Understanding Generalization in Deep Learning via Tensor Methods

Jingling Li[1,3] **Yanchao Sun**[1] **Jiahao Su**[4] **Taiji Suzuki**[2,3] **Furong Huang**[1]

[1]Department of Computer Science, University of Maryland, College Park
[2]Graduate School of Information Science and Technology, The University of Tokyo
[3]Center for Advanced Intelligence Project, RIKEN
[4]Department of Electrical and Computer Engineering, University of Maryland, College Park

Abstract

Deep neural networks generalize well on unseen data though the number of parameters often far exceeds the number of training examples. Recently proposed complexity measures have provided insights to understanding the generalizability in neural networks from perspectives of PAC-Bayes, robustness, overparametrization, compression and so on. In this work, we advance the understanding of the relations between the network's architecture and its generalizability from the compression perspective. Using tensor analysis, we propose a series of intuitive, data-dependent and easily-measurable properties that tightly characterize the compressibility and generalizability of neural networks; thus, in practice, our generalization bound outperforms the previous compression-based ones, especially for neural networks using tensors as their weight kernels (e.g. CNNs). Moreover, these intuitive measurements provide further insights into designing neural network architectures with properties favorable for better/guaranteed generalizability. Our experimental results demonstrate that through the proposed measurable properties, our generalization error bound matches the trend of the test error well. Our theoretical analysis further provides justifications for the empirical success and limitations of some widely-used tensor-based compression approaches. We also discover the improvements to the compressibility and robustness of current neural networks when incorporating tensor operations via our proposed layer-wise structure.

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

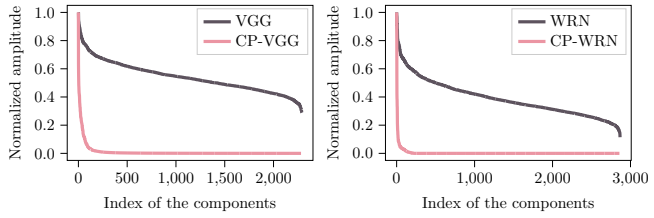
1 Introduction

Deep neural networks recently have made major breakthroughs in solving many difficult learning problems, especially in image classification (Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016; Zagoruyko and Komodakis, 2016) and object recognition (Krizhevsky et al., 2012; Sermanet et al., 2013; Simonyan and Zisserman, 2014; Zeiler and Fergus, 2014). The success of deep neural networks depends on the high expressive power and the ability to generalize. The high expressive power has been demonstrated empirically (He et al., 2016; Zagoruyko and Komodakis, 2016) and theoretically (Hornik et al., 1989; Mhaskar and Poggio, 2016). Yet, fundamental questions on why deep neural networks generalize and what enables their generalizability remain unsettled.

A recent work by Arora et al. (2018) characterizes the generalizability of a neural network from a compression perspective — the capacity of the network is characterized through its compressed version. The compression algorithm in Arora et al. (2018) is based on random projection: each weight matrix of the compressed network are represented by a linear combination of basis matrices with entries i.i.d. sampled from ± 1. The effective number of parameters in the weight matrix is the number of coefficients in this linear combination obtained via projection — the inner product between the original weight matrix and these basis matrices. Though the idea of using compression in deriving the generalization bounds is novel, the compression scheme in Arora et al. (2018) could be made more practical since (1) the cost of forwarding pass in the compressed network still remains the same as the cost in the original one, even though the effective number of parameters to represent the original weight matrices decreases; (2) storing these random projection matrices could require more spaces than storing the original set of parameters. We propose a new theoretical analysis based on a more practical, well-developed, and principled compression scheme using tensor methods. Besides, we use tensor analysis to derive a much tighter bound for the layer-wise error

propagation by exploiting additional structures in the weight tensors of neural networks, which as a result significantly tightens the generalization error bound in Arora et al. (2018).

Our approach aims to characterize the network's compressibility by measuring the low-rankness of the weight kernels. Existing compression methods in (Jaderberg et al., 2014; Denton et al., 2014; Lebedev et al., 2014; Kim et al., 2015; Garipov et al., 2016; Wang et al., 2018; Su et al., 2018) implement low-rank approximations by performing matrix/tensor decomposition on weight matrices/kernels of well-trained models. However, the layers of SOTA networks, such as VGG (Simonyan and Zisserman, 2014) and WRN (Zagoruyko and Komodakis, 2016), are not necessarily low-rank: we apply CP-tensor decompositions (Kolda and Bader, 2009; Anandkumar et al., 2014b; Huang et al., 2015; Li and Huang, 2018) to the weight tensors of well-trained VGG-16 and WRN-28-10, and *the amplitudes of the components from the CP decomposition* (a.k.a ***CP spectrum***) are demonstrated by the brown curves in Figure 1, which indicate that the layers of these pre-trained networks are not low-rank. Therefore a straightforward compression of the network cannot be easily achieved and computationally expensive fine tuning is often needed.



(a) VGG16 (layer 13) **(b)** WRN-28-10 (layer 28)

Figure 1: CP spectrum comparison (CP-VGG and CP-WRN are neural networks with CP layers).

To overcome this limitation, we propose a layer-wise structure design, *CP Layer* (CPL), by incorporating the variants of CP decompositions in (Jaderberg et al., 2014; Kossaifi et al., 2017; Howard et al., 2017). CPL re-parametrizes the weight tensors such that a *Polyadic form* (CP form) (Kolda and Bader, 2009) can be easily learned in an end-to-end fashion.

We demonstrate that empirically, CPL allows the network to learn a low-rank structure more easily, and thus helps with compression. For example, from the pink curves in Figure 1, we see that neural networks with CPL have a spiky CP spectrum, which is an indication of low-rankness. We rigorously prove that this low-rankness in return leads to a tighter generalization bound. Moreover, we are the first to provide theoretical guarantees for the usage of CP decomposition in deep neural networks in terms of compressibility and generalizability.

