End-to-end Learning of a Convolutional Neural Network via Deep Tensor Decomposition

Samet Oymak* and Mahdi Soltanolkotabi†

May 18, 2018

Abstract

In this paper we study the problem of learning the weights of a deep convolutional neural network. We consider a network where convolutions are carried out over non-overlapping patches with a single kernel in each layer. We develop an algorithm for simultaneously learning all the kernels from the training data. Our approach dubbed Deep Tensor Decomposition (DeepTD[1]) is based on a rank-1 tensor decomposition. We theoretically investigate DeepTD under a realizable model for the training data where the inputs are chosen i.i.d. from a Gaussian distribution and the labels are generated according to planted convolutional kernels. We show that DeepTD is data-efficient and provably works as soon as the sample size exceeds the total number of convolutional weights in the network. We carry out a variety of numerical experiments to investigate the effectiveness of DeepTD and verify our theoretical findings.

1 Introduction

Deep neural network (DNN) architectures have led to state of the art performance in many domains including image recognition, natural language processing, recommendation systems, and video analysis [9, 16, 18, 22, 39]. Convolutional neural networks (CNNs) are a class of deep, feed-forward neural networks with a specialized DNN architecture. CNNs are responsible for some of the most significant performance gains of DNN architectures. In particular, CNN architectures have led to striking performance improvements for image/object recognition tasks. Convolutional neural networks, loosely inspired by the visual cortex of animals, construct increasingly higher level features (such as mouth and nose) from lower level features such as pixels. An added advantage of CNNs which makes them extremely attractive for large-scale applications is their remarkable efficiency which can be attributed to: (1) intelligent utilization of parameters via weight-sharing, (2) their convolutional nature which exploits the local spatial structure of images/videos effectively, and (3) highly efficient matrix/vector multiplication involved in CNNs compared to fully-connected neural network architectures.

Despite the wide empirical success of CNNs the reasons for the effectiveness of neural networks and CNNs in particular is still a mystery. Recently there has been a surge of interest in developing more rigorous foundations for neural networks [1, 19, 23, 25, 27, 32, 34, 40, 41]. Most of this existing literature however focus on learning shallow neural networks typically consisting of zero or one hidden layer. In practical applications, depth seems to play a crucial role in constructing progressively higher-level features from pixels. Indeed, state of the art Resnet models typically have hundreds of layers. Furthermore, recent results suggest that increasing depth may substantially boost the expressive power of neural networks [7, 29].

In this paper, we propose an algorithm for approximately learning an arbitrarily deep CNN model with rigorous guarantees. Our goal is to provide theoretical insights towards better understanding when training deep CNN architectures is computationally tractable and how much data is required for successful training. We focus on a realizable model where the inputs are chosen i.i.d. from a Gaussian distribution and the labels

*Department of Electrical and Computer Engineering, University of California, Riverside, CA
†Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA
[1]Pun intended.

are generated according to planted convolutional kernels. We use both labels and features in the training data to construct a tensor. Our first insight is that, in the limit of infinite data this tensor converges to a *population* tensor which is approximately rank one and whose factors reveal the direction of the kernels. Our second insight is that even with finite data this *empirical* tensor is still approximately rank one. We show that the gap between the population and empirical tensors provably decreases with the increase in the size of the training data set and becomes negligible as soon as the size of the training data becomes proportional to the total numbers of the parameters in the planted CNN model. Combining these insights we provide a tensor decomposition algorithm to learn the kernels from training data. We show that our algorithm approximately learns the kernels (up to sign/scale ambiguities) as soon as the size of the training data is proportional to the total number of parameters of the planted CNN model. Our results can be viewed as a first step towards provable end-to-end learning of practical deep CNN models. Extending the connections between neural networks and tensors [7, 19, 40], we show how tensor decomposition can be utilized to approximately learn deep networks despite the presence of nonlinearities and growing depth. While our focus in this work is limited to tensors, we believe that our proposed algorithm may provide valuable insights for initializing local search methods (such as stochastic gradient descent) to enhance the quality and/or speed of CNN training.

2 Problem formulation and models

In this section we discuss the CNN model which is the focus of this paper. Our exposition is directly adapted from the companion paper [28]. A fully connected artificial neural network is composed of computational units called neurons. The neurons are decomposed into layers consisting of one input layer, one output layer and a few hidden layers with the output of each layer is fed in (as input) to the next layer. In a CNN model the output of each layer is related to the input of the next layer by a convolution operation. In this paper we focus on CNN model where the stride length is equal to the length of the kernel. This is sometimes referred to as a non-overlapping convolution operation. We now formally define this operation for future reference.

Definition 2.1 (Non-overlapping convolution) *For two vectors $\boldsymbol{k} \in \mathbb{R}^d$ and $\boldsymbol{h} \in \mathbb{R}^{p=d\bar{p}}$ their non-overlapping convolution, denoted by $\boldsymbol{k} \boxtimes \boldsymbol{h}$ yields a vector $\boldsymbol{u} \in \mathbb{R}^{\bar{p}=\frac{p}{d}}$ whose entries are given by*

$$\boldsymbol{u}_i = \langle \boldsymbol{k}^{(\ell)}, \boldsymbol{h}[i] \rangle \quad \textit{where} \quad \boldsymbol{h}[i] := \begin{bmatrix} \boldsymbol{h}_{(i-1)d+1} \\ \boldsymbol{h}_{(i-1)d+2} \\ \vdots \\ \boldsymbol{h}_{id} \end{bmatrix}.$$

As mentioned earlier the non-overlapping convolution resembles the standard convolution between two vectors $\boldsymbol{k} \in \mathbb{R}^d$ and $\boldsymbol{h} \in \mathbb{R}^p$, denoted by $\boldsymbol{k} \circledast \boldsymbol{h}$, which yields a vector in $\mathbb{R}^{d+p}$. The main difference is that rather than sliding the kernel one entry at a time across overlapping patches of $\boldsymbol{h}$ in the convolution summation we use a stride length equal to the size of the kernel (i.e. slide the kernel by the length of the kernel) so that the summation is carried out over non-overlapping patches of $\boldsymbol{h}$. In this paper it is often convenient to view convolutions as matrix/vector multiplications. This leads us to the definition of the kernel matrix below.

Definition 2.2 (Kernel matrix) *Consider a kernel $\boldsymbol{k} \in \mathbb{R}^d$ and any vector $\boldsymbol{h} \in \mathbb{R}^{p=\bar{p}d}$. Corresponding to the non-overlapping convolution $\boldsymbol{k} \boxtimes \boldsymbol{h}$, we associate a kernel matrix $\boldsymbol{K}_{\boxtimes} \in \mathbb{R}^{\bar{p} \times p}$ defined as*

$$\boldsymbol{K}_{\boxtimes} := \boldsymbol{I}_{\bar{p}} \otimes \boldsymbol{k}^T = \begin{bmatrix} \boldsymbol{k}^T & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{k}^T & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{k}^T \end{bmatrix}.$$

Here, $\boldsymbol{A} \otimes \boldsymbol{B}$ denotes the Kronecker product between the two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and $\boldsymbol{I}_{\bar{p}}$ denotes the $\bar{p} \times \bar{p}$ identity matrix. We note that based on this definition $\boldsymbol{k} \boxtimes \boldsymbol{h} = \boldsymbol{K}_{\boxtimes} \boldsymbol{h}$. Throughout the paper we shall use $\boldsymbol{K}$ interchangeably with $\boldsymbol{K}_{\boxtimes}$ to denote this kernel matrix with the dependence on the underlying kernel and its non-overlapping form implied.

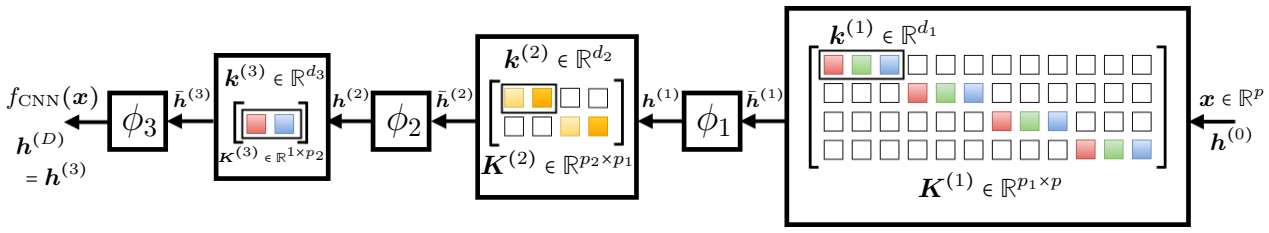


Figure 1: Depiction of the input-output relationship of a non-overlapping Convolutional Neural Network (CNN) model along with the various notations and symbols.

With the definition of the non-overlapping convolution and the corresponding kernel matrix in hand we are now ready to define the CNN model which is the focus of this paper. For the convenience of the reader the CNN input-output relationship along with the corresponding notation is depicted in Figure 1.

- **Depth and numbering of the layers.** We consider a network of depth D where we number the input as layer 0 and the output as layer D and the hidden layers 1 to $D-1$.
- **Layer dimensions and representations.** We assume the input of the CNN, denoted by $\boldsymbol{x} \in \mathbb{R}^p$, consists of p features and the output is a one dimensional label. We also assume the hidden layers (numbered by $\ell = 1, 2, \ldots, D-1$) consists of p_ℓ units with $\bar{\boldsymbol{h}}^{(\ell)} \in \mathbb{R}^{p_\ell}$ and $\boldsymbol{h}^{(\ell)} \in \mathbb{R}^{p_\ell}$ denoting the input and output values of the units in the ℓth hidden layer. For consistency of our notation we shall also define $\boldsymbol{h}^{(0)} := \boldsymbol{x} \in \mathbb{R}^p$ and note that the output of the CNN is $\boldsymbol{h}^{(D)} \in \mathbb{R}$. Furthermore, $p_0 = p$ and $p_D = 1$.
- **Kernel dimensions and representation.** For $\ell = 1, \ldots D$ we assume the kernel relating the output of layer $(\ell-1)$ to the input of layer ℓ is of dimension d_ℓ and is denoted by $\boldsymbol{k}^{(\ell)} \in \mathbb{R}^{d_\ell}$.
- **Inter-layer relationship.** We assume the inputs of layer ℓ (denoted by $\bar{\boldsymbol{h}}^{(\ell)} \in \mathbb{R}^{p_\ell}$) are related to the outputs of layer $(\ell-1)$ (denoted by $\boldsymbol{h}^{(\ell-1)} \in \mathbb{R}^{p_{\ell-1}}$) via a non-overlapping convolution

$$\bar{\boldsymbol{h}}^{(\ell)} = \boldsymbol{k}^{(\ell)} \boxtimes \boldsymbol{h}^{(\ell-1)} = \boldsymbol{K}^{(\ell)} \boldsymbol{h}^{(\ell-1)} \quad \text{for} \quad \ell = 1, \ldots, D.$$

  In the latter equality we have used the representation of non-overlapping convolution as a matrix/vector product involving the kernel matrix $\boldsymbol{K}^{(\ell)} \in \mathbb{R}^{p_\ell \times p_{\ell-1}}$ associated with the kernel $\boldsymbol{k}^{(\ell)} \in \mathbb{R}^{d_\ell}$ per Definition 2.2. We note that by the nature of the non-overlapping convolution we have that $p_\ell = p_{\ell-1}/d_\ell$.
- **Activation functions and intra-layer relationship.** We assume the input of each hidden unit is related to its output by applying an activation function $\phi_\ell : \mathbb{R} \to \mathbb{R}$. More precisely, $\boldsymbol{h}^{(\ell)} := \phi_\ell(\bar{\boldsymbol{h}}^{(\ell)})$ where for a vector $\boldsymbol{u} \in \mathbb{R}^p$, $\phi_\ell(\boldsymbol{u}) \in \mathbb{R}^p$ is a vector obtained by applying the activation function ϕ_ℓ to each of the entries of $\boldsymbol{u}$. We allow for using distinct activation functions $\{\phi_\ell\}_{\ell=1}^D$ at every layer. Throughout, we also assume all activations are 1-Lipschitz functions (i.e. $|\phi_\ell(a) - \phi_\ell(b)| \le |a-b|$).
- **Final output.** To summarize the input-output relation of our CNN model with an input $\boldsymbol{x} \in \mathbb{R}^p$ and input kernel $\boldsymbol{k}^{(1)} \in \mathbb{R}^{d_1}$ is given by

$$\boldsymbol{x} \mapsto f_{CNN}(\boldsymbol{x}) := \boldsymbol{h}^{(D)}, \tag{2.1}$$

  with

$$\boldsymbol{h}^{(\ell)} = \phi_\ell(\bar{\boldsymbol{h}}^{(\ell)}) \quad \text{and} \quad \bar{\boldsymbol{h}}^{(\ell)} = \boldsymbol{K}^{(\ell)} \boldsymbol{h}^{(\ell-1)} = \boldsymbol{K}^{(\ell)} \phi_{\ell-1}\left(\boldsymbol{K}^{(\ell-1)} \phi_{\ell-2}\left(\ldots\left(\boldsymbol{K}^{(2)} \phi_1\left(\boldsymbol{K}^{(1)} \boldsymbol{x}\right)\right)\right)\right). \tag{2.2}$$

3 Algorithm: Deep Tensor Decomposition (DeepTD)

This paper introduces an approach to approximating the convolutional kernels from training data based on tensor decompositions dubbed DeepTD, which consists of a carefully designed tensor decomposition. To connect these two problems, we begin by stating how we intend to construct the tensor from the training data. To this aim given any input data $\boldsymbol{x} \in \mathbb{R}^p$, we form a D-way tensor $\boldsymbol{X} \in \mathbb{R}^{\otimes_{\ell=1}^D d_\ell}$ as follows. First, we convert $\boldsymbol{x}$ into a matrix by placing every d_1 consecutive entries of $\boldsymbol{x}$ as a row of a matrix of size $p_1 \times d_1$. From this matrix we then create a 3-way tensor of size $p_2 \times d_2 \times d_1$ by grouping d_2 consecutive entries of each of the d_1 columns and so on. We repeat this procedure D times to arrive at the D-way tensor $\boldsymbol{X} \in \mathbb{R}^{\otimes_{\ell=1}^D d_\ell}$. We define $\mathcal{T} : \mathbb{R}^p \mapsto \mathbb{R}^{\otimes_{\ell=1}^D d_\ell}$ as the corresponding tensor operation that maps $\boldsymbol{x} \in \mathbb{R}^p$ to $\boldsymbol{X} \in \mathbb{R}^{\otimes_{\ell=1}^D d_\ell}$.

