Decomposable-Net: Scalable Low-Rank Compression for Neural Networks

Atsushi Yaguchi[1] , **Taiji Suzuki**[2,3] , **Shuhei Nitta**[1] , **Yukinobu Sakata**[1] and **Akiyuki Tanizawa**[1]

[1]Toshiba Corporation, Japan
[2]The University of Tokyo, Japan
[3]RIKEN Center for Advanced Intelligence Project, Japan
{atsushi.yaguchi,shuhei.nitta,yuki.sakata,akiyuki.tanizawa}@toshiba.co.jp, taiji@mist.i.u-tokyo.ac.jp

Abstract

Compressing DNNs is important for the real-world applications operating on resource-constrained devices. However, we typically observe drastic performance deterioration when changing model size after training is completed. Therefore, retraining is required to resume the performance of the compressed models suitable for different devices. In this paper, we propose Decomposable-Net (the network decomposable in any size), which allows flexible changes to model size without retraining. We decompose weight matrices in the DNNs via singular value decomposition and adjust ranks according to the target model size. Unlike the existing low-rank compression methods that specialize the model to a fixed size, we propose a novel backpropagation scheme that jointly minimizes losses for both of full- and low-rank networks. This enables not only to maintain the performance of a full-rank network *without retraining* but also to improve low-rank networks in multiple sizes. Additionally, we introduce a simple criterion for rank selection that effectively suppresses approximation error. In experiments on the ImageNet classification task, Decomposable-Net yields superior accuracy in a wide range of model sizes. In particular, Decomposable-Net achieves the top-1 accuracy of 73.2% with $0.27\times$MACs with ResNet-50, compared to Tucker decomposition ($67.4\%/0.30\times$), Trained Rank Pruning ($70.6\%/0.28\times$), and universally slimmable networks ($71.4\%/0.26\times$).

1 Introduction

Deep neural networks (DNNs) have achieved higher performance on various machine-learning tasks. However, since computational resources are limited on edge devices such as smartphones, it is desirable to optimize the model size suited to a specific device. Various methods for compressing DNNs have been proposed, including factorized convolutions [Howard *et al.*, 2017], low-rank approximations [Jaderberg *et al.*, 2014], pruning [Liu *et al.*, 2017], quantization [Han *et al.*, 2016], and neural architecture search [Tan *et al.*, 2019]. However, changing the model size is generally difficult for those methods once training and compression are completed; even when the same base model is reconfigured on different devices, retraining from scratch or fine-tuning according to the resources of the target devices is still required. In addition, for multi-task recognition systems operating on devices such as robots, it is preferable that the computational load of each task can be dynamically changed according to resource usage. One option is to store multiple models with different sizes in advance and switch those models, but doing so requires additional storage. Therefore, it is desirable that the model size can be flexibly changed without retraining, which we refer to as *scalability* in this paper.

To that end, [Yu *et al.*, 2019] introduced slimmable networks whose width can be changed to predefined ones after training. Moreover, [Yu and Huang, 2019] proposed universally slimmable networks (US-Nets) that extend slimmable networks to arbitrary widths. However, since these methods directly reduce the width (i.e., dimensionality) in each layer, the principal components are not taken into account. In addition, they uniformly reduce the width across all layers, which ignores differences in the importance of different layers.

In this paper, we propose Decomposable-Net as a novel scheme that allows DNNs to flexibly change their size after training. A weight matrix in each layer is decomposed into two low-rank matrices via singular value decomposition (SVD). By changing the rank in each layer, Decomposable-Net can scale the model to an arbitrary size (Figure 1(a)). In contrast to [Yu and Huang, 2019], we reduce the rank (i.e., the redundant basis in the principal subspace), for which the importance of each basis is naturally characterized by a singular value. This prevents the weight matrix in each layer from losing important directions, as illustrated in Figure 1(c). Moreover, our approach is essentially different from the existing low-rank compression methods [Kim *et al.*, 2016; Xu *et al.*, 2020] that apply low-rank approximation during or after learning and optimize the model for a specific size. Although Decomposable-Net is a single model and is not necessary to retrain, we show that it performs significantly better than those methods in a wide range of model sizes.

Our technical contributions are summarized as follows.

- We propose a novel backpropagation scheme that jointly minimizes losses for both of full- and low-rank networks (Figure 1(b)). It is designed not only to maintain the performance of a full-rank network but also to improve mul-

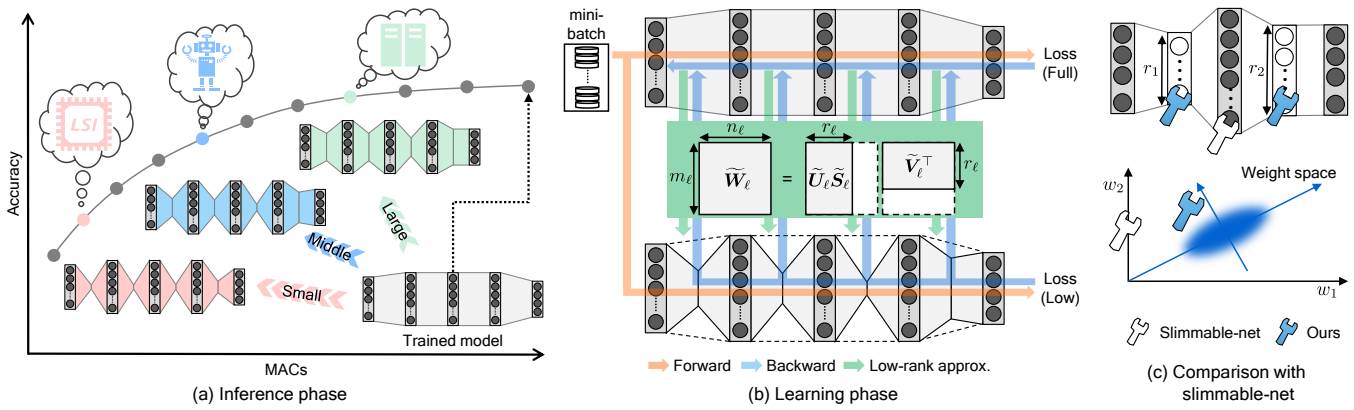

Figure 1: Illustration of Decomposable-Net. These schemes can also be applied to CNNs.

tiple low-rank networks (to be used for the inference).