Definition 1.1 (Proposed Architecture Layer). A CP Layer (CPL) with width R consists of R set of parameters $\left\{\lambda^{(r)}, \left\{\boldsymbol{v}_j^{(r)}\right\}_{j=1}^{N}\right\}_{r=1}^{R}$ where $\boldsymbol{v}_j^{(r)}$ is a vector in $\mathbb{R}^{d_j}$ with unit norm. The weight kernel of this CPL is a N-order tensor defined as $\mathcal{K} := \sum_{r=1}^{R} \lambda^{(r)} \boldsymbol{v}_1^{(r)} \otimes \cdots \otimes \boldsymbol{v}_N^{(r)}$, where $\otimes$ denotes the vector outer-product (tensor product) defined in Appendix B.9) [1]. Note that $\mathcal{K} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$.

Remark. CPL allows for flexible choices of the structures since the number of components R is a tunable hyper-parameter that controls the number of parameters in CPL. The CP spectrum of this layer is denoted by $\{\lambda^{(r)}\}_{r=1}^{R}$ in a descending order. The size of the weight kernel is $d_0 \times d_1 \times \cdots \times d_N$, while the number of parameters in CPL is $(d_0 + d_1 + \cdots + d_N + 1) \times R$.

In contrast with existing works which apply CP decomposition to each layer of a reference network, no CP decomposition is needed since the components are explicitly stored as model parameters so that they can be learned from scratch via back-propagation. Moreover, compression in CP layers is natural – simply picking the top $\hat{R}$ components to retain and pruning out the rest of them. Thus, the compression procedure using CPL does not require any costly fine-tuning while existing works on tensor-based compression may use hundreds of epochs for fine-tuning.

We further propose a series of simple, intuitive, data-dependent and easily-measurable *properties* to measure the low-rankness in current neural networks. These properties not only guide the selection of the number of components to generate a good compression, but also tighten the bound of the layer-wise error propagation via tensor analysis. The proposed properties

- characterize the compressibility of the neural network, i.e., how much the original network can be compressed without compromising the performance on a training dataset more than certain range.
- characterize the generalizability of the compressed network, i.e. tell if a neural network is trained using normal data or corrupted data.

In our theoretical analysis, we derive generalization error bounds for neural networks with CP layers, which take both the input distribution and the compressibility of the network into account. We present a rigorous proof showing the connection of our proposed properties to the generalization error of a network. We will see in experiment section that our proposed bound is very effective at predicting the generalization error.

[1] The $(i_1, i_2, \ldots, i_N)^{\text{th}}$ element of the weight kernel is $\sum_{r=1}^{R} \lambda^{(r)} \boldsymbol{v}_1^{(r)}(i_1) \times \cdots \times \boldsymbol{v}_N^{(r)}(i_N)$.

Notice that, in this paper, the Polyadic form is chosen simply as a demonstration on how tensor methods could be used to improve the analysis of generalization bounds of deep neural networks. Therefore, follow-ups works could potentially analyze the effects of other tensor decomposition methods using our theoretical framework.

Summary of Contributions

1. **Better generalization bound of practical use.** We verify that our generalization bounds can be used to guide the training of neural networks, since the calculated bound matches the trend of the test error on unseen data during the training process as shown in Figure 2b. Moreover, we demonstrate that our generalization bound is in practice tighter than the bound proposed by (Arora et al., 2018) as shown in Figure 2a and Table 4. Notice that the generalization bound in (Arora et al., 2018) is already orders of magnitude better than previous norm-based or compression based bounds.
2. **Intuitive measurements of compressibility and generalizability.** We propose a set of properties to characterize the low-rankness in the weight tensors of neural networks in Section 4.2. Our theoretical analysis connects the measured low-rankness with the generalizability of the model, and such connections are verified in Figure 3.
3. **First theoretical guarantee** on the generalizability and robustness for neural network architectures that allow fast and real time predictions on devices with limited memory (e.g. the architecture designs proposed in (Jaderberg et al., 2014; Kossaifi et al., 2017; Howard et al., 2017), which uses variants of the Polyadic form).
4. **Practical improvements.** We demonstrate that pruning out the smaller components of CP decomposition in CP layers roughly preserves the test performance without computationally expensive fine tuning (see Section 5.3 and Table 5) as our proposed layer-wise structure is easily compressible. Moreover, we discover that incorporating tensor operations via CPL reduces the generalization error of some well-known neural network architectures, and further improves the robustness of SOTA methods for learning under noisy labels (see Table 2, Table 3, Figure 5, and Figure 6).

2 Related Works

Existing Metrics to Characterizing Generalization. Classical and recent works have analyzed the generalizability of neural networks from different perspective such as VC-dimension (Bartlett et al., 1999; Harvey et al., 2017), sharpness of the solution (Keskar et al., 2016), robustness of the algorithm (Xu and Mannor, 2012), stability and robustness of the model (Hardt et al., 2016; Kuzborskij and Lampert, 2018; Gonen and Shalev-Shwartz, 2017; Sokolic et al., 2016) and over-parameterization (Neyshabur et al., 2018; Du and Lee, 2018), or using various approaches such as PAC-Bayes theory (McAllester, 1999b,a; Langford and Caruana, 2002; Neyshabur et al., 2015b, 2017b; Dziugaite and Roy, 2017; Golowich et al., 2018), norm-based analysis (Bartlett and Mendelson, 2002; Neyshabur et al., 2015a; Kawaguchi et al., 2017; Golowich et al., 2017), compression based approach (Arora et al., 2018), and combinations of the above approaches (Neyshabur et al., 2017b,a; Bartlett et al., 2017; Zhou et al., 2018) (see (Jakubovitz et al., 2018) for a complete survey). While these works provide deep theoretical insights to the understanding of the generalizability in neural networks, they did not provide practical techniques to improve generalization.

For the progress on non-vacuous generalization bounds, Dziugaite and Roy (2017) use non-convex optimization and PAC-Bayesian analysis to obtain a non-vacuous sample bound on MNIST, and Zhou et al. (2018) use a PAC-Bayesian compression approach to obtain non-vacuous generalization bounds on both MNIST and ImageNet via smart choices of the prior. While being creative, both bounds are less intuitive and provide little insight into what properties are favorable for networks to have better generalizability. In addition, the tensor-based compression methods are complementary to the compression approach used in (Zhou et al., 2018), which combines pruning, quantization and huffman coding (Han et al., 2015); the tensor-based compression methods can be combined with the approaches used in (Han et al., 2015) to potentially tighten the generalization bound obtained in (Zhou et al., 2018).