Given a set of training data consisting of n input/output pairs $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$ we construct a tensor $\boldsymbol{T}_n$ by tensorizing the input vectors as discussed above and calculating a weighted combination of these tensorized inputs where the weights depends on the output labels for the training data. More precisely,

$$\boldsymbol{T}_n := \frac{1}{n} \sum_{i=1}^n (y_i - y_{avg}) \, \boldsymbol{X}_i \quad \text{where} \quad y_{avg} = \frac{1}{n} \sum_{i=1}^n y_i \quad \text{and} \quad \boldsymbol{X}_i = \mathcal{T}(\boldsymbol{x}_i). \tag{3.1}$$

We then perform a rank-1 tensor decomposition on this T_n tensor to approximate the convolutional kernels. Specifically we solve

$$\hat{\boldsymbol{k}}^{(1)}, \ldots, \hat{\boldsymbol{k}}^{(D)} = \underset{\boldsymbol{v}_1 \in \mathbb{R}^{d_1}, \boldsymbol{v}_2 \in \mathbb{R}^{d_2}, \ldots, \boldsymbol{v}_D \in \mathbb{R}^{d_D}}{\arg\max} \left\langle \boldsymbol{T}_n, \bigotimes_{\ell=1}^D \boldsymbol{v}_\ell \right\rangle \quad \text{subject to} \quad \|\boldsymbol{v}_1\|_{\ell_2} = \|\boldsymbol{v}_2\|_{\ell_2} = \ldots = \|\boldsymbol{v}_D\|_{\ell_2} = 1. \tag{3.2}$$

In the above $\bigotimes_{\ell=1}^D \boldsymbol{v}_\ell$ denotes the tensor resulting from the outer product of the vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_D$. This tensor rank decomposition is also known as CANDECOMP/PARAFAC (CP) decomposition [5] and can be solved efficiently using Alternating Least Squares (ALS) and a variety of other algorithms [2,3,14].[2]

At this point it is completely unclear why the tensor T_n or its rank-1 decomposition can yield anything useful. The main intuition is that as the data set grows ($n \to \infty$) the *empirical* tensor $\boldsymbol{T}_n$ converges close to a *population* tensor $\boldsymbol{T}$ whose rank-1 decomposition reveals useful information about the kernels. Specifically, we will show that

$$\lim_{n \to \infty} \boldsymbol{T}_n = \boldsymbol{T} := \mathbb{E}_{\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)} [f_{\text{CNN}}(\boldsymbol{x}) \mathcal{T}(\boldsymbol{x})] \approx \alpha \bigotimes_{\ell=1}^D \boldsymbol{k}^{(\ell)}, \tag{3.3}$$

with α a scalar whose value shall be discussed later on. Here, $\boldsymbol{x}$ is a Gaussian random vector with i.i.d. $\mathcal{N}(0,1)$ entries and represents a typical input with $f_{\text{CNN}}(\boldsymbol{x})$ the corresponding output and $\mathcal{T}(\boldsymbol{x})$ the tensorized input. We will also utilize a concentration argument to show that when the training data set originates from an i.i.d. distribution, for a sufficiently large training data n, $\boldsymbol{T}_n$ yields a good approximation of the population tensor $\boldsymbol{T}$.

Another perhaps perplexing aspect of the construction of $\boldsymbol{T}_n$ in (3.1) is the subtraction by y_{avg} in the weights. The reason this may be a source of confusion is that based on the intuition above

$$\mathbb{E}[\boldsymbol{T}] = \mathbb{E}\Big[\frac{1}{n} \sum_{i=1}^n y_i \boldsymbol{X}_i\Big] = \mathbb{E}\Big[\frac{1}{n-1} \sum_{i=1}^n (y_i - y_{avg}) \, \boldsymbol{X}_i\Big] = \frac{n}{n-1} \mathbb{E}[\boldsymbol{T}_n] \approx \alpha \bigotimes_{\ell=1}^D \boldsymbol{k}^{(\ell)},$$

so that the subtraction by the average seems completely redundant. The main purpose of this subtraction is to ensure the weights $y_i - y_{avg}$ are centered (have mean zero). This centering allows for a much better concentration of the empirical tensor around its population counter part and is crucial to the success of our approach. We would like to point out that such a centering procedure is reminiscent of batch-normalization heuristics deployed when training deep neural networks.

[2] We would like to note that we believe that the assumptions under which the above stated algorithms solve the problem (3.2) holds for the empirical tensor $\boldsymbol{T}_n$ under our planted model. We do not however pursue this here and leave it to future work.

Finally, we note that based on (3.3), the rank-1 tensor decomposition step can recover the convolutional kernels $\{\boldsymbol{k}^{(\ell)}\}_{\ell=1}^{D}$ up to sign and scaling ambiguities. Unfortunately, depending on the activation function, it may be impossible to overcome these ambiguities. For instance, if the activations are homogeneous (i.e. $\phi_\ell(ax) = a\phi_\ell(x)$), then scaling up one layer and scaling down the other layer by the same amount does not change the overall function $f_{\text{CNN}}(\cdot)$. Similarly, if the activations are odd functions, negating two of the layers at the same time preserves the overall function. In Section 5 we discuss some heuristics ands theoretical guarantees for overcoming these sign/scale ambiguities.

4 Main results

In this section we introduce our theoretical results for DeepTD. We will discuss these results in three sections. In Section 4.1 we show that the empirical tensor concentrates around its population counterpart. Then in Section 4.2 we show that the population tensor is well-approximated by a rank-1 tensor whose factors reveal the convolutional kernels. Finally, in Section 4.3 we combine these results to show DeepTD can approximately learn the convolutional kernels up to sign/scale ambiguities.

4.1 Concentration of the empirical tensor

Our first result shows that the empirical tensor concentrations around the population tensor. We measure the quality of this concentration via the tensor spectral norm defined below.

Definition 4.1 *Let $\mathcal{RO}$ be the set of rank-one tensors characterized by*

$$\mathcal{RO} = \left\{ \boldsymbol{V} \in \mathbb{R}^{\otimes_{\ell=1}^{D} d_\ell} \;\middle|\; \boldsymbol{V} = \bigotimes_{\ell=1}^{D} \boldsymbol{v}_\ell,\ \|\boldsymbol{v}_i\|_{\ell_2} \le 1 \text{ for all } 1 \le i \le D \right\}.$$

The spectral norm of a tensor $\boldsymbol{X} \in \mathbb{R}^{\otimes_{\ell=1}^{D} d_\ell}$ is given by the supremum,

$$|||\boldsymbol{T}||| = \sup_{\boldsymbol{V} \in \mathcal{RO}} \langle \boldsymbol{V}, \boldsymbol{T} \rangle .$$

Theorem 4.2 *Consider a CNN model $\boldsymbol{x} \mapsto f_{CNN}(\boldsymbol{x})$ of the form (2.1) consisting of $D \ge 2$ layers with convolutional kernels $\boldsymbol{k}^{(1)}, \boldsymbol{k}^{(2)}, \ldots, \boldsymbol{k}^{(D)}$ of lengths $d_1, d_2, \ldots, d_D$. Let $\boldsymbol{x} \in \mathbb{R}^p$ be a Gaussian random vector distributed as $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$ with the corresponding labels $y = f_{CNN}(\boldsymbol{x})$ generated by the CNN model and $\boldsymbol{X} := \mathcal{T}(\boldsymbol{x})$ the corresponding tensorized input. Suppose the data set consists of n training samples where the feature vectors $\boldsymbol{x}_i \in \mathbb{R}^p$ are distributed i.i.d. $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$ with the corresponding labels $y_i = f_{CNN}(\boldsymbol{x}_i)$ generated by the same CNN model and $\boldsymbol{X}_i := \mathcal{T}(\boldsymbol{x}_i)$ the corresponding tensorized input. Then the empirical tensor $\boldsymbol{T}_n$ and population tensor $\boldsymbol{T}$ defined based on this dataset obey*

$$|||\boldsymbol{T}_n - \boldsymbol{T}||| := \left|\left|\left| \frac{1}{n} \sum_{i=1}^{n} (y_i - y_{avg}) \boldsymbol{X}_i - \mathbb{E}[y\boldsymbol{X}] \right|\right|\right| \le c \prod_{\ell=1}^{D} \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2} \frac{\sqrt{\left(\sum_{\ell=1}^{D} d_\ell\right) \log D} + t}{\sqrt{n}}, \tag{4.1}$$

with probability at least $1 - 5e^{-\min(t^2, t\sqrt{n}, n)}$, where $c > 0$ is an absolute constant.

The theorem above shows that the empirical tensor approximates the population tensor with high probability. This theorem also shows that the quality of this approximation is proportional to $\prod_{\ell=1}^{D} \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}$. This is natural as $\prod_{\ell=1}^{D} \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}$ is an upper-bound on the Lipschitz constant of the network and shows how much the CNN output fluctuates with changes in the input. The more fluctuations, the less concentrated the empirical tensor is, translating into a worse approximation guarantee. Furthermore, the quality of this approximation grows with the square root of the parameters in the model ($\sum_{\ell=1}^{D} d_\ell$) and is inversely proportional to the square root of the number of samples (n) which are typical scalings in statistical learning. We would also like to note that as we will see in the forthcoming sections, in many cases $|||\boldsymbol{T}|||$ is roughly on the order of

$\prod_{\ell=1}^{D} \left\| \boldsymbol{k}^{(\ell)} \right\|_{\ell_2}$ so that (4.1) guarantees that $\|\boldsymbol{T}_n - \boldsymbol{T}\|/\|\boldsymbol{T}\| \leq c' \sqrt{\frac{(\sum_{\ell=1}^{D} d_\ell) \log D}{n}}$. Therefore, the relative error in the approximation is less than ϵ as soon as the number of observations exceeds the number of parameters in the model by a logarithmic factor in depth i.e. $n \gtrsim (\sum_{\ell=1}^{D} d_\ell) \frac{\log D}{\epsilon^2}$.

4.2 Rank one approximation of the population tensor

Our second result shows that the population tensor can be approximated by a rank one tensor. To explain the structure of this rank one tensor and quantify the quality of this approximation we require a few definitions. The first quantity roughly captures the average amount by which the nonlinear activations amplify or attenuate the size of an input feature at the output. This quantity is formally defined below.

Definition 4.3 (CNN gain) *Let $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_p)$ and define the hidden unit/output values of the CNN based on this random input per equations (2.2) and (2.1). We define the CNN gain as*

$$\alpha_{CNN} = \prod_{\ell=1}^{D} \mathbb{E}[\phi_\ell'(\bar{\boldsymbol{h}}_1^{(\ell)})].$$

In words, this is the product of expectations of the activations evaluated at the first entry of each layer.