- We introduce a simple criterion for the rank selection that utilizes a singular value for each basis, which effectively suppresses approximation error.
- The basic performace with SVD-based (e.g., channel- and spatial-wise) decompositions are enhanced by our methods, in addition to the scalability without retraining.

2 Methods

2.1 Overview

For the ℓ-th network layer, let $\boldsymbol{y}_\ell = \boldsymbol{W}_\ell^\top \boldsymbol{x}_\ell \in \mathbb{R}^{n_\ell}$ be an output vector given by the linear transformation of an input vector $\boldsymbol{x}_\ell = \phi(\boldsymbol{y}_{\ell-1}) \in \mathbb{R}^{m_\ell}$ with the weight matrix $\boldsymbol{W}_\ell \in \mathbb{R}^{m_\ell \times n_\ell}$, where $\phi(\cdot)$ is the activation function, and m_ℓ and n_ℓ are the numbers of input and output nodes, respectively. Let $R_\ell = \min(m_\ell, n_\ell)$ be the full rank of the weight matrix. Given $\boldsymbol{U}_\ell \in \mathbb{R}^{m_\ell \times R_\ell}$ and $\boldsymbol{V}_\ell \in \mathbb{R}^{n_\ell \times R_\ell}$ as matrices that have left and right singular vectors (i.e., bases) as columns, and $\boldsymbol{S}_\ell \in \mathbb{R}^{R_\ell \times R_\ell}$ as a diagonal matrix composed of singular values (σ_ℓ), we can formulate the SVD as $\boldsymbol{W}_\ell = \boldsymbol{U}_\ell \boldsymbol{S}_\ell \boldsymbol{V}_\ell^\top$.

An example of Decomposable-Net with fully connected layers is shown in Figure 1. Each weight matrix in the network is decomposed into two matrices of rank R_ℓ via SVD, and we control the rank to change the model size. This can be viewed as inserting a sub-layer between the original layers and changing its width from R_ℓ. The number of parameters in each layer becomes $(m_\ell + n_\ell) R_\ell$ by this decomposition. Thus, we can compress the network to an arbitrary size by changing the rank r_ℓ ($\leq R_\ell$) within the range $1 \leq r_\ell < m_\ell n_\ell / (m_\ell + n_\ell)$. For CNNs, we transform convolutional tensors to the matrix form and apply the low-rank approximation to every convolutional and fully connected layer. Note that our methods do not rely on a specific decomposition form, and the SVD-based methods, including channel- and spatial-wise [Zhang *et al.*, 2016; Tai *et al.*, 2016] decompositions, can be adopted. However, these low-rank approximations solely deteriorate the performance. Thus, we propose a learning method to alleviate it.

2.2 Learning

For the L-layer network, let $\mathcal{W} = \{\boldsymbol{W}_\ell\}_{\ell=1}^L$ be a set of weight matrices and let $\widetilde{\mathcal{W}} = \{\widetilde{\boldsymbol{W}}_\ell\}_{\ell=1}^L$ be a set of their low-rank approximations, respectively. As shown in Figure 1(b), we additionally propagate each mini-batch to a low-rank network whose weight $\widetilde{\mathcal{W}}$ is generated from $\mathcal{W}$ via SVD. Other trainable parameters Θ (e.g., biases) are shared between the full- and low-rank networks [1]. Therefore, the number of total parameters is the same as in normal learning.

For the purpose to maximize the performance of smaller subnetworks, we propose to jointly minimize losses for both full- and low-rank networks as follows:

$$\min_{\mathcal{W},\Theta} \frac{1}{B} \sum_{b=1}^{B} \left\{ (1-\lambda) \underbrace{\mathcal{L}(\mathcal{D}_b, \mathcal{W}, \Theta)}_{\text{loss (full-rank)}} + \lambda \underbrace{\mathcal{L}(\mathcal{D}_b, \widetilde{\mathcal{W}}, \Theta)}_{\text{loss (low-rank)}} \right\} + \eta \mathcal{R}(\mathcal{W}) . \quad (1)$$

Here, $\mathcal{L}(\cdot)$ is a loss function, $\mathcal{D}_b$ is a training sample in a mini-batch, B is the batch size, and $\lambda \in [0, 1]$ is a hyperparameter for balancing between the two losses. $\mathcal{R}(\cdot)$ is a regularization function, for which we adopt the sum of squared Frobenius norms $\mathcal{R}(\mathcal{W}) = \frac{1}{2}\sum_{\ell=1}^L \|\boldsymbol{W}_\ell\|_F^2$, and η is a hyperparameter to control its strength. In order to learn models in an end-to-end manner, backpropagation through SVD is involved in computing $\boldsymbol{\nabla}_\ell := \partial \mathcal{L}(\mathcal{D}_b, \widetilde{\mathcal{W}}, \Theta) / \partial \boldsymbol{W}_\ell$. The detailed derivation of the gradient and a practical technique for improving the numerical stability are given in appendix A of the extended version of this paper [Yaguchi *et al.*, 2019]. The gradients for Θ are simply computed from the λ-weighted average of both networks.