Improving generalization in practice. Authors of (Neyshabur et al., 2015a) proposed an optimization method PATH-SGD which improves the generalization performance empirically. While (Neyshabur et al., 2015a) focuses on the optimization approach, we provide a different practical approach that helps the understanding of the relations between the network architecture and its generalization ability.

Comparison with Arora et al. (Arora et al., 2018). Besides practical improvements of generalization error, our work improves the results obtained by (Arora et al., 2018): 1) we provide a tightened layer-wise analysis using tensor method to directly bound the operator norm of the weight kernel (e.g. Lemma C.5 and Lemma C.8). The interlayer properties introduced by (Arora et al., 2018) are orthogonal to our proposed layer-wise properties and they can be well-combined; 2) in practice, our bound outperforms that of (Arora et al.,

2018) in terms of the achieved degree of compression (detailed discussions in Section 5.2 and Section A.2); 3) for fully connected (FC) neural networks, our proposed reshaping factor (definition E.2) further tightens the generalization bound as long as the inputs to the FC layers have some low-rank structures; 4) we extend our theoretical analysis to neural networks with skip connections, while the theoretical analysis in Arora et al. (2018) only applies to FC and CNN.

Comparison with existing CP decomposition for network compression. While CP decomposition has been commonly used in neural network compression (Denton et al., 2014; Lebedev et al., 2014; Kossaifi et al., 2017), our proposed compression method is very different from theirs. First, the the tensor contraction layer Kossaifi et al. (2017) is a special case of our CPL for FC layers when we set the number of components to be 1. Second, the number of components in our proposed CPL can be arbitrarily large (as it is a tunable hyper-parameter), while the number of components of layers in (Denton et al., 2014; Lebedev et al., 2014; Kossaifi et al., 2017) are determined by the compression ratio. Third, no tensor decomposition is needed for evaluating the generalizability and compressing neural networks with CP layers as the components from the CP decomposition are already stored as model parameters. Moreover, as the smaller components in CPL are pruned during the compression, the performance of the compressed neural net is often preserved and thus no expensive fine tuning is required (see Table 5). The depthwise-separable convolution used in MobileNet (Howard et al., 2017) is a specific implementation of CPL; thus, our theoretical analysis can provide generalization guarantees for the MobileNet architecture.

3 Notations and Preliminaries

In this paper, we use S to denote the set of training samples drawn from a distribution D with $|S| = m$. Let n denote the number of layers in a given neural network, and superscripts of form $^{(k)}$ denote properties related to the k^{th} layer. We put "CP" in front of a network's name to denote such a network with CP layers (e.g. CP-VGG denotes a VGG with CP layers). For any positive integer n, let $[n] := \{1, 2, ..., n\}$. Let $|a|$ denote the absolute value of a scalar a. Given a vector $\boldsymbol{a} \in \mathbb{R}^d$, a matrix $\boldsymbol{A} \in \mathbb{R}^{d\times k}$, and a tensor $\mathcal{A} \in \mathbb{R}^{d_1\times d_2\times d_3}$, their norms are defined as follows: (1) **Vector norm:** $\|\boldsymbol{a}\|$ denotes the ℓ_2 norm. (2) **Matrix norms:** Let $\|\boldsymbol{A}\|_*$ denote its nuclear norm, $\|\boldsymbol{A}\|_{\mathsf{F}}$ denote its Frobenius norm, and $\|\boldsymbol{A}\|$ denote its operator norm (spectral norm), where $\sigma_i(\boldsymbol{A})$ denotes the i^{th} largest singular value of $\boldsymbol{A}$. (3) **Tensor norms:** Let $\|\mathcal{A}\| = \max_{x\in\mathbb{R}^{d_1}, y\in\mathbb{R}^{d_2}, z\in\mathbb{R}^{d_3}} \frac{|A(x,y,z)|}{\|x\|\|y\|\|z\|}$ denote its operator norm, and $\|\mathcal{A}\|_{\mathsf{F}}$ its Frobenius norm. Moreover, we use $\otimes$ to denote the **outer product operator**, and $*$ to denote **the convolution operator**. We use $\mathcal{F}_m$ to denote m**-dimensional discrete Fourier transform**, and use tilde symbols to denote tensors after DFT (e.g. $\tilde{\mathcal{T}} = \mathcal{F}_m(\mathcal{T})$). A **Polyadic decomposition (CP decomposition)** (Kruskal, 1989; Kolda and Bader, 2009) of a N-order tensor $\mathcal{K} \in \mathbb{R}^{d_1\times d_2\times\cdots\times d_N}$ is a linear combination of rank-one tensors that is equal to $\mathcal{K}$: $\mathcal{K} = \sum_{r=1}^{R} \lambda^{(r)} \boldsymbol{v}_1^{(r)} \otimes \cdots \otimes \boldsymbol{v}_N^{(r)}$ where $\forall r \in [R], \forall j \in [N], \left\|\boldsymbol{v}_j^{(r)}\right\| = 1$. **Margin loss Arora et al. (2018):** we use $L_\gamma(\mathbb{M})$ and $\hat{L}_\gamma(\mathbb{M})$ to denote the expected and empirical margin loss of a neural network $\mathbb{M}$ with respect to a margin $\gamma \geq 0$. The expected margin loss of a neural network $\mathbb{M}$ is defined as $L_\gamma(\mathbb{M}) := \mathbb{P}_{(\boldsymbol{x},y)\in D}\left[\mathbb{M}(\boldsymbol{x})[y] \leq \gamma + \max_{i\neq y} \mathbb{M}(\boldsymbol{x})[i]\right]$.

4 CNNs with CPL: Compressibility and Generalization

In this section, we derive the generalization bound for a convolutional neural network (denoted as $\mathbb{M}$) using tensor methods and standard Fourier analysis. The complete proof is in Appendix Section D. For simplicity, we assume that there is no pooling layer (e.g. max pooling) in $\mathbb{M}$ since adding pooling layer will only lead to a smaller generalization bound (the perturbation error in our analysis decreases with the presence of pooling layers). The derived generalization bound can be directly extended to various neural network architectures (e.g. neural networks with pooling layers, and neural networks with batch normalization). The generalization bounds for fully connected neural networks and neural networks with skip connections are presented in Appendix Section E.4 and F.3 respectively.