This quantity is the product of the average slopes of the activations evaluated along a path connecting the first input feature to the first hidden units across the layers all the way to the output. We note that this quantity is the same when calculated along any path connecting an input feature to the output passing through the hidden units. Therefore, this quantity can be thought of as the average gain (amplification or attenuation) of a given input feature due to the nonlinear activations in the network. To gain some intuition consider a ReLU network which is mostly inactive. Then the network is dead and $\alpha_{\text{CNN}} \approx 0$. On the other extreme if all ReLU units are active the network operates in the linear regime and $\alpha_{\text{CNN}} = 1$. We would like to point out that α_{CNN} can in many cases be bounded from below by a constant. For instance for ReLU activations as long as the kernels obey

$$\left(\mathbf{1}^T \boldsymbol{k}^{(\ell)}\right) \geq 4 \left\| \boldsymbol{k}^{(\ell)} \right\|_{\ell_2}, \tag{4.2}$$

then $\alpha_{\text{CNN}} \geq \gamma$ with γ a fixed numerical constant (formally proved in our companion paper [28]). We note that an assumption similar to (4.2) is needed for the network to be active. This is because if the kernel sums are negative one can show that with high probability, all the ReLUs after the first layer will be inactive and the network will be dead. In [28] we also show that similar results hold for other popular activations including softplus. With this definition in hand, we are now ready to describe the form of the rank one tensor that approximates the population tensor.

Definition 4.4 (Rank one CNN tensor) *We define the rank one CNN tensor $\boldsymbol{L}_{CNN} \in \mathbb{R}^{\otimes_{\ell=1}^{D} d_\ell}$ as*

$$\boldsymbol{L}_{CNN} = \alpha_{CNN} \bigotimes_{\ell=1}^{D} \boldsymbol{k}^{(\ell)}.$$

In words, this is the product of the kernels $\{\boldsymbol{k}^{(\ell)}\}_{\ell=1}^{D}$ scaled by the CNN gain α_{CNN}.

To quantify how well the rank one CNN tensor approximates the population tensor we need two definitions. The first definition concerns the activation functions.

Definition 4.5 (Activation smoothness) *We assume the activations are differentiable everywhere and S-smooth (i.e. $|\phi_\ell'(x) - \phi_\ell'(y)| \leq S|x - y|$ for all $x, y \in \mathbb{R}$) for some $S \geq 0$.*

The reason smoothness of the activations play a role in the quality of the rank one approximation is that smoother activations translate into smoother variations in the entries of the population tensor. Therefore, the population tensor can be better approximated by a low-rank tensor. The second definition captures how diffused the kernels are.

Definition 4.6 (Kernel diffuseness parameter) *Given kernels* $\{\boldsymbol{k}^{(\ell)}\}_{\ell=1}^{D}$ *with dimensions* $\{d_\ell\}_{\ell=1}^{D}$*, the kernel diffuseness parameter* μ *is defined as*

$$\mu = \sup_{1\le\ell\le D} \frac{\sqrt{d_\ell}\left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_\infty}}{\left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}}.$$

The less diffused (or more spiky) the kernels are, the more the population tensor fluctuates and thus the quality of the approximation to a rank one tensor decreases. With these definitions in place, we are now ready to state our theorem on approximating a population tensor with a rank one tensor.

Theorem 4.7 *Consider the setup of Theorem 4.2. Furthermore, assume the activations are* S*-smooth per Definition 4.5 and the convolutional kernels are* μ*-diffused per Definition 4.6. Then, the population tensor* $\boldsymbol{T} := \mathbb{E}[y\boldsymbol{X}]$ *can be approximated by the rank-1 tensor* $\boldsymbol{L}_{CNN} := \alpha_{CNN}\bigotimes_{\ell=1}^{D}\boldsymbol{k}^{(\ell)}$ *as follows*

$$\left|\!\left|\!\left|\boldsymbol{T}-\boldsymbol{L}_{CNN}\right|\!\right|\!\right| \le \|\boldsymbol{T}-\boldsymbol{L}_{CNN}\|_F \le \sqrt{8\pi}\mu S\cdot\prod_{i=1}^{D}\left\|\boldsymbol{k}^{(i)}\right\|_{\ell_2}\cdot\sup_{\ell}\prod_{i=1}^{\ell}\left\|\boldsymbol{k}^{(i)}\right\|_{\ell_2}\frac{D}{\sqrt{\min_{\ell} d_\ell}}.$$

The theorem above states that the quality of the rank one approximation deteriorates with increase in the smoothness of the activations and the diffuseness of the convolutional kernels. As mentioned earlier increase in these parameters leads to more fluctuations in the population tensor making it less likely that it can be well approximated by a rank one tensor. We also note that $\left|\!\left|\!\left|\boldsymbol{L}_{\mathrm{CNN}}\right|\!\right|\!\right| = \alpha_{\mathrm{CNN}}\prod_{\ell=1}^{D}\left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}$ and therefore the relative error in this approximation is bounded by

$$\frac{\left|\!\left|\!\left|\boldsymbol{T}-\boldsymbol{L}_{\mathrm{CNN}}\right|\!\right|\!\right|}{\left|\!\left|\!\left|\boldsymbol{L}_{\mathrm{CNN}}\right|\!\right|\!\right|} \le \sqrt{8\pi}\frac{\mu S}{\alpha_{\mathrm{CNN}}}\sup_{\ell}\prod_{i=1}^{\ell}\left\|\boldsymbol{k}^{(i)}\right\|_{\ell_2}\frac{D}{\sqrt{\min_{\ell} d_\ell}}.$$

We would like to note that for many activations the smoothness is bounded by a constant. For instance, for the softplus activation ($\phi(x) = \log(1+e^x)$) and one can show that $S \le 1$. As stated earlier, under appropriate assumptions on the kernels and activations, the CNN gain α_{CNN} is also bounded from below by a constant. Assume the convolutional kernels have unit norm and are sufficiently diffused so that the diffuseness parameter is bounded by a constant. We can then conclude that

$$\frac{\left|\!\left|\!\left|\boldsymbol{T}-\boldsymbol{L}_{\mathrm{CNN}}\right|\!\right|\!\right|}{\left|\!\left|\!\left|\boldsymbol{L}_{\mathrm{CNN}}\right|\!\right|\!\right|} \le \frac{c}{\alpha_{\mathrm{CNN}}}\frac{D}{\sqrt{\min_{\ell} d_\ell}}.$$

This implies that as soon as the length of the convolutional patches scale with the square of depth of the network by a constant factor the rank one approximation is sufficiently good. Our back-of-the-envelope calculations suggest that the correct scaling is linear in D versus the quadratic result we have established here. Improving our result to achieve the correct scaling is an interesting future research direction. Finally, we would like to note that while we have assumed differentiable and smooth activations we expect our results to apply to popular non-differentiable activations such as ReLU activations. Indeed, as it will become clear in our proofs (e.g. see Lemma 9.14) an average notion of smoothness appears to be sufficient for our results to apply.

4.3 Learning the convolutional kernels

We demonstrated in the previous two sections that the empirical tensor concentrates around its population counter part and that the population tensor is well-approximated by a rank one tensor. We combine these two results along with a perturbation argument to provide guarantees for the DeepTD algorithm.

Theorem 4.8 (Main theorem) *Consider a CNN model* $\boldsymbol{x} \mapsto f_{CNN}(\boldsymbol{x})$ *of the form (2.1) consisting of* $D \ge 2$ *layers with convolutional kernels* $\boldsymbol{k}^{(1)}, \boldsymbol{k}^{(2)}, \ldots, \boldsymbol{k}^{(D)}$ *of lengths* $d_1, d_2, \ldots, d_D$*. Let* $\boldsymbol{x} \in \mathbb{R}^p$ *be a Gaussian*

random vector distributed as $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_p)$ *with the corresponding labels* $y = f_{CNN}(\boldsymbol{x})$ *generated by the CNN model and* $\boldsymbol{X} := \mathcal{T}(\boldsymbol{x})$ *the corresponding tensorized input. Suppose the data set consists of* n *training samples where the feature vectors* $\boldsymbol{x}_i \in \mathbb{R}^p$ *are distributed i.i.d.* $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_p)$ *with the corresponding labels* $y_i = f_{CNN}(\boldsymbol{x}_i)$ *generated by the same CNN model and* $\boldsymbol{X}_i := \mathcal{T}(\boldsymbol{x}_i)$ *the corresponding tensorized input. Also, assume the activations are* 1*-Lipschitz (i.e.* $|\phi'_\ell(z)| \le 1$*), are* S*-smooth per Definition 4.5 and the convolutional kernels are* μ*-diffused per Definition 4.6. Then the empirical tensor* $\boldsymbol{T}_n := \frac{1}{n}\sum_{i=1}^n (y_i - y_{avg})\boldsymbol{X}_i$ *and the rank one approximation* $\boldsymbol{L}_{CNN} = \alpha_{CNN} \bigotimes_{\ell=1}^D \boldsymbol{k}^{(\ell)}$ *to the population tensor* $\boldsymbol{T} = \mathbb{E}[y\boldsymbol{X}]$ *obey*

$$\left\|\boldsymbol{T}_n - \boldsymbol{L}_{CNN}\right\| \le \prod_{\ell=1}^D \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2} \left(c\frac{\sqrt{\left(\sum_{\ell=1}^D d_\ell\right)\log D} + t}{\sqrt{n}} + \sqrt{8\pi}\mu S \cdot \sup_\ell \prod_{i=1}^\ell \left\|\boldsymbol{k}^{(i)}\right\|_{\ell_2} \frac{D}{\sqrt{\min_\ell d_\ell}} \right),$$

with probability at least $1 - 5e^{-\min\left(t^2, t\sqrt{n}, n\right)}$*, where* $c > 0$ *is an absolute constant. Furthermore, the DeepTD estimates of the convolutional kernels given by (3.2) using the empirical tensor* $\boldsymbol{T}_n$ *obeys*

$$\frac{\prod_{\ell=1}^D \left|\left\langle \boldsymbol{k}^{(\ell)}, \hat{\boldsymbol{k}}^{(\ell)} \right\rangle\right|}{\prod_{\ell=1}^D \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}} \ge 1 - \frac{2}{\alpha_{CNN}} \left(c\frac{\sqrt{\left(\sum_{\ell=1}^D d_\ell\right)\log D} + t}{\sqrt{n}} + \sqrt{8\pi}\mu S \cdot \sup_\ell \prod_{i=1}^\ell \left\|\boldsymbol{k}^{(i)}\right\|_{\ell_2} \frac{D}{\sqrt{\min_\ell d_\ell}} \right),$$

with probability at least $1 - 5e^{-\min\left(t^2, t\sqrt{n}, n\right)}$*, where* $c > 0$ *is an absolute constant.*

The above theorem is our main result on learning a non-overlapping CNN with a single kernel at each layer. It demonstrates that estimates $\hat{\boldsymbol{k}}^{(\ell)}$ obtained by DeepTD have significant inner product with the ground truth kernels $\boldsymbol{k}^{(\ell)}$ with high probability, using only few samples. Indeed, similar to the discussion after Theorem 4.7 assuming the activations are sufficiently smooth and the convolutional kernels are unit norm and sufficiently diffused, the theorem above can be simplified as follows

$$\frac{\prod_{\ell=1}^D \left|\left\langle \boldsymbol{k}^{(\ell)}, \hat{\boldsymbol{k}}^{(\ell)} \right\rangle\right|}{\prod_{\ell=1}^D \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}} \ge 1 - c\left(\frac{\sqrt{\left(\sum_{\ell=1}^D d_\ell\right)\log D}}{\sqrt{n}} + \frac{D}{\sqrt{\min_\ell d_\ell}} \right).$$

Thus the kernel estimates obtained via DeepTD are well aligned with the true kernels as soon as the number of samples scales with the total number of parameters in the model and the length of the convolutional kernels (i.e. the size of the batches) scales quadratically with the depth of the network.

5 Resolving sign and scaling ambiguities

We note that DeepTD operates by accurately approximating the rank one tensor $\bigotimes_{\ell=1}^D \boldsymbol{k}^{(\ell)}$ from data. Therefore, DeepTD can only recover the convolutional kernels up to Sign/Scale Ambiguities (SSA). In general, it may not be possible to recover the ground truth kernels from the training data. For instance, when activations are ReLU, the norms of the kernels cannot be estimated from data as multiplying a kernel and dividing another by the same positive scalar leads to the same training data. However, we can try to learn a good approximation $\hat{f}_{\text{CNN}}()$ of the network $f_{\text{CNN}}()$ to minimize the risk $\mathbb{E}[(f_{\text{CNN}}(\boldsymbol{x}) - \hat{f}_{\text{CNN}}(\boldsymbol{x}))^2]$.