Since we need a single model that performs well at multiple sizes, we aim to optimize all the low-rank networks to be selected at the inference phase. However, it is computationally intractable to do so in each iteration step. Thus, we select one low-rank network with a random size from a predifined range. Specifically, a ratio Z of the number of bases

[1] For the batch normalization (BN) layer, γ and β are shared. μ and σ^2 are not shared and are not tracked by moving average during training. These are computed on the whole training set for each model after training, as in US-Nets.

(referred to as *rank ratio*) is sampled from uniform distribution $\mathcal{U}(\alpha_l, \alpha_u)$ with $0 < \alpha_l < \alpha_u \leq 1$. Then, we reduce $d = (1 - Z)\sum_{\ell=1}^{L} R_\ell$ bases across the entire network using a criterion described in the next subsection. The pseudo-code for learning Decomposable-Net is given in Algorithm 1.

2.3 Rank Selection

At the stage of compression, it is critically important to select an appropriate rank for maximizing the performance (as we see in Figure 2(c)). Given a target size for a model, we select the rank of each layer by reference to a simple criterion: $C(\ell, i) = \sigma_{\ell,i}$ (a singular value), where $i \in [1, R_\ell]$ is a basis index. In practice, we globally sort the singular values of all bases in the entire network in ascending order and store them as a single list. The only step necessary for selection is to reduce the first d bases in the list. We apply the same criterion to learning and inference. d is determined by the rank ratio Z during learning and the target model size for inference.

The impact of removing a weight $w \in \boldsymbol{W}$ on a loss $\mathcal{L}(\boldsymbol{W})$ can be approximated via the first-order Taylor expansion: $\mathcal{L}(\boldsymbol{W}|_{w=0}) - \mathcal{L}(\boldsymbol{W}) \approx -w \cdot (\partial\mathcal{L}(\boldsymbol{W})/\partial w)$, and its absolute value can be used as an importance measure for pruning [Lubana and Dick, 2021]. In pruning during learning, [Lubana and Dick, 2021] showed that keeping large weights results in faster convergence, and using only the magnitude $|w|$ works well as the importance. According to that, we can characterize the importance of each basis by $\sigma_{\ell,i} \cdot |\partial\mathcal{L}(\boldsymbol{W}_\ell)/\partial\sigma_{\ell,i}|$, which is proportional to the magnitude of singular value $\sigma_{\ell,i}$. Therefore, it is expected that our criterion simply using singular values can supress the impact on the loss (i.e., approximation error). Note that our criterion is different from the energy-based criterion [Alvarez and Salzmann, 2017; Xu *et al.*, 2020]: $C'(\ell, i) = \sum_{k=1}^{i} \sigma_{\ell,k}^2 / \sum_{j=1}^{R_\ell} \sigma_{\ell,j}^2$. Because it evaluates the accumulated energy ratio for each matrix, the importance characterized by the magnitude of singular value is ignored. In section 4, we experimentally show that our criterion outperforms it.

2.4 Analysis

Here, we describe the mathematical interpretation for our methods and show that Decomposable-Net possesses favorable properties as a scalable model. Proofs of the propositions described below are given in appendix B and C of [Yaguchi *et al.*, 2019].

Proposition 1. *Let $\widetilde{\boldsymbol{W}}_\ell(r_\ell)$ be a rank-r_ℓ approximation for $\boldsymbol{W}_\ell$ via SVD, and let $\hat{\boldsymbol{y}}_\ell(r_\ell) = \widetilde{\boldsymbol{W}}_\ell(r_\ell)^\top \boldsymbol{x}_\ell$. Then, the squared error between an original $\boldsymbol{y}_\ell$ and $\hat{\boldsymbol{y}}_\ell(r_\ell)$ satisfies $\|\boldsymbol{y}_\ell - \hat{\boldsymbol{y}}_\ell(1)\|^2 \geq \cdots \geq \|\boldsymbol{y}_\ell - \hat{\boldsymbol{y}}_\ell(r_\ell)\|^2 \geq \|\boldsymbol{y}_\ell - \hat{\boldsymbol{y}}_\ell(r_\ell + 1)\|^2 \geq \cdots \geq \|\boldsymbol{y}_\ell - \hat{\boldsymbol{y}}_\ell(R_\ell)\|^2 = 0$.*

Therefore, the errors between the original output $\boldsymbol{y}_\ell$ and its approximation $\hat{\boldsymbol{y}}_\ell$ monotonically decrease as the rank increases. Although this is a natural property of SVD, it is desirable for changing the model size with maintaining the performance. It does not exactly hold for the entire network, but we can show that changes of the upper bound depends on the layer-wise error $\|\boldsymbol{y}_\ell - \hat{\boldsymbol{y}}_\ell\|^2$:

Algorithm 1 Learning Decomposable-Net

Require: Training set $\mathcal{S}$, loss function $\mathcal{L}$, neural network $\mathcal{N}$ with a set of weight matrices $\mathcal{W} = \{\boldsymbol{W}_\ell\}_{\ell=1}^{L}$ in which $\boldsymbol{W}_\ell$ has rank R_ℓ, and other trainable parameters Θ. Regularization parameter η, balancing parameter $\lambda \in [0, 1]$, lower and upper rank ratios of α_l and α_u ($0 < \alpha_l < \alpha_u \leq 1$), batch size B, and criterion C for rank selection
Ensure: Return $\mathcal{N}$ with learned $\mathcal{W}$ and Θ
1: Initialize $\mathcal{W}$ and Θ
2: **repeat**
3: Draw Z from the uniform distribution $\mathcal{U}(\alpha_l, \alpha_u)$
4: Select the ranks of the weight matrices $\{r_\ell\}_{\ell=1}^{L}$ by reducing $d = (1 - Z)\sum_{\ell=1}^{L} R_\ell$ bases across the entire network, according to the criterion C
5: Build a low-rank network with a set of weight matrices $\widetilde{\mathcal{W}}$, via rank-r_ℓ approximation of $\boldsymbol{W}_\ell$ for $\ell \in [1, L]$
6: Draw a batch of samples $\{\mathcal{D}_b\}_{b=1}^{B}$ from $\mathcal{S}$
7: Compute the gradients of the objective function in Eq. (1), with respect to $\boldsymbol{W}_\ell \in \mathcal{W}$ and $\theta \in \Theta$
8: Update $\boldsymbol{W}_\ell \in \mathcal{W}$ and $\theta \in \Theta$ with the gradients
9: **until** stopping criterion is met