4.1 Compression of a CNN with CPL

We first illustrate how to compress any given CNN $\mathbb{M}$ by presenting a compression algorithm (Algorithm 1). We will see that this compression algorithm guarantees a good estimation of the generalization bound for the compressed network $\hat{\mathbb{M}}$.

Original CNN $\mathbb{M}$ is of n layers with ReLU activation, its k^{th} layer weight tensor $\mathcal{M}^{(k)}$ is a 4^{th} order tensor of size = *# of input channel* $s^{(k)}$ × *# of output channel* $o^{(k)}$ × kernel height $k_x^{(k)}$ × kernel width $k_y^{(k)}$. Let the 3^{rd} order tensor $\mathcal{X}^{(k)} \in \mathbb{R}^{H^{(k)}\times W^{(k)}\times s^{(k)}}$ denote the input to the k^{th} layer, and $\mathcal{Y}^{(k)} \in \mathbb{R}^{H^{(k)}\times W^{(k)}\times o^{(k)}}$ denote the output of the k^{th} layer before activation. Therefore $\mathcal{X}^{(k)} = \mathsf{ReLU}\left(\mathcal{Y}^{(k-1)}\right)$. We use i to denote the index of input channels, and j to denote the index of output channels. We further use f and g to denote the indices of width and height in the frequency domain.

Proposition 4.1 (**Polyadic Form of original CNN** $\mathbb{M}$)**.** For each layer k, the weight tensor $\mathcal{M}^{(k)}$ has a

Polyadic form with number of components $R^{(k)} \leq \min\{s^{(k)}o^{(k)}, s^{(k)}k_x^{(k)}k_y^{(k)}, o^{(k)}k_x^{(k)}k_y^{(k)}\}$ (Kolda and Bader, 2009): $\mathcal{M}^{(k)} = \sum_{r=1}^{R^{(k)}} \lambda_r^{(k)} \boldsymbol{a}_r^{(k)} \otimes \boldsymbol{b}_r^{(k)} \otimes \boldsymbol{C}_r^{(k)}$, where the CP-spectrum is in a descending order, i.e., $\lambda_1^{(k)} \geq \lambda_2^{(k)} \geq \cdots \geq \lambda_{R^{(k)}}^{(k)}$. All $\boldsymbol{a}_r^{(k)}, \boldsymbol{b}_r^{(k)}$ are unit vectors in $\mathbb{R}^{s^{(k)}}$ and $\mathbb{R}^{o^{(k)}}$ respectively, and $\boldsymbol{C}_r^{(k)}$ is a matrix in $\mathbb{R}^{k_x^{(k)} \times k_y^{(k)}}$ with $\|\boldsymbol{C}_r^{(k)}\|_{\mathsf{F}} = 1$. The $R^{(k)}$ required for the Polyadic Form is called tensor rank.

Transform original CNN to a CNN with CP layers. By Proposition 4.1, each weight tensor $\mathcal{M}^{(k)}$ in $\mathbb{M}$ can be represented in a Polyadic form (CP form) and thus is transformed to a CPL. The total number of parameters in CPL is $R^{(k)} \times (s^{(k)} + o^{(k)} + k_x^{(k)}k_y^{(k)} + 1)$. Thus, a smaller $R^{(k)}$ leads to fewer number of effective parameters and indicates more compression.

Compress Original CNN $\mathbb{M}$ to $\hat{\mathbb{M}}$. We illustrate the compression procedure in Algorithm 1. Feeding a CNN $\mathbb{M}$ to the compression algorithm, we obtain a compressed CNN $\hat{\mathbb{M}}$, where for each layer k, the weight tensor in $\hat{\mathbb{M}}$ is $\hat{\mathcal{M}}^{(k)} = \sum_{r=1}^{\hat{R}^{(k)}} \lambda_r^{(k)} \boldsymbol{a}_r^{(k)} \otimes \boldsymbol{b}_r^{(k)} \otimes \boldsymbol{c}_r^{(k)}$ for some $\hat{R}^{(k)} \leq R^{(k)}$. Similarly, we use $\hat{\mathcal{X}}^{(k)}$ to denote the input tensor of the k^{th} layer in $\hat{\mathbb{M}}$ and $\hat{\mathcal{Y}}^{(k)}$ to denote the output tensor of the k^{th} layer in $\hat{\mathbb{M}}$ before activation. Therefore $\hat{\mathcal{X}}^{(k)} = \mathsf{ReLU}\left(\hat{\mathcal{Y}}^{(k-1)}\right)$. Notice that $\hat{\mathcal{X}}^{(k)}, \hat{\mathcal{Y}}^{(k)}$ are of the same shapes as $\mathcal{X}^{(k)}, \mathcal{Y}^{(k)}$ respectively and $\mathcal{X}^{(1)} = \hat{\mathcal{X}}^{(1)}$ since the input data to both networks $\mathbb{M}$ and $\hat{\mathbb{M}}$ is the same.

The compression Algorithm 1 is designed to compress any CNN, and therefore requires applying explicit CP decompositions to the weight tensors of traditional CNNs (the step 3 in Algorithm 1). However, for a CNN with CP layers, these CP components are already stored as weight parameters in our CPL structure, and thus are known to the compression algorithm in advance. Therefore, no tensor decomposition is needed when compressing CNNs with CPL as we can prune out the components with smaller amplitudes directly.

4.2 Characterizing Compressibility of CNN with CPL: Network Properties

In this section, we propose the following layer-wise properties that can be evaluated based on the training data S: *tensorization factor (TF)*, *tensor noise bound (TNB)*, and *layer cushion (LC)* (Arora et al., 2018). These proposed properties are very effective at characterizing the compressibility of a neural network. As Algorithm 1's sub-procedure FBRC selects a set of number of components $\{\hat{R}^{(k)}\}_{k=1}^n$ to obtain a compressed network $\hat{\mathbb{M}}$ whose output is similar to that of the original network (i.e., $\left\|\mathbb{M}(\mathcal{X}) - \hat{\mathbb{M}}(\mathcal{X})\right\|_{\mathsf{F}} \leq \epsilon \|\mathbb{M}(\mathcal{X})\|_{\mathsf{F}}$ for any input $\mathcal{X} \in S$), our proposed properties will assist the selections of $\{\hat{R}^{(k)}\}_{k=1}^n$ to guarantee that Algorithm 1 returns a "good" compressed network.