To this aim, we introduce Centered Empirical Risk Minimization (CERM) which is a slight modification of Empirical Risk Minimization (ERM). Let us first describe how finding a good $\hat{f}_{\text{CNN}}()$ can be formulated with CERM. Given n i.i.d. data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \sim (\boldsymbol{x}, y)$, and a function class $\mathcal{F}$, CERM applies ERM after centering the residuals. Given $f \in \mathcal{F}$, define the average residual function $r_{\text{avg}}(f) = \frac{1}{n}\sum_{i=1}^n y_i - f(\boldsymbol{x}_i)$. We define the Centered Empirical Risk Minimizer as

$$\begin{aligned}
\hat{f} &= \min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n (y_i - f(\boldsymbol{x}_i) - r_{\text{avg}}(f))^2, \\
&= \min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n (y_i - f(\boldsymbol{x}_i) - \mathbb{E}[(y_i - f(\boldsymbol{x}_i))])^2 - \frac{1}{n^2}\left(\sum_{i=1}^n y_i - f(\boldsymbol{x}_i) - \mathbb{E}[(y_i - f(\boldsymbol{x}_i))]\right)^2. \qquad (5.1)
\end{aligned}$$

The remarkable benefit of CERM over ERM is the fact that, the learning rate doesn't suffer from the label or function bias. This is in similar nature to the DeepTD algorithm that applies label centering. In the proofs (in particular Section 9.5, Theorem 9.17) we provide a generalization bound on the CERM solution (5.1) in terms of the Lipschitz covering number of the function space. While (5.1) can be used to learn all kernels, it does not provide an efficient algorithm. Instead, we will use CERM to resolve SSA after estimating the kernels via DeepTD. Interestingly, this approach only requires a few ($\mathcal{O}(D)$) extra training samples. Inspired from CERM, in Section 5.1, we propose a greedy algorithm to address SSA. We will apply CERM to the following function class with bounded kernels,

$$\mathcal{F}_{\hat{\boldsymbol{k}},B} := \Big\{ f : \mathbb{R}^p \mapsto \mathbb{R} \;\Big|\; \text{f is a CNN function of the form (2.1) with kernels } \{\beta_\ell \hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^{D} \text{ with } |\beta_\ell| \le B^{\frac{1}{D}} \Big\}. \tag{5.2}$$

In words this is the function class of all CNN functions with kernels the same as those obtained by DeepTD up to sign/scale ambiguities $\{\beta_\ell\}_{\ell=1}^{D}$ where the maximum scale ambiguity is B.

Theorem 5.1 *Let $f_{CNN}()$ be defined via (2.1) with convolutional kernels $\{\boldsymbol{k}^{(\ell)}\}_{\ell=1}^{D}$ obeying $\|\boldsymbol{k}^{(\ell)}\|_{\ell_2} \le B^{1/D}$ for some $B > 0$ and consider the function class $\mathcal{F}_{\hat{\boldsymbol{k}},B}$ above with the same choice of B. Assume we have n i.i.d. samples $(\boldsymbol{x}_i, y_i) \sim (\boldsymbol{x}, y)$ where $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_p)$ and $y = f_{CNN}(\boldsymbol{x})$. Suppose for some $\varepsilon \le B$,*

$$n \ge cB^2 D \log\left(\frac{CDBp}{\epsilon}\right) \max\left(\frac{1}{\epsilon}, \frac{1}{\epsilon^2}\right),$$

holds for fixed numerical constants $c, C > 0$. Then the solution $\hat{f}$ to the CERM problem (5.1) obeys

$$\mathbb{E}\Big[\big(\hat{f}(\boldsymbol{x}) - f_{CNN}(\boldsymbol{x})\big)^2 \Big] \le \min_{f \in \mathcal{F}_{\hat{\boldsymbol{k}},B}} \mathbb{E}[(f(\boldsymbol{x}) - f_{CNN}(\boldsymbol{x}))^2] + \varepsilon. \tag{5.3}$$

on a new sample $\boldsymbol{x} \in \mathcal{N}(0, \boldsymbol{I}_p)$ with probability at least $1 - e^{-\gamma n} - 4n\exp(-p)$ with $\gamma > 0$ an absolute constant.

The above theorem states that CERM finds the sign/scale ambiguity that accurate estimates the labels on new data as long as the number of samples which are used in CERM exceeds the depth of the network by constant/log factors. In the next section we present a greedy heuristic for finding the CERM estimate.

5.1 Greedy algorithm for resolving sign and scale ambiguities

In order to resolve SSA, inspired from CERM, we propose Algorithm 1 which operates over the function class,

$$\mathcal{F}_{\hat{\boldsymbol{k}}} := \Big\{ \gamma f : \mathbb{R}^p \mapsto \mathbb{R} \;\Big|\; \text{f is a CNN of the form (2.1) with kernels } \{\beta_\ell \hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^{D} \text{ with } \beta_\ell \in \{1, -1\},\ \gamma \ge 0 \Big\}. \tag{5.4}$$

It first determines the signs β_ℓ by locally optimizing the kernels and then finds a global scaling $\gamma > 0$. In the first phase, the algorithm attempts to maximize the correlation between the centered labels $y_{c,i} = y_i - n^{-1}\sum_{i=1}^{n} y_i$ and the $\hat{f}_{\text{CNN}}()$ predictions given by $\hat{y}_{c,i} = \hat{y}_i - n^{-1}\sum_{i=1}^{n} \hat{y}_i$. It goes over all kernels one by one and it flips a kernel ($\hat{\boldsymbol{k}}^{(\ell)} \to -\hat{\boldsymbol{k}}^{(\ell)}$) if flipping increases the correlation. This process goes on as long as there is an improvement. Afterwards, we use a simple linear regression to get the best scaling γ by minimizing the centered empirical loss $\sum_{i=1}^{n}(y_{c,i} - \gamma\hat{y}_{c,i})^2$. While our approach is applicable to arbitrary activations, it is tailored towards homogeneous activations ($\phi(cx) = c\phi(x)$). The reason is that for homogeneous activations, function classes (5.2) and (5.4) coincide and a single global scaling γ is sufficient. Note that ReLU and the identity activation (i.e. no activation) are both homegeneous, in fact they are elements of a larger homogeneous activation family named Leaky ReLU. Leaky ReLU is parametrized by some scalar $0 \le \beta \le 1$ and defined as follows

$$LReLU(x) = \begin{cases} x \text{ if } x \ge 0, \\ \beta x \text{ if } x < 0. \end{cases}$$

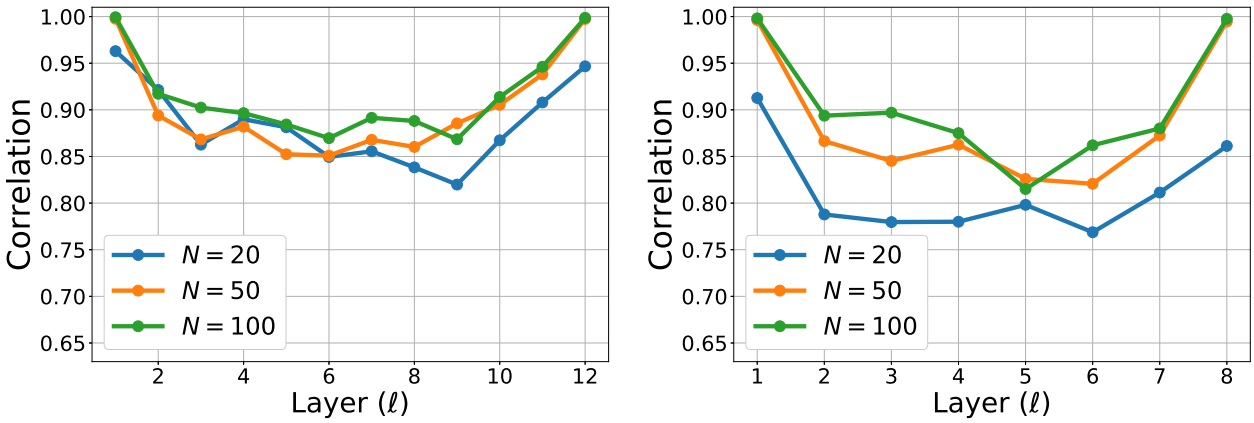
As we shall discuss in our numerical experiments in Section 6, Algorithm 1 works well for different depths ($D = 4, 8, 12$) and kernel widths $2 \le d_\ell \le 10$. We would like to point out however that it appears that this local search heuristic can indeed get stuck in a suboptimal local minima even for a depth 4 convolutional network (see Figure 3 and the associated discussion). Developing efficient algorithms that can reliably construct a good $\hat{f}_{\text{CNN}}()$ from $\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^{D}$ is an interesting direction for future research.

Algorithm 1 Greedily algorithm for resolving sign/scale ambiguities for Leaky ReLU activations.

procedure MAXCORR
Inputs: Data $(y_i, \boldsymbol{x}_i)_{i=1}^n$, estimates $\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^D$.
  $\rho_{\max} \leftarrow |\mathrm{Corr}(\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^D, 0)|$, FLIP←TRUE.
  while FLIP **do**
    FLIP←FALSE.
    for $1 \le \ell \le D$ **do**
      $\rho \leftarrow |\mathrm{Corr}(\{\hat{\boldsymbol{k}}^{(1)}, \ \ldots, \ -\hat{\boldsymbol{k}}^{(\ell)}, \ \ldots, \ \hat{\boldsymbol{k}}^{(D)}\}, 0)|$.
      if $\rho > \rho_{\max}$ **then**
        $\rho_{\max} \leftarrow \rho$
        $\hat{\boldsymbol{k}}^{(\ell)} \leftarrow -\hat{\boldsymbol{k}}^{(\ell)}$
        FLIP←TRUE
  $\gamma \leftarrow \mathrm{Corr}(\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^D, 1)$.
return kernels $\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^D$, scaling γ.

Algorithm 2 Return the correlation between centered labels.

procedure CORR($\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^D$, opt)
  $\hat{y}_i \leftarrow f_{\mathrm{CNN}}(\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^D; \boldsymbol{x}_i)$.
  $y_{\mathrm{c},i} \leftarrow y_i - \frac{1}{n}\sum_{i=1}^n y_i$ and $\hat{y}_{\mathrm{c},i} \leftarrow \hat{y}_i - \frac{1}{n}\sum_{i=1}^n \hat{y}_i$.
  $\rho \leftarrow \sum_{i=1}^n y_{\mathrm{c},i}\hat{y}_{\mathrm{c},i}$.
return ρ **if** opt = 0, $\rho/(\sum_{i=1}^n \hat{y}_{\mathrm{c},i}^2)$ **if** opt = 1.



(a) $d = 2$, $D = 12$.

(b) $d = 3$, $D = 8$.

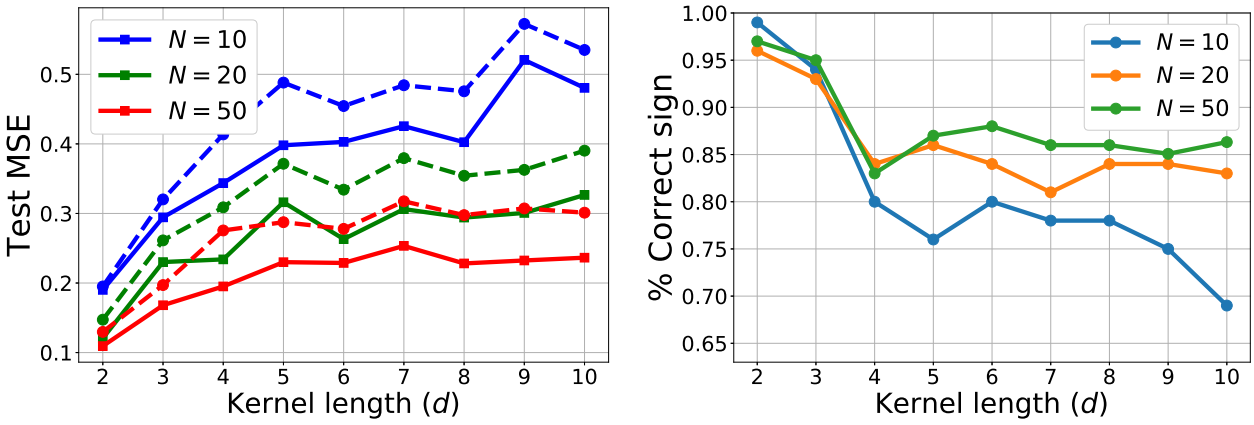
Figure 2: Correlations ($\mathrm{corr}(\hat{\boldsymbol{k}}^{(\ell)}, \boldsymbol{k}^{(\ell)}) = \left|\left\langle \hat{\boldsymbol{k}}^{(\ell)}, \boldsymbol{k}^{(\ell)} \right\rangle\right|$) between the DeepTD estimate and the ground truth kernels for different layers and various over-sampling ratios ($N = \frac{n}{\sum_{\ell=1}^D d_\ell}$).

6 Numerical experiments

Our goal in this section is to numerically corroborate the theoretical predictions of Section 4. To this aim we use a CNN model of the form (2.1) with D layers and ReLU activations and set the kernel lengths to be all equal to each other i.e. $d_4 = \ldots = d_1 = d$. We use the identity activation for the last layer (i.e. $\phi_D(z) = z$) with the exception of the last experiment where we use a ReLU activation (i.e. $\phi_D(z) = \max(0, z)$). We conducted our experiments in Python using the Tensorly library for the tensor decomposition in DeepTD [21].