Proposition 2. *For a fully connected network with 1-Lipschitz activation function (e.g., ReLU), let $\boldsymbol{p}$ and $\widetilde{\boldsymbol{p}}$ be K-class softmax outputs from the full- and low-rank networks, respectively. Then, KL divergence between them can be bounded as* $\mathrm{KL}(\boldsymbol{p}||\widetilde{\boldsymbol{p}}) \leq$

$$\sqrt{2\left(\|\boldsymbol{y}_L - \hat{\boldsymbol{y}}_L\|^2 + \sum_{\ell=1}^{L-1}\left\{\|\boldsymbol{y}_\ell - \hat{\boldsymbol{y}}_\ell\|^2 \prod_{j=\ell+1}^{L} \|\boldsymbol{W}_j\|_2^2\right\}\right)},$$

where $\|\cdot\|_2$ indicates the spectral norm.

Hence, it can be expected that the performance of entire network changes in conjunction with the rank in each layer. Regarding the Lipschitz constant $\prod_{j=\ell+1}^{L} \|\boldsymbol{W}_j\|_2^2$, we experimentally verified that the empirical value is much smaller than the theoretical one [Wei and Ma, 2019] (see appendix E of [Yaguchi *et al.*, 2019]).

Under the condition that a trained network has near low-rank weight matrices, [Arora *et al.*, 2018; Suzuki *et al.*, 2020a; Suzuki *et al.*, 2020b] proved that the condition contributes not only to compressing the network efficiently, but also to yielding a better generalization error bound for the non-compressed (full-rank) network. That motivates our approach, namely, a learning scheme that aims to facilitate the performance of low-rank networks as well as that of a full-rank network.

3 Related Work

3.1 Low-Rank Approximations

Compression methods based on low-rank approximation have been proposed in the literature, and those can be divided into three categories. (1) Before learning, [Ioannou *et al.*, 2016] factorized a convolutional kernel into horizontal and vertical kernels. However, the rank in each layer needs to be manually selected. (2) After learning, [Jaderberg *et al.*, 2014;

Zhang *et al.*, 2016] applied data-dependent optimization with the low-rank constraint. Alternatively, tensor methods (e.g., CP and Tucker decompositions) that directly approximate four-way convolutional tensors have been adopted [Lebedev *et al.*, 2015; Kim *et al.*, 2016]. However, since the approximation is applied after learning the model in each of these methods, the low-rankness is not necessarily ensured. Moreover, they optimize networks under a given target size, and thus optionally require retraining or fine-tuning to reconfigure the models for different devices. (3) During learning, [Alvarez and Salzmann, 2017; Xu *et al.*, 2020] utilized trace-norm regularization as a low-rank constraint. The performance of these methods depends on a hyperparameter for adjusting the strength of regularization. It is difficult to decide on an appropriate range for the hyperparameter in advance, meaning that selection requires trial and error to achieve a particular model size.

Our methods are related to (3), in particular to Trained Rank Pruning (TRP) [Xu *et al.*, 2020]. TRP minimizes the loss only for the low-rank network produced by SVD. Since it applies trace-norm regularization and truncates small singular values during learning, the resulting ranks in the low-rank network are fixed. Thus, the network does not perform well with other ranks, as shown in section 4.

3.2 Scalable Networks

[Yu *et al.*, 2019; Yu and Huang, 2019] proposed slimmable networks and US-Nets, which are scalable in the width direction. These works are closely related to ours, but there are differences in some aspects. First, since these methods directly and uniformly reduce the width for every layer, the principal components are not taken into account, and the relative importance of each layer is not considered. Second, although the US-Nets allow arbitrary changes in width, monotonicity of error is not guaranteed when changing widths in each layer.

Recent works further acquired scalability in depth [Zhang *et al.*, 2019], resolution [Cai *et al.*, 2020; Yu *et al.*, 2020; Yang *et al.*, 2020], and kernel size [Cai *et al.*, 2020; Yu *et al.*, 2020]. The proposed methods are focused on improving the width scalability, but can be complementarily combined with those directions. This topic remains as our future works.

4 Experiments

We evaluate our methods on image-classification tasks of CIFAR-10/100 [Krizhevsky, 2009] and ImageNet [Deng *et al.*, 2009] datasets using deep CNNs. For the CIFAR-10 dataset, we adopt a VGG-like network with fifteen layers (VGG-15) [Zagoruyko, 2015; Liu *et al.*, 2017]. For the CIFAR-100 and ImageNet datasets, we adopt ResNet-34 and ResNet-50 [He *et al.*, 2016], respectively. We follow the same baseline setup for the CIFAR datasets as used by [Zagoruyko and Komodakis, 2016], and the setup of [Yu *et al.*, 2019] for the ImageNet dataset. The additional details of experiments are described in appendix D of [Yaguchi *et al.*, 2019].