Algorithm 1 Compression of Convolutional Neural Networks

$^{\square}$FBRC (in Appendix G) calculates a set of number of components $\{\hat{R}^{(k)}\}_{k=1}^n$ for the compressed network such that $\left\|\mathbb{M}(\mathcal{X}) - \hat{\mathbb{M}}(\mathcal{X})\right\|_{\mathsf{F}} \leq \epsilon \|\mathbb{M}(\mathcal{X})\|_{\mathsf{F}}$ holds for any given ϵ and for any input $\mathcal{X}$ in the training dataset S.

$^{\triangle}$CNN-Project (in Appendix G) takes a given set of number of components $\{\hat{R}^{(k)}\}_{k=1}^n$ and returns a compressed network $\hat{\mathbb{M}}$ by pruning out the smaller components in the CP spectrum of the weight tensors of $\mathbb{M}$.

More intuitions of the sub-procedures FBRC and CNN-Project are described in Section 4.2 and Appendix G.

Input: A CNN $\mathbb{M}$ of n layers and a margin γ

Output: A compressed $\hat{\mathbb{M}}$ whose expected error $L_0(\hat{\mathbb{M}}) \leq \hat{L}_\gamma(\mathbb{M}) + \tilde{O}\left(\sqrt{\frac{\sum_{k=1}^n \hat{R}^{(k)}(s^{(k)}+o^{(k)}+k_x^{(k)}\times k_y^{(k)}+1)}{m}}\right)$

1: Calculate all layer cushions $\{\zeta^{(k)}\}_{k=1}^n$ based on definition 4.4

2: Pick $R^{(k)} = \min\{s^{(k)}o^{(k)}, s^{(k)}k_x^{(k)}k_y^{(k)}, o^{(k)}k_x^{(k)}k_y^{(k)}\}$ for each layer k

3: If $\mathbb{M}$ does not have CPL, apply a CP-decomposition to the weight tensor of each layer k

4: Set the perturbation parameter $\epsilon := \frac{\gamma}{2\max_{\mathcal{X}}\|\mathbb{M}(\mathcal{X})\|_{\mathsf{F}}}$

5: Compute number of components needed for each layer of the compressed network $\{\hat{R}^{(k)}\}_{k=1}^n \leftarrow$ FBRC$^{\square}\left(\{\mathcal{M}^{(k)}\}_{k=1}^n, \{R^{(k)}\}_{k=1}^n, \{\zeta^{(k)}\}_{k=1}^n, \epsilon\right)$

6: $\hat{\mathbb{M}} \leftarrow$ CNN-Project$^{\triangle}\left(\mathbb{M}, \{\hat{R}^{(k)}\}_{i=1}^n\right)$

7: Return the compressed convolutional neural network $\hat{\mathbb{M}}$

Definition 4.2. [tensorization factor $t_j^{(k)}$] The *tensorization factors* $\left\{t_j^{(k)}\right\}_{j=1}^{R^{(k)}}$ of the k^{th} layer is defined as

$$t_j^{(k)} := \max_{f,g} \sum_{r=1}^{j} \left|\lambda_r^{(k)}\right| \left|\tilde{C}_r^{(f,g)}\right| \tag{1}$$

where $\lambda_r^{(k)}$ is the r^{th} largest value in the CP spectrum of the weight tensor $\mathcal{M}^{(k)}$ and $\tilde{C}_r^{(f,g)}$ denotes the amplitude at the frequency (f, g).

Remark. The tensorization factor characterizes both the generalizability and the expressive power of a given network. For a fixed j, a smaller tensorization factor indicates the original network is more compressible and thus has a smaller generalization bound. However, a smaller tensorization factor may also indicate that the given network do not possess enough expressive power. Thus, during the compression of a neural network with good generalizability, we need to find a "good" j that generates a tensorization factor demonstrating the balance between a small generalization gap

and high expressive power.

Definition 4.3. [tensor noise bound $\xi_j^{(k)}$] The *tensor noise bound* $\left\{\xi_j^{(k)}\right\}_{j=1}^{R^{(k)}}$ of the k^{th} layer measures the amplitudes of the remaining components after pruning the ones with amplitudes smaller than the $\lambda_j^{(k)}$:

$$\xi_j^{(k)} := \max_{f,g} \sum_{r=j+1}^{R^{(k)}} \left|\lambda_r^{(k)}\right| \left|\tilde{C}_r^{(f,g)}\right| \tag{2}$$

Remark. For a fixed j, a smaller tensor noise bound indicates the original neural network's weight tensor is more low-rank and thus more compressible.

Definition 4.4. [layer cushion $\zeta^{(k)}$] As introduced in Arora et al. (2018), the layer cushion of the k^{th} layer is defined to be the largest value $\zeta^{(k)}$ such that for any $\mathcal{X}^{(k)} \in S$,

$$\zeta^{(k)} \left(\left\|\mathcal{M}^{(k)}\right\|_{\mathsf{F}} \Big/ \sqrt{H^{(k)}W^{(k)}}\right) \left\|\mathcal{X}^{(k)}\right\|_{\mathsf{F}} \leq \left\|\mathcal{M}^{(k+1)}\right\|_{\mathsf{F}} \tag{3}$$

Following Arora et al. (2018), layer cushion considers how much the output tensor $\left\|\mathcal{M}^{(k+1)}\right\|_{\mathsf{F}}$ grows w.r.t. the weight tensor $\left\|\mathcal{M}^{(k)}\right\|_{\mathsf{F}}$ and the input $\left\|\mathcal{X}^{(k)}\right\|_{\mathsf{F}}$.

Remark. As introduced in Arora et al. (2018), the layer cushion considers how much smaller the output $\left\|\mathcal{X}^{(k+1)}\right\|_{\mathsf{F}}$ of the k^{th} layer (after activation) compares with the product between the weight tensor $\left\|\mathcal{M}^{(k)}\right\|_{\mathsf{F}}$ and the input $\left\|\mathcal{X}^{(k)}\right\|_{\mathsf{F}}$. Note that our layer cushion can be larger than 1 if models use batchnorm, and larger layer cushions will render smaller generalization bounds as also shown in (Arora et al., 2018).