	$d=2,\ D=12$			$d=3,\ D=8$		
	$N=20$	$N=50$	$N=100$	$N=20$	$N=50$	$N=100$
% Correct sign	0.83	0.95	0.93	0.65	0.87	0.94
Test loss (greedy)	0.53	0.45	0.43	0.62	0.48	0.42
Test loss (oracle)	0.50	0.44	0.41	0.58	0.44	0.40

Table 1: Test performance of DeepTD estimates (Test MSE = $\mathbb{E}\left[\left(\boldsymbol{y}-\hat{f}_{\mathrm{CNN}}(\boldsymbol{x})\right)^2\right]/\mathbb{E}[\boldsymbol{y}^2]$). Greedy is the test loss when the signs of $\hat{\boldsymbol{k}}^{(\ell)}$ are determined by Algorithm 1. Oracle is the test loss by picking the signs to ensure non-negative correlation with the ground truth kernels.



(a) (b)

Figure 3: a) The solid lines are the Test MSE ($\mathbb{E}\left[\left(\boldsymbol{y}-\hat{f}_{\mathrm{CNN}}(\boldsymbol{x})\right)^2\right]/\mathbb{E}[\boldsymbol{y}^2]$) error when using the correct kernel signs (oracle). The dashed lines are the Test MSE by determining the kernel signs with Algorithm 1. b) Fraction of times Algorithm 1 successfully identified all signs.

Each curve in every figure is obtained by averaging 100 independent realizations of the same CNN learning procedure. Similar to our theory, we use Gaussian data points $\boldsymbol{x}$ and ground truth labels $\boldsymbol{y}=f_{\mathrm{CNN}}(\boldsymbol{x})$.

We conduct two sets of experiments: The first set focuses on larger values of depth D and the second set focuses on larger values of width d. In all experiments kernels are generated with random Gaussian entries and are normalized to have unit Euclidean norm. For the ReLU activation if one of the kernels have all negative entries, the output is trivially zero and learning is not feasible. To address this, we consider *operational* networks where at least 50% of the training labels are nonzero. Here, the number 50% is arbitrarily chosen and we verified that similar results hold for other values. To ensure the kernels do not have all negative entries and the network is operational we use a rejection sampling scheme. That is, if our generated network does not obey the 50% assumption we discard that experiment and generate a new one. In practice, this is mainly an issue for small d values only as the chance of an all-negative kernel is equal to 2^{-d}. Finally, to study the effect of finite samples, we let the sample size grow proportional to the total degrees of freedom $\sum_{\ell=1}^{D} d_\ell$. In particular, we set an oversampling factor $N=\frac{n}{\sum_{\ell=1}^{D} d_\ell}$ and carry out the experiments for $N\in\{10,20,50,100\}$. While our theory requires $N\gtrsim \log D$, in our experiments, we typically observe that improvement is marginal after $N=50$.

In Figure 2, we consider two networks with $d=2, D=12$ and $d=3, D=8$ configurations. We plot the absolute correlation between the ground truth and the estimates as a function of layer depth. For each hidden layer $1\le \ell\le D$, our correlation measure (y-axis) is

$$\mathrm{corr}(\hat{\boldsymbol{k}}^{(\ell)},\boldsymbol{k}^{(\ell)})=\left|\left\langle \hat{\boldsymbol{k}}^{(\ell)},\boldsymbol{k}^{(\ell)}\right\rangle\right|.$$

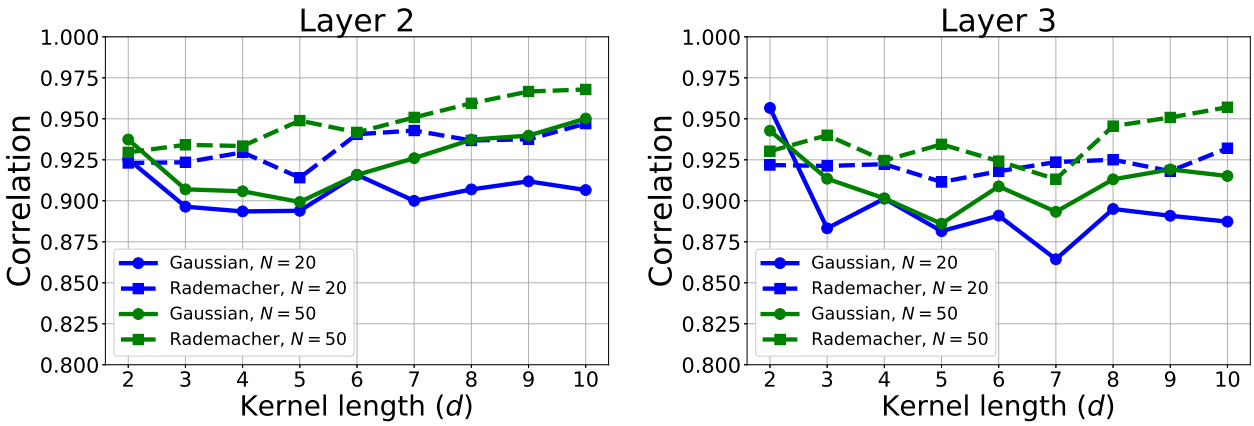


Figure 4: Correlations ($\text{corr}(\hat{\boldsymbol{k}}^{(\ell)}, \boldsymbol{k}^{(\ell)}) = \left|\left\langle \hat{\boldsymbol{k}}^{(\ell)}, \boldsymbol{k}^{(\ell)} \right\rangle\right|$) when the ground truth kernels are Gaussian vs Rademacher. Gaussian is more spiky and results in lower correlation.

This number is between 0 and 1 as the kernels and their estimates both have unit norm. We observe that for both $d = 2$ and $d = 3$, DeepTD consistently achieves correlation values above 75% for $N = 20$. While our theory requires d to scale quadratically with depth i.e. $d \gtrsim D^2$, we find that even small d values work well in our experiments. The effect of sample size becomes evident by comparing $N = 20$ and $N = 50$ for the input and output layers ($\ell = 1, \ell = D$). In this case $N = 50$ achieves perfect correlation. Interestingly, correlation values are smallest in the middle layers. In fact this even holds when N is large suggesting that the rank one approximation of the population tensor provides worst estimates for the middle layers.

Table 1 shows the test performance of DeepTD using an oracle and Algorithm 1. Our test loss is defined as the normalized regression loss

$$\text{Test MSE} = \frac{\mathbb{E}\left[\left(\boldsymbol{y} - \hat{f}_{\text{CNN}}(\boldsymbol{x})\right)^2\right]}{\mathbb{E}[\boldsymbol{y}^2]}, \tag{6.1}$$

where $\hat{f}_{\text{CNN}}$ is our estimate of $f_{\text{CNN}}()$. Here, the oracle knows the correct signs (i.e. guaranteed to have $\left\langle \hat{\boldsymbol{k}}^{(\ell)}, \boldsymbol{k}^{(\ell)} \right\rangle \geq 0$ for all $1 \leq \ell \leq D$) but the scale is found by solving a regression problem in a similar fashion to Line 12 of Algorithm 1. We observe that for $N \geq 50$, signs of all kernels are correctly identified with $\gtrsim 90\%$ accuracy. This table also demonstrates that there is a minor difference in test loss between the oracle and our proposed greedy algorithm. We would like to note that a test loss of $\approx 40\%$ may appear large, especially as we achieve high kernel correlations $\gtrsim 85\%$ (for $N = 100$). We believe that this behavior is due to the depth of the network which aggregates the small estimation errors of individual layers and results in a larger error in the overall function estimate $\hat{f}_{\text{CNN}}$.

For the remaining experiments, we set $D = 4$ and vary the width d from 2 to 10. N is varied between 10 to 50. Figure 3 summarizes the performance of the overall algorithm which is composed of DeepTD and Algorithm 1. In Figure 3a, we observe that small d values $d = 2, 3$ achieve the best test performance. Performance appears to saturate after $d \geq 7$ both for greedy algorithm (dashed lines) and oracle (solid lines) which has the correct sign information. The greedy algorithm is consistently worse than the oracle which indicates that it may have reached a local minima (this claim is verified on several examples where the oracle and greedy approaches have differing signs). While signs are often correctly identified in Figure 3b, the greedy algorithm fails in 15% of the instances even for large N which indicates the need for better heuristics heuristics to resolve sign and scale ambiguities.

Next we study our theoretical predictions regarding DeepTD. In particular, Figure 4 assesses the impact of diffuseness by plotting correlations for Layers 2 and 3. This is done by contrasting different kernel generation models, namely, kernels with i.i.d. standard normal entries and Rademacher (± 1) entries (normalized to have

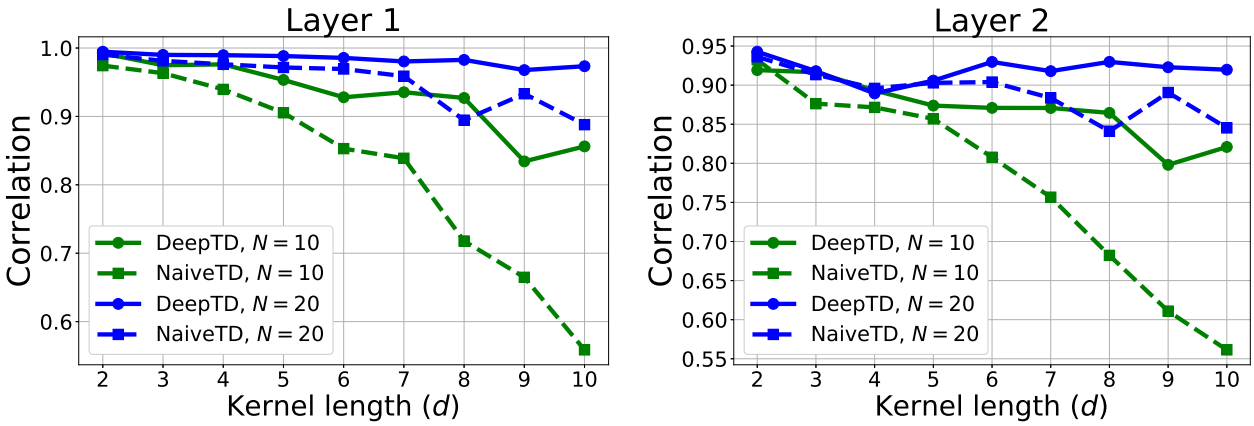


Figure 5: DeepTD estimate vs NaiveTD estimate when final activation is ReLU. Bias of NaiveTD results in significantly worse performance.

unit Euclidean norm). Rademacher kernels are strictly more diffused than Gaussian kernels as all entries have the same magnitude by construction. We observe that Rademacher kernels (dashed lines) achieve consistently higher correlations which is supported by our theory.

In Figure 5, we use a ReLU activation in the final layer and assess the impact of the centering procedure of the DeepTD algorithm which is a major theme throughout the paper. We define the NaiveTD algorithm which solves (3.2) without centering in the empirical tensor i.e.

$$\hat{\boldsymbol{k}}^{(1)}, \ldots, \hat{\boldsymbol{k}}^{(D)} = \underset{\boldsymbol{v}_1 \in \mathbb{R}^{d_1}, \boldsymbol{v}_2 \in \mathbb{R}^{d_2}, \ldots, \boldsymbol{v}_D \in \mathbb{R}^{d_D}}{\arg\max} \left\langle \frac{1}{n} \sum_{i=1}^{n} y_i \boldsymbol{X}_i, \bigotimes_{\ell=1}^{D} \boldsymbol{v}_\ell \right\rangle \quad \text{subject to} \quad \|\boldsymbol{v}_1\|_{\ell_2} = \ldots = \|\boldsymbol{v}_D\|_{\ell_2} = 1. \tag{6.2}$$

Since the activation of the final layer is ReLU, the output has a clear positive bias in expectation which will help demonstrating the importance of centering. We find that for smaller oversampling factors of $N = 10$ or $N = 20$, DeepTD has a visibly better performance compared with NaiveTD. The correlation difference is persistent among different layers (we plotted only Layers 1 and 2) and appears to grow with increase in the kernel size d.

Finally, in Figure 6, we assess the impact of activation nonlinearity by comparing the ReLU and identity activations in the final layer. We plot the first and final layer correlations for this setup. While the correlation performances of the first layer are essentially identical, the ReLU activation (dashed lines) achieves significantly lower correlation at the final layer. This is not surprising as the final layer passes through an additional nonlinearity.

7 Related work

Our work is closely related to the recent line of papers on neural networks as well as tensor decompositions. We briefly discuss this related literature.