For all experiments, we report the results of the models after the last epoch. The results are averaged over multiple runs with different random seeds, specifically, with five seeds $(0 \sim 4)$ on the CIFAR datasets and three seeds $(0 \sim 2)$ on the ImageNet dataset. All methods are evaluated in terms of the tradeoff between validation (top-1) accuracy and the number of multiply-accumulate operations (MACs). The number of MACs is defined as $\sum_{\ell=1}^{L} P_\ell H_\ell W_\ell$, where P_ℓ, H_ℓ, and W_ℓ are the number of parameters, and the height and width of an output feature map, respectively. We compare channel- and spatial-wise decompositions for convolutional layers, but use the channel-wise one for 1×1 kernels, since they have no spatial redundancy. We experimentally tuned hyperparameters, and set $\alpha_u = 0.25$, 0.5, and 0.8, respectively for CIFAR-10, CIFAR-100, and ImageNet datasets, while fixing $\alpha_l = 0.01$ for all datasets. Except for the results shown in Figure 2(a), we set $\lambda = 0.5$ to balance the performance of a full- and low-rank models. We implemented other methods in the following experiments and evaluated them under our settings.

4.1 Ablation Study

We verify each component in our methods with ResNet-34 on the CIFAR-100 dataset. Firstly, we evaluate our learning method with various values for parameters $\lambda \in [0, 1]$ that balance between two losses. Each model learned with a specific λ is evaluated for several sizes of the rank ratio $Z \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1\}$ without retraining. The results are illustrated in Figure 2(a). Each curve in the figure shows the performance changes in a specific model size (e.g., `0.05` and `1` correspond to 5% and the full model, respectively) against λ, where points at $\lambda = 0$ correspond to results from normal learning. We can see that the performances of compressed models improve as λ increases, while that of the full model is almost flat. This implies that our learning method not only maintains the performance of full-rank network, but also improves multiple low-rank networks in a single model. Notably, the model with $\lambda = 0.3$ achieves 77.65% at full size, which is better than the 77.49% under normal learning, demonstrating that the compressible network generalizes well [Suzuki *et al.*, 2020b]. The effect of our method is also verified in Figure 2(b), where ours (`decomp-net`) is consistently better than normal learning (`base`), with both channel- (`ch`) and spatial-wise (`sp`) decompositions.

Secondly, we evaluate the criterion for rank selection with fixed learning method. A method that uniformly reduce the rank across all layers (`uni`) and the energy-based method [Alvarez and Salzmann, 2017; Xu *et al.*, 2020] (`eng`) are compared with ours (`sv`) in Figure 2(c). While `eng` is better than `uni`, `sv` successfully maintains the performance and is the best among them. The difference in selected ranks is illustrated in Figure 2(d). It can be observed that the rank distribution of `sv` is sharper than that of `eng`, which is near-uniform. We consider that `eng` does not fully characterize the importance of each basis, since it ignores the differences of the magnitude of singular values accross layers.

4.2 Comparison with Low-Rank Networks

First, we compare Decomposable-Net with TRP [Xu *et al.*, 2020]. We implemented TRP referencing to their code[2]. TRP has three parameters, namely, the strength τ of trace

[2]https://github.com/yuhuixu1993/Trained-Rank-Pruning

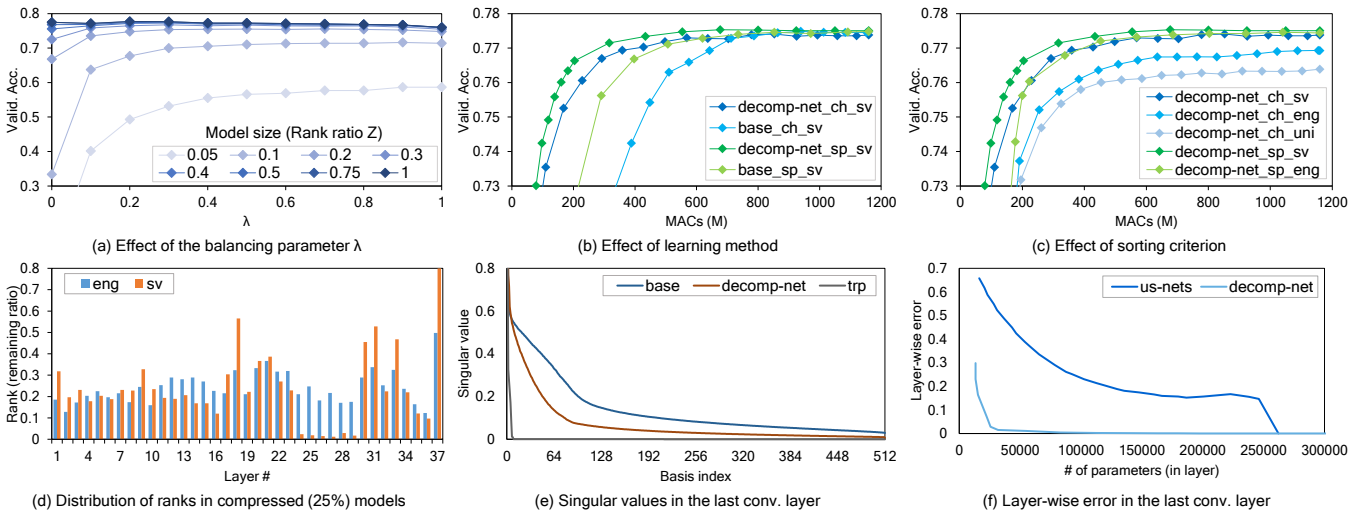

Figure 2: Ablation study with ResNet-34 on CIFAR-100. `base` indicates normal learning as our baseline, likewise `decomp-net` indicates the proposed method. `ch` and `sp` indicate channel- and spatial-wise decompositions, and `uni`, `eng` and `sv` indicate uniform rank selection, energy-based and our singular value-based criterion, respectively.

norm regularization, a threshold e for truncating singular values, and a truncation interval T. TRP optimizes a model of a specific size, but the size of an obtained model depends on those parameters. Thus, we compare several models individually trained with $\tau \in \{0.0001, 0.0003\}$ and $e \in \{0.995, 0.999, 0.9995, 0.9999\}$ for the CIFAR datasets, and $\tau \in \{0.0001, 0.0003, 0.0005\}$ and $e \in \{0.999995, 0.999999\}$ for the ImageNet dataset. We report the results with $T = 1$ since it was the best among $\{1, 2, 4, 8\}$ for all datasets.