Our proposed properties, orthogonal to the interlayer properties introduced in (Arora et al., 2018), provide better measurements of the compressibility in each individual convolutional layer via the use of tensor analysis and Fourier analysis, and thus lead to a tighter bound of the layer-wise error propagation.

4.3 Generalization Guarantee of CNNs

Based on Algorithm 1 and our proposed properties in section 4.2, we obtain a generalization bound for the compressed convolutional neural network $\hat{\mathbb{M}}$ and, in section 5, we will evaluate this bound explicitly.

Theorem 4.5 (Main Theorem). For any convolutional neural network $\mathbb{M}$ with n layers, Algorithm 1 generates a compressed CNN $\hat{\mathbb{M}}$ such that with high probability, the expected error $L_0(\hat{\mathbb{M}})$ is bounded by the empirical margin loss $\hat{L}_\gamma(\mathbb{M})$ (for any margin $\gamma \geq 0$) and a complexity term defined as follows

$$L_0(\hat{\mathbb{M}}) \leq \hat{L}_\gamma(\mathbb{M}) + \tilde{O}\left(\sqrt{\frac{\sum_{k=1}^n \hat{R}^{(k)}(s^{(k)} + o^{(k)} + k_x^{(k)} k_y^{(k)} + 1)}{m}}\right) \tag{4}$$

given that for all layer k, the number of components $\hat{R}^{(k)}$ in the compressed network satisfies that

$$\hat{R}^{(k)} = \min\left\{j \in [R^{(k)}] | \xi_j^{(k)} \Pi_{i=k+1}^n t_j^{(i)} \leq C\right\} \tag{5}$$

$$\text{with } C = \frac{\gamma}{2n \max_{\mathcal{X}\in S} \|\mathbb{M}(\mathcal{X})\|_{\mathsf{F}}} \Pi_{i=k}^n \zeta^{(i)} \left\|\mathcal{M}^{(i)}\right\|_{\mathsf{F}}$$

where $t_j^{(k)}$, $\xi_j^{(k)}$ and $\zeta^{(k)}$ are data dependent measurable properties — tensorization factor, tensor noise bound, and layer cushion of the k^{th} layer in definitions 4.2, 4.3 and 4.4 respectively.

Remark. How well the compressed neural network approximates the original network is related to the choice of $\hat{R}^{(k)}$. Inside equation (5), C is some value independent of the choice of j in the inequality. Therefore, the number of components for the k^{th} layer in the compressed network, $\hat{R}^{(k)}$, is the smallest $j \in [R^{(k)}]$ such that the inequality $\xi_j^{(k)} \Pi_{i=k+1}^n t_j^{(i)} \leq C$ holds. Hence, smaller tensorization factors and tensor noise bounds will make the LHS smaller, and larger layer cushions will make the RHS, C, larger. As a result, if the above inequality for each layer can be satisfied by a smaller j, the obtained generalization bound will be tighter as we can obtain a smaller $\hat{R}^{(k)}$.

Analysis of generalization bounds in Theorem 4.5: This proposed generalization error bound is proportional to the number of components in the CP layers of the compressed neural network. Therefore, when the original neural network is highly compressible or very low-rank, the number of components needed will be lower, which thus renders a smaller generalization error bound.

The proof of Theorem 4.5 is in Appendix Section D), and the proof sketch is as follows.

Proof sketch of Theorem 4.5: We first establish that the difference of the outputs between the compressed CNN $\hat{\mathbb{M}}$ and the original CNN $\mathbb{M}$ is bounded by $\frac{\gamma}{2\max_{\mathcal{X}} \|\mathbb{M}(\mathcal{X})\|_{\mathsf{F}}}$ using Lemma D.5. Then we show the covering number of the compressed network $\hat{\mathbb{M}}$ is $\tilde{O}(d)$ via Lemma D.7, where d denotes the total number of parameters in the compressed network. Bounding the covering number of CNNs with CPL to be of order $\tilde{O}(d)$ is non-trivial as we need careful handlings of the error propagations to avoid a dependence on the product of number of components. After bounding the

Table 1: Comparison of the training and test accuracies between neural networks (NNs) with CPL (CP-VGG-16, CP-WRN-28-10) and traditional NNs (VGG-16. WRN-28-10) on CIFAR10 dataset.

Acc. / Architect. / Dataset		VGG-16		WRN-28-10	
		with CPL	without CPL	with CPL	without CPL
CIFAR10	Training	100%	100%	100%	100%
	Test	93.68%	92.64%†	95.09%	95.83%*
CIFAR100	Training	100%	100%	100%	100%
	Test	71.8%	70.84%‡	76.36%	79.5%* [2]

Table 2: Test accuracy on CIFAR10 with various label corruptions rates (CR).

Network / CR		0.2	0.4	0.6	0.8
CIFAR10	VGG-16	68.76	44.26	24.89	13.21
	CP-VGG-16	**71.09**	**51.76**	**35.60**	**20.06**
CIFAR100	VGG-16	50.94	30.46	13.6	1.11
	CP-VGG-16	**54.51**	**34.13**	**15.23**	**3.10**

covering number, the rest of the proof follows from conventional learning theory and Theorem 2.1 in (Arora et al., 2018).

5 Experiments

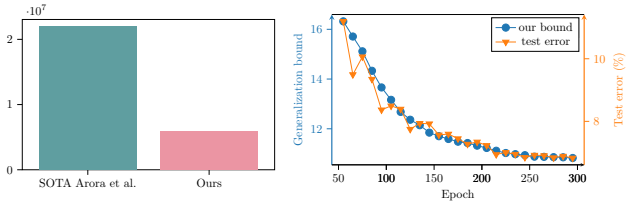
Architecture and optimization setting. The architectures we use in the experiments consist of VGG-16 (Simonyan and Zisserman, 2014), CP-VGG-16, WRN-28-10 (Zagoruyko and Komodakis, 2016) and CP-WRN-28-10 (all with batch normalization). Details of the optimization settings are in A.1.