Neural networks: Learning neural networks is a nontrivial task involving non-linearities and non-convexities. Consequently, existing theory works consider different algorithms, network structures and assumptions. A series of recent work focus on learning zero or one-hidden layer fully connected neural networks with random inputs and planted models [6, 13, 15, 17, 25, 27, 32, 33, 41]. Other recent publications [1, 11, 12, 27, 40] consider the problem of learning a convolutional neural network with 1-hidden layer. In particular, [1, 11, 12] focuses on learning non-overlapping networks as in this paper (albeit in the limit of infinite training data). The papers above either focus on characterizing the optimization landscape, or population landscape or providing exact convergence guarantees for gradient descent. In comparison, in this paper we focus on approximate convergence

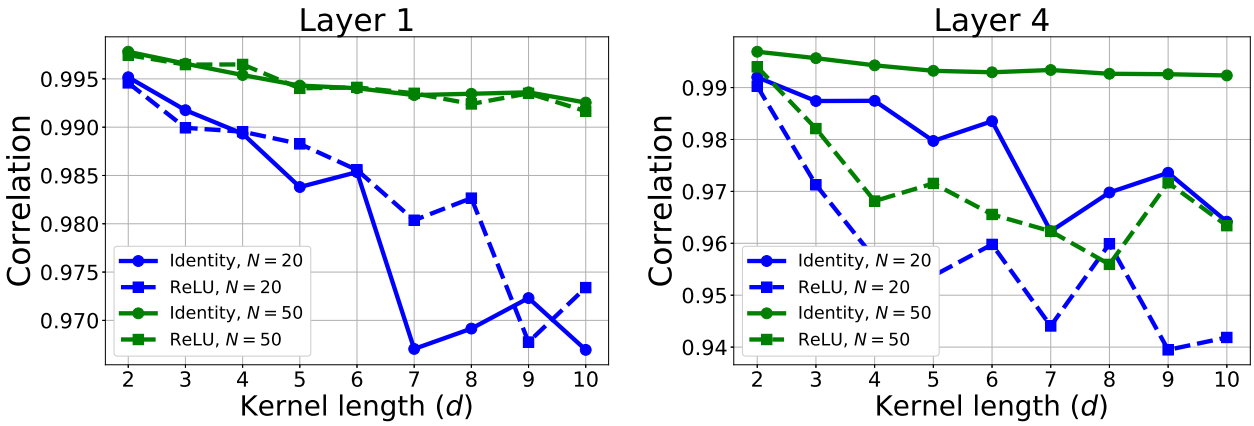


Figure 6: Comparison of performance of DeepTD when the final activation is ReLU in lieu of the identity activation.

guarantees using tensor decompositions for arbitrary deep networks. A few recent publications [30, 34, 35] consider the training problem when the network is over-parametrized and study the over-fitting ability of such networks. Closer to our work, [24] and [4] considers provable algorithms for deep networks using layer-wise algorithms. In comparison to our work, [24] applies to a very specific generative model that assumes a discrete data distribution and studies the population loss in lieu of the empirical loss. Arora et al. [4] studies deep models but uses activation functions that are not commonly used and assumes random network weights. In comparison, we work with practical activations and do not assume the weights are generated at random.

Tensor decomposition: Tensors are powerful tools to model a wide variety of big-data problems [2, 31]. Recent years have witnessed a growing interest in tensor decomposition techniques to extract useful latent information from the data [2, 14]. The connection between tensors and neural networks have been noticed by several papers [7, 8, 19, 20, 26]. Cohen et al. [7, 8] relate convolutional neural networks and tensor decompositions to provide insights on the expressivity of CNNs. Mondelli and Montanari [26] connects the hardness of learning shallow networks to tensor decompositions. Closer to this paper, Janzamin et al. [19] and Zhong et al. [41] apply tensor decomposition on *one*-hidden layer, fully connected networks to approximately learn the latent weight matrices.

8 Conclusion

In this paper we studied a multilayer CNN model with depth D. We assumed a non-overlapping structure where each layer has a single convolutional kernel and has stride length equal to the dimension of its kernel. We establish a connection between approximating the CNN kernels and higher order tensor decompositions. Based on this connection we proposed an algorithm for simultaneously learning all the kernels called the Deep Tensor Decomposition (DeepTD) algorithm. This algorithm builds a D-way tensor based on the training data and applies a rank one tensor factorization algorithm to this tensor to simultaneously estimate all the convolutional kernels. Assuming the input data is distributed i.i.d. according to a Gaussian model with corresponding output generated by a planted set of convolutional kernels we prove DeepTD can approximately learn all the kernels with a near minimal number of training data. A variety of numerical experiments complement our theoretical findings.

9 Proofs

In this section we will prove our main results. Throughout, for a random variable X, we use $\mathbf{zm}(X)$ to denote $X - \mathbb{E}[X]$. Simply stated, $\mathbf{zm}(X)$ is the *centered* version of X. For a random vector/matrix/tensor $\boldsymbol{X}$, $\mathbf{zm}(\boldsymbol{X})$ denotes the vector/matrix/tensor obtained by applying the $\mathbf{zm}()$ operation to each entry. For a tensor $\boldsymbol{T}$ we use $\|\boldsymbol{T}\|_F$ to denote the square root of the sum of squares of the entries of the tensor. Stated differently, this is the Euclidean norm of a vector obtained by rearranging the entries of the tensor. Throughout we use c, c_1, c_2, and C to denote fixed numerical constants whose values may change from line to line. We begin with some useful definitions and lemmas.

9.1 Useful concentration lemmas and definitions

In this section we gather some useful definitions and well-known lemmas that will be used frequently throughout our concentration arguments.

Definition 9.1 (Orlicz norms) *For a scalar random variable Orlicz-a norm is defined as*

$$\|X\|_{\psi_a} = \sup_{k \geq 1} k^{-1/a} (\mathbb{E}[|X|^k])^{1/k}$$

Orlicz-a norm of a vector $\boldsymbol{x} \in \mathbb{R}^p$ is defined as $\|\boldsymbol{x}\|_{\psi_a} = \sup_{\boldsymbol{v} \in \mathcal{B}^p} \|\boldsymbol{v}^T \boldsymbol{x}\|_{\psi_a}$ where $\mathcal{B}^p$ is the unit ℓ_2 ball. The sub-exponential norm is the function $\|\cdot\|_{\psi_1}$ and the sub-gaussian norm the function $\|\cdot\|_{\psi_2}$.

We now state a few well-known results that we will use throughout the proofs. This results are standard and are stated for the sake of completeness. The first lemma states that the product of sub-gaussian random variables are sub-exponential.

Lemma 9.2 *Let X, Y be subgaussian random variables. Then $\|XY\|_{\psi_1} \leq \|X\|_{\psi_2} \|Y\|_{\psi_2}$.*

The next lemma connects Orlicz norms of sum of random variables to the sum of the Orlicz norm of each random variable.

Lemma 9.3 *Suppose X, Y are random variables with bounded $\|\cdot\|_{\psi_a}$ norm. Then $\|X + Y\|_{\psi_a} \leq 2 \max\{\|X\|_{\psi_a}, \|Y\|_{\psi_a}\}$. In particular $\|X - \mathbb{E}\, X]\|_{\psi_a} \leq 2\|X\|_{\psi_a}$.*

The lemma below can be easily obtained by combining the previous two lemmas.

Lemma 9.4 *Let X, Y be subgaussian random variables. Then $\|\mathbf{zm}(XY)\|_{\psi_1} \leq 2\|X\|_{\psi_2} \|Y\|_{\psi_2}$.*

Finally, we need a few standard chaining definitions.

Definition 9.5 (Admissible sequence [37]) *Given a set T an admissible sequence is an increasing sequence $\{A_n\}_{n=0}^{\infty}$ of partitions of T such that $|A_n| \leq N_n$ where $N_0 = 1$ and $N_n = 2^{2^n}$ for all $n \geq 1$.*

For the following discussion $\Delta_d(A_n(t))$, will be the diameter of the set $S \in A_n$ that contains t, with respect to the d metric.

Definition 9.6 (γ_a functional [37]) *Given $a > 0$, and a metric space (T, d) we define*

$$\gamma_a(T, d) = \inf \sup_{t \in T} \sum_{n \geq 0} 2^{n/a} \Delta_d(A_n(t)),$$

where the infimum is taken over all admissible sequences.

The following lemma upper bounds γ_α functional with covering numbers of T. The reader is referred to Section 1.2 of [36], Equation (2.3) of [10], and Lemma D.17 of [27].

Lemma 9.7 (Dudley's entropy integral) *Let $N(\varepsilon)$ be the ε covering number of the set T with respect to the d metric. Then*

$$\gamma_\alpha(T, d) \leq C_\alpha \int_0^\infty \log^{1/\alpha} N(\varepsilon) d\varepsilon,$$

where $C_\alpha > 0$ depends only on $\alpha > 0$.

9.2 Concentration of the empirical tensor (Proof of Theorem 4.2)

To prove this theorem, first note that given labels $\{y_i\}_{i=1}^n \sim y$ and their empirical average $y_{\text{avg}} = n^{-1}\sum_{i=1}^n y_i$, we have $\mathbb{E}[y_{\text{avg}}] = \mathbb{E}[y]$. Hence $y - y_{\text{avg}} = \mathbf{zm}(y - y_{\text{avg}})$ and we can rewrite the empirical tensor as follows

$$\begin{aligned}\boldsymbol{T}_n :=& \frac{1}{n}\sum_{i=1}^n (y_i - y_{\text{avg}})\boldsymbol{X}_i \\ =& \frac{1}{n}\sum_{i=1}^n \mathbf{zm}(y_i - y_{\text{avg}})\boldsymbol{X}_i \\ =& \frac{1}{n}\sum_{i=1}^n \mathbf{zm}(y_i)\boldsymbol{X}_i - \mathbf{zm}(y_{\text{avg}})\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i\right). \end{aligned} \tag{9.1}$$

Recall that the population tensor is equal to $\boldsymbol{T} := \mathbb{E}[y_i\boldsymbol{X}_i]$. Furthermore, $\mathbb{E}[\mathbb{E}[y_i]\boldsymbol{X}_i] = \mathbb{E}[y_i]\,\mathbb{E}[\boldsymbol{X}_i] = 0$. Thus the population tensor can alternatively be written as $\boldsymbol{T} = E[\mathbf{zm}(y_i)\boldsymbol{X}_i] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\mathbf{zm}(y_i)\boldsymbol{X}_i]$. Combining the latter with (9.1) we conclude that

$$\begin{aligned}\boldsymbol{T}_n - \boldsymbol{T} =& \frac{1}{n}\sum_{i=1}^n (\mathbf{zm}(y_i)\boldsymbol{X}_i - \mathbb{E}[\mathbf{zm}(y_i)\boldsymbol{X}_i]) - \mathbf{zm}(y_{\text{avg}})\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i\right), \\ =& \frac{1}{n}\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(y_i)\boldsymbol{X}_i) - \mathbf{zm}(y_{\text{avg}})\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i\right), \\ =& \frac{1}{n}\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(y_i)\boldsymbol{X}_i) - \left(\frac{1}{n}\sum_{i=1}^n \mathbf{zm}(y_i)\right)\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i\right). \end{aligned}$$

Now using the triangular inequality for tensor spectral norm we conclude that

$$\|\boldsymbol{T}_n - \boldsymbol{T}\| \le \left\|\frac{1}{n}\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(y_i)\boldsymbol{X}_i)\right\| + \left\|\left(\frac{1}{n}\sum_{i=1}^n \mathbf{zm}(y_i)\right)\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i\right)\right\|.$$

We now state two lemmas to bound each of these terms. The proofs of these lemmas are defered to Sections 9.2.1 and 9.2.2.

Lemma 9.8 *For $i = 1, 2, \ldots, n$ let $\boldsymbol{x}_i \in \mathbb{R}^p$ be i.i.d. random Gaussian vectors distributed as $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_p)$. Also let $\boldsymbol{X}_i \in \mathbb{R}^{\otimes_{\ell=1}^D d_\ell}$ be the tensorized version of $\boldsymbol{x}_i$ i.e. $\boldsymbol{X}_i = \mathcal{T}(\boldsymbol{x}_i)$. Finally, assume $f : \mathbb{R}^p \mapsto \mathbb{R}$ is an L Lipschitz function. Furthermore, assume $n \ge \left(\sum_{\ell=1}^D d_\ell\right)\log D$ and $D \ge 2$. Then*

$$\mathbb{P}\left\{\left\|\frac{1}{n}\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(f(\boldsymbol{x}_i))\boldsymbol{X}_i)\right\| \le \frac{c_1 L}{\sqrt{n}}\left(\sqrt{\left(\sum_{\ell=1}^D d_\ell\right)\log D} + t\right)\right\} \le e^{-\min\left(t^2, t\sqrt{n}\right)},$$

holds with $c_1 > 0$ a fixed numerical constant.

Lemma 9.9 *Consider the setup of Lemma 9.8. Then*

$$\mathbb{P}\left\{\left\|\left(\frac{1}{n}\sum_{i=1}^n \mathbf{zm}(f(\boldsymbol{x}_i))\right)\left(\frac{1}{n}\sum_{i=1}^n \boldsymbol{X}_i\right)\right\| \ge \frac{c_2 t_1 L}{n}\left(\sqrt{\left(\sum_{\ell=1}^D d_\ell\right)\log D} + t_2\right)\right\} \le 2\left(e^{-t_1^2} + e^{-t_2^2}\right),$$

holds with $c_2 > 0$ a fixed numerical constant.

Combining Lemma 9.8 with $f = f_{\text{CNN}}()$, $L = \prod_{\ell=1}^D \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}$, and $c_1 = c/2$ together with Lemma 9.9 with $t_1 = \sqrt{n}$, $t_2 = t$, and $c_2 = c/2$ concludes the proof of Theorem 4.2. All that remains is to prove Lemmas 9.8 and 9.9 which are the subject of the next two sections.