The results with spatial-wise decomposition are summarized in Figure 3(a) (the additional results with channel-wise decomposition are illustrated in Figure 6 in the appendix of [Yaguchi *et al.*, 2019]). TRP adopts the energy-based criterion for rank selection, and thus we compress their models with it at several sizes after training. Since TRP truncates small singular values during learning, the resulting ranks in the low-rank network are fixed. Thus, the network does not perform well at sizes other than obtained ranks, as shown in the figure. In contrast, although Decomposable-Net is a single model, it is basically better at almost all sizes. Moreover, our methods do not require exhaustive parameter searches to yield a good model at a specific size, since they require training only one time and changing size. Figure 2(e) shows an example of singular values obtained from each learning method. The singular values with TRP rapidly decrease to zero, because TRP applies trace-norm regularization. The bases with a singular value of zero can be reduced without inducing errors. However, we consider that this limits the representational power of larger models, as we can see the flat lines for TRP in Figure 3(a). On the other hand, our methods reduce singular values in comparison with normal learning, but do not push them to zero. We see that this contributes to improving the performance of low-rank networks while maintaining that of the full-rank network.

Next, we compare with retraining-based methods in which low-rank approximation and fine-tuning are applied after normal learning. Specifically, we adopt a method by [Kim *et al.*, 2016] that directly approximates convolutional tensors with Tucker decomposition, where variational Bayesian matrix factorization (VBMF) [Nakajima *et al.*, 2012] is applied to determine the ranks. We used publicly available code[3] for VBMF. For Tucker decomposition, we apply higher-order orthogonal iteration [De Lathauwer *et al.*, 2000] under given ranks. Then, we perform fine-tuning with a grid search for selecting training parameters. The ranges for the search and the obtained parameters are described in appendix D of [Yaguchi *et al.*, 2019].

The results with the best parameters are shown in Figure 3(b). We additionally evaluate channel- (`ch`) and spatial-wise (`sp`) decompositions by switching them and individually optimizing the parameters. Note that `sp_vbmf_ft` corresponds to the method by [Tai *et al.*, 2016], in which the rank selection method is changed from manual to VBMF. It can be observed that the performance significantly drops from `base` after decomposition, but can be recovered by retraining (dashed vertical lines). This is because the low-rankness is not ensured in the models trained with normal learning. Tucker decomposition is more efficient than other ones and is comparable to ours at a specific size on the CIFAR datasets. However, the performance drops when the size is changed, and retraining is required to recover the performance again. For practical use, such methods are not suited to reconfigure the same model on other devices with different resources. In this paper, we adopt the matrix (channel- and spatial-wise) decompositions, but it is possible to extend our methods with Tucker decomposition since it is a generalization of SVD to higher orders.