5.1 Evaluation of Proposed Properties and Generalization Bounds

Tighter Generalization Bound. As shown in Fig 2a, our bound is much tighter than the the state-of-the-art bound achieved in Arora et al. (2018). The effective number of parameters in Arora et al. (2018) is orders of magnitude tighter than other capacity measures, such as $\ell_{1,\infty}$ (Bartlett and Mendelson, 2002), Frobenius (Neyshabur et al., 2015b), spec $\ell_{1,2}$ (Bartlett et al., 2017) and spec-fro (Neyshabur et al., 2017a) as shown in their Figure 4 Left. The use of a more effective and practical compression approach allows us to achieve better compression (detailed discussions are in Appendix Section A.2).

Generalization Bounds Correlated with Test Error. We demonstrate how our generalization bound in Theorem 4.5 is practically useful in characterizing the generalizability during training. In Figure 2b, **(1)** our calculated generalization bound matches well with the trend of the generalization error: after 140 epochs, the training error is almost zero but the test error continues to decrease in later epochs and our computed generalization bound captures these improvements especially well since epoch 150; **(2)** our calculated bound

[2] †https://github.com/kuangliu/pytorch-cifar
‡https://github.com/geifmany/cifar-vgg
* Zagoruyko and Komodakis (2016)

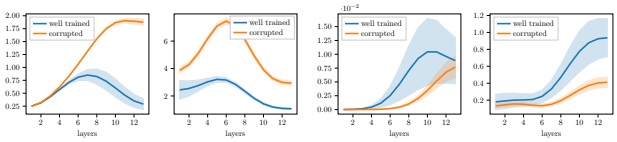


(a) Bound comparison **(b)** Generalization bound

Figure 2: (a) Effective number of parameters (proportional to the generalization bound) compared with the one derived by the current state-of-the-art (Arora et al., 2018) for VGG-16. (b) Generalization bound vs test error for CP-VGG-16. Two y-axes are applied for better visualization of the comparisons between the bound and the actual generalization/test error.

in Figure 2b for the well-trained CP-VGG-16 at epoch 300 is around 10 while the total number of parameters in this CP-VGG-16 is around 14.7M.

Compressibility of CPL: Property Evaluation. We evaluate and compare our proposed properties measuring compressibility, *tensorization factor (TF), tensor noise bound (TNB) and layer cushion (LC)*, on two different sets of models — well-trained models with small generalization errors (thus expected to obtain small $\{\hat{R}^{(k)}\}_{k=1}^{n}$) vs. corrupted models with large generalization errors (thus expected to obtain large $\{\hat{R}^{(k)}\}_{k=1}^{n}$). In Figure 3**(a)**, the number of components $\{\hat{R}^{(k)}\}_{k=1}^{n}$ returned by the compression algorithm is much smaller for well-trained models than that for corrupted models, which indicates that well-trained models have higher compressibility compared to corrupted ones as expected in our theory. Moreover, in Figure 3**(b-d)**, we can indeed tell if the model is trained using "good" data or corrupted data by evaluating our proposed properties.



(a) Rank $r^{(k)}$ **(b)** TF $t_j^{(k)}$ **(c)** TNB $\xi_j^{(k)}$ **(d)** LC $\varsigma^{(k)}$

Figure 3: Comparison of our proposed properties across layers between well-trained and corrupted CP-VGG-16. The statistics are obtained from 200 models trained under the same optimization settings.

We further apply Algorithm 1 to these well-trained and corrupted models to investigate the consistency between the compression performance of Algorithm 1 and our theoretical results: on average, Algorithm 1 achieves a 31.83% compression rate on the well-trained models, but only an 89.7% compression rate on the corrupted models (lower compression rate is better as it implies a smaller generalization error bound). Clearly,

Table 3: Average test accuracy on MNIST over the last ten epochs. Baseline simply denotes training a neural network on the corrupted training set without further processing. PairFlip denotes that the label mistakes can only happen within very similar classes and Symmetric denotes that the label mistakes may happen across different classes uniformly (Han et al., 2018).

Task: Rate	Baseline (Han et al., 2018)	F-correction (Han et al., 2018)	MentorNet (Jiang et al., 2017)	CT (Han et al., 2018)	CT + CPL
PairFlip: 45%	56.52 ± 0.55	0.24 ± 0.03	80.88 ± 4.45	87.63 ± 0.21	$\mathbf{92.43 \pm 0.01}$
Symmetric: 50%	66.05 ± 0.61	79.61 ± 1.96	90.05 ± 0.30	91.32 ± 0.06	$\mathbf{94.70 \pm 0.05}$
Symmetric: 20%	94.05 ± 0.16	$\mathbf{98.80 \pm 0.12}$	96.70 ± 0.22	97.25 ± 0.03	97.91 ± 0.01

the low-rank structures in well-trained models allow them to be compressed much further, consistent with our theoretical analysis of Algorithm 1.

5.2 Generalization Improvement on Real Data Experiments

Expressive Power of Neural Networks with CP layers. As shown in Table 1, neural networks equipped with CP layers maintain competitive training and test accuracies.

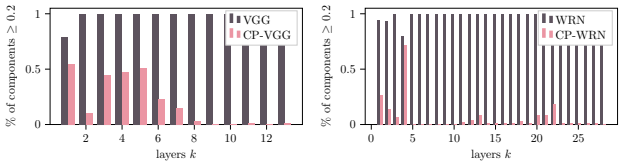
Generalization Improvements under Label Noise. The memorization effect is directly linked to the deteriorated generalization performance of the network (Zhang et al., 2017). Therefore we study how our proposed CPL structure affects the generalizability of a neural network with presence of strong memorization effect — under label noise setting. We assign random labels to a proportion of the training data and train the neural network until convergence. Then we test the network's performance on the uncorrupted test data. As shown in Table 2, CP-VGG consistently achieves better generalization performance compared to the traditional VGG under various label corruption ratios.

Our CPL, combined with co-teaching (CT) (Han et al., 2018) (the SOTA method for defeating label noise) further improves its performance as shown in Table 3 where we also compare our method CT+CPL against other different label-noise methods (Han et al., 2018). Besides, in Figure 5, our method CT+CPL consistently outperforms the SOTA method (CT) with various choices of number of components.