9.2.1 Proof of Lemma 9.8

It is more convenient to carryout the steps of the proof on $\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(f(\boldsymbol{x}_i))\boldsymbol{X}_i)$ in liue of $\frac{1}{n}\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(f(\boldsymbol{x}_i))\boldsymbol{X}_i)$. The lemma trivally follows by a scaling by a factor $1/n$. We first write the tensor spectral norm as a supremum

$$\left|\!\left|\!\left|\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(f(\boldsymbol{x}_i))\boldsymbol{X}_i)\right|\!\right|\!\right| = \sup_{\boldsymbol{T}\in\mathcal{RO}} \left|\sum_{i=1}^n \langle \mathbf{zm}(\mathbf{zm}(f(\boldsymbol{x}_i))\boldsymbol{X}_i), \boldsymbol{T}\rangle\right|. \tag{9.2}$$

Let $\boldsymbol{Y}_i = \mathbf{zm}(f(\boldsymbol{x}_i))\boldsymbol{X}_i$. Define the random process $g(\boldsymbol{T}) = \sum_{i=1}^n \langle \mathbf{zm}(\boldsymbol{Y}_i), \boldsymbol{T}\rangle$. We claim that $g(\boldsymbol{T})$ has a mixture of subgaussian and subexponential increments (see Definition 9.1 for subgaussian and subexponential random variables). Pick two tensors $\boldsymbol{T}, \boldsymbol{H} \in \mathbb{R}^{\otimes_{\ell=1}^D d_\ell}$. Increments of g satisfy the linear relation

$$g(\boldsymbol{T}) - g(\boldsymbol{H}) = \sum_{i=1}^n \langle \mathbf{zm}(\boldsymbol{Y}_i), \boldsymbol{T} - \boldsymbol{H}\rangle.$$

By construction $\mathbb{E}[g(\boldsymbol{T}) - g(\boldsymbol{H})] = 0$. We next claim that $\boldsymbol{Y}_i$ is a sub-exponential vector. Consider a tensor $\boldsymbol{T}$ with unit length $\|\boldsymbol{T}\|_F = 1$ i.e. the sum of squares of entries are equal to one. We have $\langle \boldsymbol{Y}_i, \boldsymbol{T}\rangle = \mathbf{zm}(f(\boldsymbol{x}_i))\langle \boldsymbol{X}_i, \boldsymbol{T}\rangle$. $f(\boldsymbol{X}_i)$ is a Lipschitz function of a Gaussian random vector. Thus, by the concentration of Lipschitz functions of Gaussians we have

$$\mathbb{P}(|\mathbf{zm}(f(\boldsymbol{X}_i))| \geq t) \leq 2\exp(-\frac{t^2}{2L^2}). \tag{9.3}$$

This immediately implies that $\|\mathbf{zm}(f(\boldsymbol{X}_i))\|_{\psi_2} \leq cL$ for a fixed numerical constant c. Also note that $\langle \boldsymbol{X}_i, \boldsymbol{T}\rangle \sim \mathcal{N}(0,1)$ hence $\|\langle \boldsymbol{X}_i, \boldsymbol{T}\rangle\|_{\psi_2} \leq c$. These two identities combined with Lemma 9.4 implies a bound on the sub-exponential norm

$$\|\mathbf{zm}(\langle \boldsymbol{Y}_i, \boldsymbol{T}\rangle)\|_{\psi_1} \leq cL.$$

Next, we observe that $g(\boldsymbol{T}) - g(\boldsymbol{H})$ is sum of n i.i.d. sub-exponentials each obeying $\|\langle \mathbf{zm}(\boldsymbol{Y}_i), \boldsymbol{T} - \boldsymbol{H}\rangle\|_{\psi_1} \leq cL\|\boldsymbol{T} - \boldsymbol{H}\|_F$. Applying a standard sub-exponential Bernstein inequality, we conclude that

$$\mathbb{P}(|g(\boldsymbol{T}) - g(\boldsymbol{H})| \geq t) \leq 2\exp\left(-c\cdot\min\left(\frac{t}{L\|\boldsymbol{T}-\boldsymbol{H}\|_F}, \frac{t^2}{nL^2\|\boldsymbol{T}-\boldsymbol{H}\|_F^2}\right)\right), \tag{9.4}$$

holds with γ a fixed numerical constant. This tail bound implies that g is a mixed tail process that is studied by Talagrand and others [10, 37]. In particular, supremum of such processes are characterized in terms of a linear combination of Talagrand's γ_1 and γ_2 functionals (see Definition 9.6 as well as [36, 37] for an exposition). We pick the following distance metrics on tensors induced by the Frobenius norm: $d_1(\boldsymbol{T}, \boldsymbol{H}) = L\|\boldsymbol{H} - \boldsymbol{T}\|_F/c$ and $d_2(\boldsymbol{T}, \boldsymbol{H}) = \|\boldsymbol{H} - \boldsymbol{T}\|_F L\sqrt{n/c}$. We can thus rewrite (9.4) in the form

$$\mathbb{P}\big\{|g(\boldsymbol{T}) - g(\boldsymbol{H})| \geq t\big\} \leq 2\exp\left(-\min\left(\frac{t}{d_1(\boldsymbol{T},\boldsymbol{H})}, \frac{t^2}{d_2^2(\boldsymbol{T},\boldsymbol{H})}\right)\right),$$

which implies $\mathbb{P}\big\{|g(\boldsymbol{T}) - g(\boldsymbol{H})| \geq \sqrt{t}d_2(\boldsymbol{T},\boldsymbol{H}) + td_1(\boldsymbol{T},\boldsymbol{H})\big\} \leq 2\exp(-t)$. Observe that the radius of $\mathcal{RO}$ with respect to $\|\cdot\|_F$ norm is 1 hence radius with respect to d_1, d_2 metrics are $L/c, L\sqrt{n/c}$ respectively. Applying Theorem 3.5 of Dirksen [10], we obtain

$$\mathbb{P}\left\{\sup_{\boldsymbol{T}\in\mathcal{RO}} |g(\boldsymbol{T})| \geq C\left(\gamma_2(\mathcal{RO}, d_2) + \gamma_1(\mathcal{RO}, d_1) + L\sqrt{un/c} + uL/c\right)\right\} \leq e^{-u}.$$

Observe that we can use the change of variable $t = L\cdot\max\left(\sqrt{un}, u\right)$ to obtain

$$\mathbb{P}\left\{\sup_{\boldsymbol{T}\in\mathcal{RO}} |g(\boldsymbol{T})| \geq C\left(\gamma_2(\mathcal{RO}, d_2) + \gamma_1(\mathcal{RO}, d_1) + t\right)\right\} \leq \exp\left(-\min\left(\frac{t^2}{L^2 n}, \frac{t}{L}\right)\right), \tag{9.5}$$

with some updated constant $C > 0$. To conclude, we need to bound the γ_2 and γ_1 terms. To achieve this we will upper bound the γ_α functional in terms of Dudley's entropy integral which is stated in Lemma 9.7. First, let us find the ε covering number of $\mathcal{RO}$. Pick $0 < \delta \le 1$ coverings $\mathcal{C}_\ell$ of the unit ℓ_2 balls $\mathcal{B}^{d_\ell}$. These covers have size at most $(1 + 2/\delta)^{d_\ell}$. Consider the set of rank 1 tensors $\mathcal{C} = \mathcal{C}_1 \otimes \ldots \otimes \mathcal{C}_D$ with size $(1 + 2/\delta)^{\sum_{\ell=1}^D d_\ell}$. For any $\otimes_{\ell=1}^D \boldsymbol{v}_\ell \in \mathcal{RO}$, we can pick $\otimes_{\ell=1}^D \boldsymbol{u}_\ell \in \mathcal{C}$ satisfying $\|\boldsymbol{v}_\ell - \boldsymbol{u}_\ell\|_{\ell_2} \le \delta$ for all $1 \le \ell \le D$. This implies

$$\|(\{\boldsymbol{v}_\ell\}_{\ell=1}^D) - (\{\boldsymbol{u}_\ell\}_{\ell=1}^D)\|_F \le \sum_{\ell=1}^D \|(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell, \boldsymbol{u}_{\ell+1}, \ldots, \boldsymbol{u}_D) - (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\ell-1}, \boldsymbol{u}_\ell, \ldots, \boldsymbol{u}_D)\|_F \tag{9.6}$$

$$= \sum_{\ell=1}^D \|\boldsymbol{v}_\ell\|_{\ell_2} \ldots \|\boldsymbol{v}_{\ell-1}\|_{\ell_2} \|\boldsymbol{v}_\ell - \boldsymbol{u}_\ell\|_{\ell_2} \|\boldsymbol{u}_{\ell+1}\|_{\ell_2} \ldots \|\boldsymbol{u}_D\|_{\ell_2} \le D\delta. \tag{9.7}$$

Denoting Frobenius norm covering number of $\mathcal{RO}$ by $N(\varepsilon)$, this implies that, for $0 < \varepsilon \le 1$,

$$N(\varepsilon) \le (1 + 2D/\varepsilon)^{\sum_{\ell=1}^D d_\ell}.$$

Clearly, $N(\varepsilon) = 1$ for $\varepsilon \ge 1$ by picking the cover $\{0\}$. Consequently,

$$\gamma_1(\mathcal{RO}, \|\cdot\|_F) \le \int_0^1 \log N(\varepsilon) d\varepsilon \le c\left(\sum_{\ell=1}^D d_\ell\right) \log D, \;\; \gamma_2(\mathcal{RO}, \|\cdot\|_F) \le \int_0^1 \sqrt{\log N(\varepsilon)} d\varepsilon \le c\sqrt{\left(\sum_{\ell=1}^D d_\ell\right) \log D}. \tag{9.8}$$

Thus the metrics d_1, d_2 metrics are $\|\cdot\|_F$ norm scaled by a constant. Hence, their γ_α functions are scaled versions of (9.8) given by

$$\gamma_1(\mathcal{RO}, d_1) \le cL\left(\sum_{\ell=1}^D d_\ell\right) \log D, \;\; \gamma_2(\mathcal{RO}, d_2) \le cL\sqrt{n\left(\sum_{\ell=1}^D d_\ell\right) \log D}.$$

Now, observe that if $n \ge \left(\sum_{\ell=1}^D d_\ell\right) \log D$, we have $\gamma_1(\mathcal{RO}, d_1) \le cL\sqrt{n\left(\sum_{\ell=1}^D d_\ell\right) \log D}$. Substituting these in (9.5) and using $n \ge \left(\sum_{\ell=1}^D d_\ell\right) \log D$ we find

$$\mathbb{P}\left\{ \sup_{\boldsymbol{T} \in \mathcal{RO}} |g(\boldsymbol{T})| \ge C\left(L\sqrt{n\left(\sum_{\ell=1}^D d_\ell\right) \log D} + t \right) \right\} \le e^{-\min\left(\frac{t^2}{L^2 n}, \frac{t}{L}\right)}.$$

Substituting $t \to L\sqrt{n}t$ and recalling (9.2), concludes the proof.

9.2.2 Proof of Lemma 9.9

We first rewrite,

$$\left\| \left(\frac{1}{n} \sum_{i=1}^n \mathbf{zm}(f(\boldsymbol{x}_i)) \right) \left(\frac{1}{n} \sum_{i=1}^n \boldsymbol{X}_i \right) \right\| = \left| \frac{1}{n} \sum_{i=1}^n \mathbf{zm}(f(\boldsymbol{x}_i)) \right| \cdot \left\| \frac{1}{n} \sum_{i=1}^n \boldsymbol{X}_i \right\|. \tag{9.9}$$

As discussed in (9.3), $\|\mathbf{zm}(f(\boldsymbol{x}_i))\|_{\psi_2} \le cL$ for c a fixed numerical constant. Since $\boldsymbol{x}_i$'s are i.i.d the empirical sum $\boldsymbol{f}_{\text{avg}} = \frac{1}{n} \sum_{i=1}^n \mathbf{zm}(f(\boldsymbol{x}_i))$ obeys the bound $\|\boldsymbol{f}_{\text{avg}}\|_{\psi_2} \le cL/\sqrt{n}$ as well. Hence,

$$\mathbb{P}\left\{ |\boldsymbol{f}_{\text{avg}}| \ge c' \frac{t_1 L}{\sqrt{n}} \right\} \le 2e^{-t_1^2}. \tag{9.10}$$

Also note that $\frac{1}{\sqrt{n}} \sum_{i=1}^n \boldsymbol{X}_i$ is a tensor with standard normal entries, applying [38, Theorem 1] we conclude that

$$\left\| \frac{1}{\sqrt{n}} \sum_{i=1}^n \boldsymbol{X}_i \right\| \le c''\left(\sqrt{\left(\sum_{\ell=1}^D d_\ell\right) \log D} + t_2 \right) \quad \Rightarrow \quad \left\| \frac{1}{n} \sum_{i=1}^n \boldsymbol{X}_i \right\| \le c'' \frac{\sqrt{\left(\sum_{\ell=1}^D d_\ell\right) \log D} + t_2}{\sqrt{n}}, \tag{9.11}$$

holds with probability $1 - 2e^{-t_2^2}$. Combining (9.10) and (9.11) via the union bound together with (9.9) concludes the proof.

