initialize the SVD training. SGD optimizer with momentum 0.9 is used for optimizing all the models, with weight decay 5e-4 for CIFAR-10 models and 1e-4 for ImageNet models. The accuracy reported in the experiment is the best testing accuracy achieved during the finetuning process.

During the SVD training, the decay parameter of the orthogonality regularizer λ_o is set to 1.0 for both channel-wise and spatial-wise decomposition on CIFAR-10. On ImageNet, λ_o is set to 10.0 for channel-wise decomposition and 5.0 for spatial-wise decomposition. The decay parameter λ_s for the sparsity-inducing regularizer and the energy threshold used for singular value pruning are altered through different set of experiments to fully explore the accuracy-#FLOPs tradeoff. In most cases, the energy threshold is selected through a line search, where we find the highest percentage of energy that can be pruned without leading to a sudden accuracy drop. The λ_s and the energy thresholds used in each set of the experiments are reported alongside the experiment results in Appendix B.

B DETAILED EXPERIMENT RESULTS

In this section we list the exact data used to plot the experiment result figures in Section 4. The results of our proposed method with various choice of decomposition method and sparsity-inducing regularizer tested on the CIFAR-10 dataset are listed in Table 2. All of these data points are visualized in Figure 2 to compare the tradeoff tendency. As discussed in Section 4.3, the results of spatial-wise decomposition with the Hoyer regularizer for ResNet-20 and ResNet-32 are shown in Figure 3 to compare with previous methods. The results of both channel-wise and spatial-wise decomposition with the Hoyer regularizer are compared with previous methods in Figure 3 for ResNet-56 and ResNet-110. The results of our method for the ResNet-50 model tested on the ImageNet dataset are listed in Table 3. All reported ImageNet results are compared with previous methods in Figure 3.

Model	Reg Type	Decay	Energy Pruned	Accuracy Gain(%)	Speed Up
ResNet-20	Hoyer	0.03	1.5e-5	0.04	2.20 ×
Channel		0.07	6.0e-6	-0.27	2.66 ×
		0.1	3.0e-6	-0.54	2.94 ×
Base Acc:	L1	0.01	7.0e-2	1.13	1.43 ×
90.93%		0.001	2.7e-2	0.63	1.59 ×
		0.1	1.0e-1	0.32	2.10 ×
		0.3	1.0e-1	-0.48	2.84 ×
ResNet-20	Hoyer	0.01	1.0e-3	0.40	3.26 ×
Spatial		0.03	2.0e-5	-0.10	3.87 ×
		0.1	4.0e-6	-0.86	4.77 ×
Base Acc:		0.01	7.0e-3	-1.03	5.16 ×
90.99%	L1	0.01	6.0e-2	0.58	2.26 ×
		0.1	1.0e-1	-0.52	3.55 ×
		0.3	1.0e-1	-0.83	4.79 ×
ResNet-32	Hoyer	0.003	3.0e-3	0.04	2.22 ×
Channel		0.01	1.0e-4	-0.10	2.44 ×
		0.03	2.0e-6	-0.86	2.56 ×
Base Acc:	L1	0.03	2.0e-2	0.22	1.58 ×
92.12%		0.1	1.0e-1	-0.21	2.84 ×

		0.3	5.0e-2	-0.96	3.08 ×
ResNet-32	Hoyer	0.001	5.0e-2	0.52	2.56 ×
Spatial		0.005	5.0e-3	-0.38	3.93 ×
		0.01	8.0e-4	-0.62	4.57 ×
Base Acc:		0.03	8.0e-6	-1.12	5.30 ×
92.14%	L1	0.03	7.0e-2	0.13	2.60 ×
		0.1	2.5e-2	-0.34	4.20 ×
		0.1	1.5e-1	-0.96	5.32 ×
ResNet-56	Hoyer	0.001	2.0e-2	0.39	2.70 ×
Channel		0.003	1.0e-3	-0.29	3.49 ×
		0.01	7.0e-6	-0.41	4.35 ×
Base Acc:		0.01	2.0e-5	-0.68	4.94 ×
93.28%		0.03	3.0e-7	-1.20	5.16 ×
	L1	0.1	3.0e-2	-0.30	4.25 ×
		0.1	1.5e-1	-0.59	4.86 ×
ResNet-56	Hoyer	0.001	3.0e-2	0.17	3.07 ×
Spatial		0.003	1.0e-3	-0.09	3.75 ×
		0.01	1.0e-4	-0.70	5.43 ×
Base Acc:		0.03	1.0e-6	-1.37	6.90 ×
93.36%	L1	0.03	5.0e-3	-0.24	3.19 ×
		0.03	5.0e-2	-0.90	5.61 ×
		0.03	2.5e-1	-1.38	6.76 ×
ResNet-110	Hoyer	0.001	5.0e-3	0.38	3.85 ×
Channel		0.003	3.0e-4	-0.34	5.00 ×
		0.01	3.0e-7	-0.60	6.66 ×
Base Acc:		0.03	1.0e-6	-1.27	8.76 ×
93.58%	L1	0.03	1.0e-1	-0.28	5.02 ×
		0.03	3.0e-1	-1.27	7.44 ×
ResNet-110	Hoyer	0.001	1.3e-2	0.10	4.75 ×
Spatial		0.003	7.0e-4	-0.46	6.42 ×
		0.01	2.0e-5	-1.28	8.76 ×
Base Acc:		0.03	2.0e-8	-2.03	10.06×
93.93%	L1	0.03	3.0e-2	-0.42	5.02 ×
		0.03	1.0e-1	-0.67	6.45 ×
		0.03	1.5e-1	-1.01	7.21 ×
		0.03	2.5e-1	-1.36	8.66 ×