[3] https://sites.google.com/site/shinnkj23/downloads

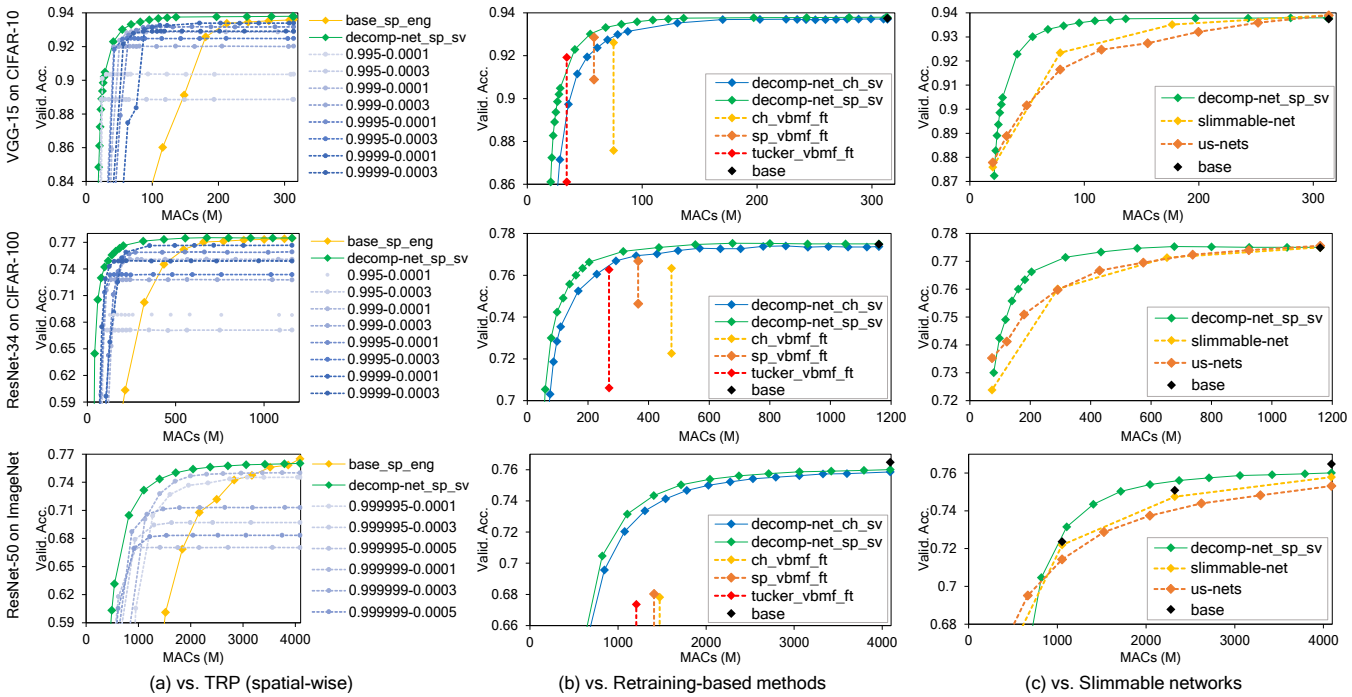

Figure 3: Comparison results: (top) VGG-15 on CIFAR-10, (middle) ResNet-34 on CIFAR-100, and (bottom) ResNet-50 on ImageNet. (a) shows the results vs. TRP [Xu *et al.*, 2020] with spatial-wise decomposition. Dashed lines are the results from TRP, individually trained with different parameters ($e - \tau$). `base_sp_eng` is a baselline, in which normal learning is adopted instead of TRP. (b) shows the results vs. retraining-based methods. `base` is a baseline for the full model. `tucker_vbmf_ft` indicates the method by [Kim *et al.*, 2016]. Dashed vertical lines show the performance improvements of retraining. (c) shows the results vs. slimmable networks [Yu *et al.*, 2019] and US-Nets [Yu and Huang, 2019]. For the ImageNet dataset, three baseline models are trained with diffrent ratios of the number of channels ($1.0\times$, $0.75\times$, and $0.5\times$).

4.3 Comparison with Slimmable Networks

We compare Decomposable-Net with slimmable networks [Yu *et al.*, 2019] and US-Nets [Yu and Huang, 2019]. We implemented them by referencing their code[4]. Slimmable networks are trained at four fixed widths of ratios $\{0.25, 0.5, 0.75, 1\}$. US-Nets are trained with a lower width of ratio 0.25, upper width of ratio 1, and two random widths between them. In-place distillation [Yu and Huang, 2019] is applied to US-Nets. The original loss of slimmable networks is an un-weighted sum of all training losses with different widths. Although it works well with ResNet-50 on the ImageNet dataset, the performance on the CIFAR datasets is unsatisfactory especially when comparing the accuracy (76.6%) of a full-width ResNet-34 with the baseline (77.5%) on the CIFAR-100 dataset. Therefore, we use the average instead of the sum on the CIFAR datasets.

The results are shown in Figure 3(c). For the CIFAR datasets, all methods have performance with the full model comparable to that of the baseline (`base`). However, we can see that the performances of slimmable networks and US-Nets decrease as MACs reduce, which are slightly faster than ours. This can be interpreted as those methods directly reducing the width (i.e., dimensionality) in each layer, and thus losing important directions in the weight space. To better understand this property, we illustrate an example of layer-wise error in Figure 2(f). It can be seen that the error for US-Nets rapidly increases as the number of parameters decreases in the layer, and this increase is not monotonic. In comparison, the error for Decomposable-Net increases slower than that for US-Nets, and monotonicity is ensured as expected. We see this is due to taking the principal components into account for Decomposable-Net, which contributes to maintaining performance over a wide range of model sizes. Moreover, we believe our criterion, which ununiformly select ranks in the network (as we see in Figure 2(d)) contributes too, in contrast to slimmable networks and US-Nets, which reduce the width uniformly across all layers.

For the ImageNet dataset, we can see that the accuracy of Decomposable-Net is maintained for the full model better than under slimmable networks and US-Nets. Specifically, the accuracy of the full model is 76.48% for the baseline and 76.01% for Decomposable-Net, which is better than 75.31% for US-Nets. Moreover, Decomposable-Net achieves better accuracy than the baselines individually trained with $0.75\times$ and $0.5\times$ number of channels, while slimmable networks and US-Nets show lower accuracies than the baselines. It can be also observed that Decomposable-Net is better up to the model with around $0.2\times$MACs, but lower than slimmable networks and US-Nets when the compression rate is extremely high. Our methods use SVD and reduce the

[4]https://github.com/JiahuiYu/slimmable_networks

Method	Acc. (%)	MACs (×)
Baseline ($1.0\times$)	76.48 ± 0.09	1.00 ± 0.000
Baseline ($0.5\times$)	72.37 ± 0.11	0.26 ± 0.000
TRP (`sp`) [Xu *et al.*, 2020]	70.57 ± 0.22	0.28 ± 0.005
Tucker [Kim *et al.*, 2016]	67.36 ± 0.06	0.30 ± 0.005
Slimmable-net [Yu *et al.*, 2019]	72.19 ± 0.12	0.26 ± 0.000
US-Nets [Yu and Huang, 2019]	71.43 ± 0.12	0.26 ± 0.000
Decomposable-Net (`sp`)	$\mathbf{73.15} \pm 0.06$	0.27 ± 0.003

Table 1: Results with ResNet-50 on the ImageNet dataset.

Method	Acc. (%)	Time (ms)	Speed (×)
Baseline	77.49	4.57	1.00
TRP (`sp`) [Xu *et al.*, 2020]	76.65	2.67	1.71
Tucker [Kim *et al.*, 2016]	76.28	1.96	2.33
US-Nets [Yu and Huang, 2019]	77.24	3.27	1.40
Decomposable-Net (`sp`)	77.15	2.02	2.26

Table 2: CPU inference time per image with ResNet-34 on the CIFAR-100 dataset.

bases, which means it does not change the number of inputs and outputs (i.e., the input and output dimensionalities). The number of parameters in each layer is $(m_\ell + n_\ell)r_\ell$, thus decreasing linearly with respect to the rank. US-Nets reduces both input and output dimensionality, meaning the number of parameters decreases at a quadratic rate. This makes it easier for US-Nets to achieve extremely high compression. Therefore, it can be worthwhile to combine our methods with pruning to reduce the number of input or output channels, by incorporating a channel-level sparseness constraint during learning [Alvarez and Salzmann, 2017]. We will consider this for future improvements.