5.3 CPL Is Natural for Compression

Applying CPL for neural network compression is extensively studied in Su et al. (2018), therefore we focus on explaining why CPL is natural for compression and analyzing the compressibility of CPLs.

Low Rankness in Neural Networks with CPL vs Traditional Neural Networks. The low rankness of a CP-VGG and a traditional VGG is demonstrated by Figure 4 where we display the ratios of the number of components with amplitudes above a given threshold 0.2. We clearly see that VGG with CPL exhibits low rankness consistently for all layers while the traditional VGG is not low-rank. Notice that the CP spectrum in each CPL is normalized by dividing the largest amplitude and the CP components of traditional VGG are obtained via explicit CP decompositions with reconstruction error set to 1e-3.



(a) VGG-16 **(b)** WRN-28-10

Figure 4: Comparison of low rankness (compressibility) across layers between neural networks with CPL and standard neural networks

No Fine-tuning Needed for CPL. Many works using tensor methods for neural network compression require computationally expensive fine-tuning (e.g. 200 epochs end-to-end training on the compressed networks) to recover the compressed network's test performance Jaderberg et al. (2014); Denton et al. (2014); Lebedev et al. (2014); Kim et al. (2015); Garipov et al. (2016); Wang et al. (2018); Su et al. (2018). However, the compression we perform does not require any fine tuning since it directly prunes out the components with amplitudes below some given threshold. *In experiments, we compress a CP-WRN-28-10, which has the same number of parameters as WRN-28-10, by 8× with only 0.56% performance drop on CIFAR10 image classification.* The full compression results for CP-WRN-28-10 under different cutting-off thresholds are shown in Table 5, where components whose amplitudes are under the cutting-off threshold are pruned.

6 Conclusion and Discussion

In this work, we derive a practical compression-based generalization bound via the proposed layerwise structure CP layers, and demonstrate the effectiveness of using tensor methods in theoretical analyses of deep neural networks. With a series of benchmark experiments, we show the practical usage of our generalization bound and the effectiveness of our proposed structure CPL in terms of compression and generalization. A possible future direction is studying the effectiveness of other tensor decomposition methods such as Tucker or Tensor Train.

Acknowledgement

This research was supported by startup fund from Department of Computer Science of University of Maryland, National Science Foundation IIS-1850220 CRII Award 030742- 00001, DOD-DARPA-Defense Advanced Research Projects Agency Guaranteeing AI Robustness against Deception (GARD), Laboratory for Physical Sciences at University of Maryland. This research was also supported in part by JSPS Kakenhi (26280009, 15H05707 and 18H03201), Japan Digital Design and JST-CREST. Huang was also supported by Adobe, Capital One and JP Morgan faculty fellowships. We thank Ziyin Liu for supporting this research with great advice and efforts. We thank Jin-peng Liu, Kai Wang, and Dongruo Zhou for helpful discussions and comments. We thank Jingxiao Zheng for supporting additional computing resources.

References

Anima Anandkumar, Rong Ge, and Majid Janzamin. Guaranteed non-orthogonal tensor decomposition via alternating rank-1 updates. *arXiv preprint arXiv:1402.5180*, 2014a.

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014b.

Animashree Anandkumar, Rong Ge, and Majid Janzamin. Learning overcomplete latent variable models through tensor methods. In *Conference on Learning Theory (COLT)*, June 2015.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. 2018.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Peter L Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear vc dimension bounds for piecewise polynomial networks. In *Advances in Neural Information Processing Systems*, pages 190–196, 1999.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

Simon S Du and Jason D Lee. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214*, 2016.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 297–299. PMLR, 06–09 Jul 2018. URL http://proceedings.mlr.press/v75/golowich18a.html.

Alon Gonen and Shai Shalev-Shwartz. Fast rates for empirical risk minimization of strict saddle problems. In *COLT*, 2017.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pages 8536–8546, 2018.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1225–1234. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045520.

Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension bounds for piecewise linear neural networks. In *Conference on Learning Theory*, pages 1064–1068, 2017.

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*, pages 770–778, 2016.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

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.

Furong Huang, UN Niranjan, Mohammad Umar Hakeem, and Animashree Anandkumar. Online tensor methods for learning latent variable models. *Journal of Machine Learning Research*, 16:2797–2835, 2015.

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

Daniel Jakubovitz, Raja Giryes, and Miguel RD Rodrigues. Generalization error in deep learning. *arXiv preprint arXiv:1808.01174*, 2018.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 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. *arXiv preprint arXiv:1511.06530*, 2015.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

Jean Kossaifi, Aran Khanna, Zachary Lipton, Tommaso Furlanello, and Anima Anandkumar. Tensor contraction layers for parsimonious deep nets. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1940–1946. IEEE, 2017.

Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *The Journal of Machine Learning Research*, 20(1):925–930, 2019.

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

Joseph B Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. *Multiway data analysis*, pages 7–18, 1989.

Ilja Kuzborskij and Christoph H. Lampert. Data-dependent stability of stochastic gradient descent. In *ICML*, 2018.

John Langford and Rich Caruana. (not) bounding the true error. In *Advances in Neural Information Processing Systems*, pages 809–816, 2002.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

Jialin Li and Furong Huang. Guaranteed simultaneous asymmetric tensor decomposition via orthogonalized alternating least squares. *arXiv preprint arXiv:1805.10348*, 2018.

David A McAllester. Pac-bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 164–170. ACM, 1999a.

David A McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999b.

Hrushikesh N Mhaskar and Tomaso Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

Behnam Neyshabur, Ruslan R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2015a.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401, 2015b.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for

neural networks. *arXiv preprint arXiv:1707.09564*, 2017a.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017b.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Generalization error of invariant classifiers. *arXiv preprint arXiv:1610.04574*, 2016.

Jiahao Su, Jingling Li, Bobby Bhattacharjee, and Furong Huang. Tensorial neural networks: Generalization of neural networks and application to model compression. *https://arxiv.org/pdf/1805.10352.pdf*, 2018.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. *learning*, 14(15):13–31, 2018.

Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. *arXiv preprint arXiv:1804.05862*, 2018.