Table 2: Results of applying the proposed method on ResNet models on the CIFAR-10 dataset with various hyperparameters. [Decay] marks the decay variable for the sparse regularization, i.e. λ_s. [Energy Pruned] means the energy threshold used for singular value pruning, i.e. e. [Accuracy Gain] denotes the gain of Top-1 accuracy from the accuracy of the baseline full-rank model. [Speed Up] is computed as the ratio of #FLOPs of the original model and the achieved low-rank model.

The baseline results of previous works on compressing CIFAR-10 and ImageNet models used for comparison in Figure 3 are listed in Table 4 and Table 5 respectively. As there are a large amount of previous works in this field, we only list the results of the most recent works here to show the state-of-the-art Pareto frontier. Therefore we choose state of the art low-rank compression methods like TRP (Xu et al., 2018) and C-SGD (Ding et al., 2019), as well as recent filter pruning methods like NISP (Yu et al., 2018), SFP (He et al., 2018) and CNN-FCF (Li et al., 2019) as the baseline to compare our results against.

Table 3: Results of applying the proposed method on ResNet-50 model on the ImageNet dataset. Hoyer regularizer is used as the sparsity-inducing regularizer for the singular values. Top-5 testing accuracy is reported in the [Base Acc] and the [Accuracy Gain] columns. [Speed Up] is computed as the ratio of #FLOPs of the original model and the achieved low-rank model.

Decompose	Base Acc	Decay	Energy Pruned	Accuracy Gain	Speed Up
Channel	91.72%	0.001	1.0e-4	0.02%	1.37 ×
		0.002	1.0e-4	-0.12%	1.92 ×
		0.003	5.0e-5	-0.54%	2.51 ×
		0.005	5.0e-5	-1.56%	4.17 ×
Spatial	91.91%	0.0005	1.0e-3	0.06%	1.44 ×
		0.001	1.0e-4	-0.10%	1.79 ×
		0.002	2.0e-4	-1.09%	3.05 ×

Table 4: Baselines on the CIFAR-10 dataset. [Accu.↑] means the Top-1 accuracy gain comparing to that of the full model. [Sp. Up] denotes speed up computed as the ratio of #FLOPs before and after the model compression. [-] is marked when no result is available in the paper.

Method	ResNet-20		ResNet-32		ResNet-56		ResNet-110	
	Accu.↑	Sp. Up	Accu.↑	Sp. Up	Accu.↑	Sp. Up	Accu.↑	Sp. Up
TRP-Ch	-0.43%	2.17×	-0.72%	2.20×	-	-	-	-
TRP-Sp	-0.37%	2.84×	-0.75%	3.40×	-	-	-	-
SFP	-1.37%	1.79×	-0.55%	1.71×	0.19%	1.70×	0.18%	1.69×
CNN-FCF	-1.07%	1.71×	-0.25%	1.73×	0.24%	1.75×	0.09%	1.76×
	-2.67%	3.17×	-1.69%	3.36×	-1.22%	3.44×	-0.62%	2.55×
C-SGD-5/8	-	-	-	-	0.23%	2.55×	0.03%	2.56×
Nisp	-	-	-	-	-0.03%	1.77×	-0.18%	1.78×

Table 5: Baselines on the ImageNet dataset. [Accu.↑] means the Top-5 accuracy gain comparing to that of the full model. [Sp. Up] denotes speed up computed as the ratio of #FLOPs before and after the model compression.

Method	Accu.↑	Sp. Up	Method	Accu.↑	Sp. Up
SFP	-0.81%	1.72×	NISP-50-A	-0.21%	1.38×
CNN-FCF-A	+0.26%	1.41×	NISP-50-B	-0.89%	1.79×
CNN-FCF-B	-0.19%	1.85×	C-SGD-70	-0.10%	1.58×
CNN-FCF-C	-0.69%	2.33×	C-SGD-50	-0.29%	1.86×
CNN-FCF-D	-1.37%	2.96×	C-SGD-30	-0.47%	2.26×