4.4 Remarks

The results on the ImageNet dataset are summarized in Table 1, where we picked up results from the methods compared in this paper, for around $0.25\times$MACs models. To further validate our learning method, we decomposed ResNet-50 with the same ranks (i.e., exactly same MACs) as Decomposable-Net in Table 1. Then, we trained it normally from scratch, and obtained the top-1 accuracy of $71.55 \pm 0.17\%$. Therefore, we see that the performance of our method ($73.15 \pm 0.06\%$) is not easily achieved with the normal training, even by specializing the model to a specific size.

The wall-clock time for training Decomposable-Net with ResNet-50 on the ImageNet dataset was 6.9 days with eight NVIDIA V100 GPUs, whereas TRP (that also applies SVD during training) and US-Nets took 5.7 and 6.9 days, respectively. Therefore, the training cost of Decomposable-Net is not particularly high and comparable to them. Additionally, we measured the actual inference time per image with Intel Core i7-8700K CPU on the CIFAR-100 dataset. We used approximately $0.25\times$MACs model for Decomposable-Net with ResNet-34. For the other methods, we chose the models closest to ours in terms of MACs or accuracy. As shown in Table 2, we can see that Decomposable-Net is as fast as Tucker decomposition with the accuracy comparable to US-Nets.

5 Conclusions

We proposed a novel method that allows DNNs to flexibly change their size after training, by means of the rank of low-rank matrices in each layer. To simultaneously maintain the performance of a full-rank network and improve multiple low-rank networks in a single model, we proposed a novel learning method that explicitly minimizes losses for both of full- and low-rank networks in a differentiable way. In experiments on multiple image-classification tasks using deep CNNs, Decomposable-Net exhibited favorable performance at several sizes of compressed models, relative to that of the low-rank compression methods and slimmable networks.

Acknowledgements

TS was partially supported by JSPS KAKENHI (18K19793, 18H03201, and 20H00576), and JST CREST.

References

[Alvarez and Salzmann, 2017] Jose M Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 856–867. 2017.

[Arora *et al.*, 2018] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning (ICML)*, pages 254–263, 2018.

[Cai *et al.*, 2020] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-All: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020.

[De Lathauwer *et al.*, 2000] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-$(r_1, r_2, \ldots, r_n)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl*, 21(4):1324–1342, 2000.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[Han *et al.*, 2016] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[Howard *et al.*, 2017] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Ioannou *et al.*, 2016] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training CNNs with low-rank filters for efficient image classification. In *International Conference on Learning Representations (ICLR)*, 2016.

[Jaderberg *et al.*, 2014] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *British Machine Vision Conference (BMVC)*, 2014.

[Kim *et al.*, 2016] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *International Conference on Learning Representations (ICLR)*, 2016.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

[Lebedev *et al.*, 2015] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In *International Conference on Learning Representations (ICLR)*, 2015.

[Liu *et al.*, 2017] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2755–2763, 2017.

[Lubana and Dick, 2021] Ekdeep Singh Lubana and Robert P. Dick. A gradient flow framework for analyzing network pruning. In *International Conference on Learning Representations (ICLR)*, 2021.

[Nakajima *et al.*, 2012] Shinichi Nakajima, Ryota Tomioka, Masashi Sugiyama, and S. D. Babacan. Perfect dimensionality recovery by variational Bayesian PCA. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 971–979. 2012.

[Suzuki *et al.*, 2020a] Taiji Suzuki, Hiroshi Abe, Tomoya Murata, Shingo Horiuchi, Kotaro Ito, Tokuma Wachi, So Hirai, Masatoshi Yukishima, and Tomoaki Nishimura. Spectral pruning: Compressing deep neural networks via spectral analysis and its generalization error. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2839–2846, 2020.

[Suzuki *et al.*, 2020b] Taiji Suzuki, Hiroshi Abe, and Tomoaki Nishimura. Compression based bound for non-compressed network: unified generalization error analysis of large compressible deep neural network. In *International Conference on Learning Representations (ICLR)*, 2020.

[Tai *et al.*, 2016] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization. In *International Conference on Learning Representations (ICLR)*, 2016.

[Tan *et al.*, 2019] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2815–2823, 2019.

[Wei and Ma, 2019] Colin Wei and Tengyu Ma. Data-dependent sample complexity of deep neural networks via Lipschitz augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9725–9736. 2019.

[Xu *et al.*, 2020] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. TRP: Trained rank pruning for efficient deep neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 977–983, 2020.

[Yaguchi *et al.*, 2019] Atsushi Yaguchi, Taiji Suzuki, Shuhei Nitta, Yukinobu Sakata, and Akiyuki Tanizawa. Decomposable-Net: Scalable low-rank compression for neural networks. *arXiv preprint arXiv:1910.13141*, 2019.

[Yang *et al.*, 2020] Taojiannan Yang, Sijie Zhu, Chen Chen, Yan Shen, Mi Zhang, and Andrew Willis. MutualNet: Adaptive convnet via mutual learning from network width and resolution. In *European Conference on Computer Vision (ECCV)*, pages 299–315, 2020.

[Yu and Huang, 2019] Jiahui Yu and Thomas Huang. Universally slimmable networks and improved training techniques. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1803–1811, 2019.

[Yu *et al.*, 2019] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[Yu *et al.*, 2020] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. BigNAS: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision (ECCV)*, pages 702–717, 2020.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016.

[Zagoruyko, 2015] Sergey Zagoruyko. 92.45% on CIFAR-10 in Torch. http://torch.ch/blog/2015/07/30/cifar.html, 2015. Accessed: 2017-11-21.

[Zhang *et al.*, 2016] 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 (TPAMI)*, 38(10):1943–1955, 2016.

[Zhang *et al.*, 2019] Linfeng Zhang, Zhanhong Tan, Jiebo Song, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. SCAN: A scalable neural networks framework towards compact and efficient models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4027–4036. 2019.