9.3 Rank one approximation of the population tensor (Proof of Theorem 4.7)

We begin the proof of this theorem by a few definitions regarding non-overlapping CNN models that simplify our exposition. For these definitions it is convenient to view non-overlapping CNNs as a tree with the root of the tree corresponding to the output of the CNN and the leaves corresponding to input features. In this visualization $D-\ell$th layer of the tree corresponds to the ℓth layer. Figure 7 depicts such a tree visualization along with the definitions discussed below.

Definition 9.10 (Path vector) *A vector $\mathbf{i} \in \mathbb{R}^{D+1}$ is a path vector if its zeroth coordinate satisfies $1 \le \mathbf{i}_0 \le p$ and for all $D-1 \ge j \ge 0$, $\mathbf{i}_{j+1}$ obeys $\mathbf{i}_{j+1} = \lceil \mathbf{i}_j / d_j \rceil$. This implies $1 \le \mathbf{i}_j \le p_j$ and $\mathbf{i}_D = 1$. We note that in the tree visualization a path vector corresponds to a path connecting a leaf (input feature) to the root of the tree (output). We use $\mathcal{I}$ to denote the set of all path vectors and note that $|\mathcal{I}| = p$. We also define $\mathbf{i}(i) \in \mathcal{I}$ be the vector whose zeroth entry is $\mathbf{i}_0 = i$. Stated differently $\mathbf{i}(i)$ is the path connecting the input feature i to the output. Given a path $\mathbf{i}$ and a p dimensional vector $\boldsymbol{v}$, we define $\boldsymbol{v}_{\mathbf{i}} := \boldsymbol{v}_{\mathbf{i}_0}$. A sample path vector is depicted in bold in Figure 7 which corresponds to $\mathbf{i} = (d_1, 1, 1, 1)$.*

Definition 9.11 (Kernel and activation path gains) *Consider a CNN model of the form (2.1) with input $\boldsymbol{x}$ and inputs of hidden units given by $\{\bar{\boldsymbol{h}}^{(\ell)}\}_{\ell=1}^{D}$. To any path vector $\boldsymbol{i}$ connecting an input feature to the output we associate two path gains: a kernel path gain denoted by $\boldsymbol{k}_{\mathbf{i}}$ and activation path gain denoted by $\phi'_{\mathbf{i}}(\boldsymbol{x})$ defined as*

$$\boldsymbol{k}_{\mathbf{i}} = \prod_{\ell=1}^{D} \boldsymbol{k}^{(\ell)}_{mod(\mathbf{i}_{\ell-1}, d_\ell)} \quad \text{and} \quad \phi'_{\mathbf{i}}(\boldsymbol{x}) = \prod_{\ell=1}^{D} \phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell}),$$

where $mod(a,b)$ denotes the remainder of dividing integer a by b. In words the kernel path gain is multiplication of the kernel weights along the path and the activation path gain is the multiplication of the derivatives of the activations evaluated at the hidden units along the path. For the path depicted in Figure 7 in bold the kernel path gain is equal $k_{\mathbf{i}} = \boldsymbol{k}^{(1)}_{d_1} \boldsymbol{k}^{(2)}_1 \boldsymbol{k}^{(3)}_1$ and the activation path gain is equal to $\phi'_{\mathbf{i}}(\boldsymbol{x}) = \phi'_1(\bar{h}^{(1)}_1)\phi'_2(\bar{h}^{(2)}_1)\phi'_3(\bar{h}^{(3)}_1)$.

Definition 9.12 (CNN offsprings) *Consider a CNN model of the form (2.1) with input $\boldsymbol{x}$ and inputs of hidden units given by $\{\bar{\boldsymbol{h}}^{(\ell)}\}_{\ell=1}^{D}$. We will associate a set $set_\ell(i) \subset \{1, \dots, p\}$ to the ith hidden unit of layer ℓ defined as $set_\ell(i) = \{(i-1)r_\ell + 1, (i-1)r_\ell + 2, \dots, ir_\ell\}$ where $r_\ell = p/p_\ell$. By construction, this corresponds to the set of entries of the input data $\boldsymbol{x}$ that $\bar{\boldsymbol{h}}^{(\ell)}_i(\boldsymbol{x})$ is dependent on. In the tree analogy $set_\ell(i)$ are the leaves of the tree connected to hidden unit i in the ℓth layer i.e. the set of offsprings of this hidden node. We depict $set_2(p_2)$ which are the offsprings of the last hidden node in layer two in Figure 7.*

We now will rewrite the population tensor in a form such that it is easier to see why it can be well approximated by a rank one tensor. Note that since the tensorization operation is linear the population tensor is equal to

$$\boldsymbol{T} = \mathbb{E}[f_{\text{CNN}}(\boldsymbol{x})\boldsymbol{X}] = \mathbb{E}[f_{\text{CNN}}(\boldsymbol{x})\mathcal{T}(\boldsymbol{x})] = \mathcal{T}\left(\mathbb{E}[f_{\text{CNN}}(\boldsymbol{x})\boldsymbol{x}]\right). \tag{9.12}$$

Define the vector $\boldsymbol{g}^{\text{CNN}}$ to be the population gradient vector i.e. $\boldsymbol{g}^{\text{CNN}} = \mathbb{E}[\nabla f_{\text{CNN}}(\boldsymbol{x})]$ and note that Stein's lemma combined with (9.12) implies that

$$\boldsymbol{T} = \mathcal{T}\left(\mathbb{E}[f_{\text{CNN}}(\boldsymbol{x})\boldsymbol{x}]\right) = \mathcal{T}\left(\mathbb{E}[\nabla f_{\text{CNN}}(\boldsymbol{x})]\right) = \mathcal{T}\left(\boldsymbol{g}^{\text{CNN}}\right). \tag{9.13}$$

Also note that

$$\frac{\partial f_{\text{CNN}}(\boldsymbol{x})}{\partial \boldsymbol{x}_i} = \boldsymbol{k}_{\mathbf{i}(i)} \phi'_{\mathbf{i}(i)}(\boldsymbol{x}).$$

Thus we have

$$\boldsymbol{g}^{\text{CNN}}_i = \mathbb{E}\left[\frac{\partial f_{\text{CNN}}(\boldsymbol{x})}{\partial \boldsymbol{x}_i}\right] = \boldsymbol{k}_{\mathbf{i}(i)}\, \mathbb{E}[\phi'_{\mathbf{i}(i)}(\boldsymbol{x})]. \tag{9.14}$$

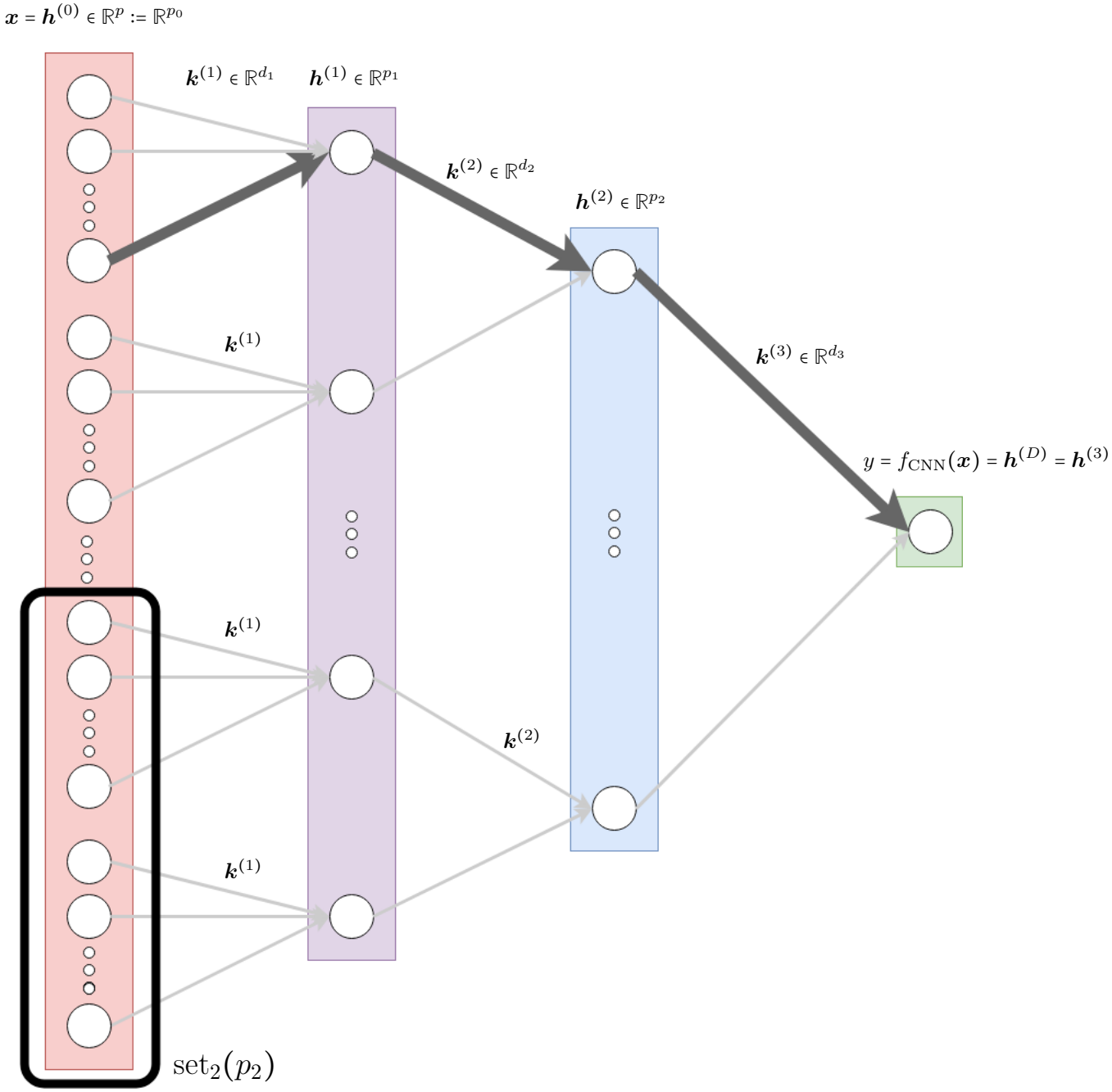


Figure 7: Tree visualization of a non-overlapping CNN model. The path in bold corresponds to the path vector $\mathbf{i} = (d_1, 1, 1, 1)$ from Definition 9.10. For this path the kernel path gain is equal $k_{\mathbf{i}} = \boldsymbol{k}_{d_1}^{(1)} \boldsymbol{k}_1^{(2)} \boldsymbol{k}_1^{(3)}$ and the activation path gain is equal to $\phi'_{\mathbf{i}}(\boldsymbol{x}) = \phi'_1(\bar{h}_1^{(1)}) \phi'_2(\bar{h}_1^{(2)}) \phi'_3(\bar{h}_1^{(3)})$. The set $\text{set}_2(p_2)$ (offsprings of the last hidden node in layer two) is outlined.

Define a vector $\boldsymbol{k} \in \mathbb{R}^p$ such that $\boldsymbol{k}_i = \boldsymbol{k}_{\mathbf{i}(i)}$. Since $\boldsymbol{k}_{\mathbf{i}(i)}$ consists of the product of the kernel values across the path $\mathbf{i}(i)$ it is easy to see that the tensor $\boldsymbol{K} := \mathcal{T}(\boldsymbol{k})$ is equal to

$$\boldsymbol{K} = \bigotimes_{\ell=1}^{D} \boldsymbol{k}^{(\ell)}. \tag{9.15}$$

Similarly define the vector $\boldsymbol{v} \in \mathbb{R}^p$ such that $\boldsymbol{v}_i = E[\phi'_{\mathbf{i}(i)}(\boldsymbol{x})]$ and define the corresponding tensor $\boldsymbol{V} = \mathcal{T}(\boldsymbol{v})$. Therefore, (9.14) can be rewritten in the vector form $\boldsymbol{g}^{\text{CNN}} = \boldsymbol{k} \odot \boldsymbol{v}$ where $\boldsymbol{a} \odot \boldsymbol{b}$ denotes entry-wise (Hadamard) product between two vectors/matrices/tensors $\boldsymbol{a}$ and $\boldsymbol{b}$ of the same size. Thus using (9.13) the population tensor can alternatively be written as

$$\boldsymbol{T} = \mathcal{T}\left(\boldsymbol{g}^{\text{CNN}}\right) = \boldsymbol{K} \odot \boldsymbol{V} = \left(\bigotimes_{\ell=1}^{D} \boldsymbol{k}^{(\ell)}\right) \odot \boldsymbol{V}.$$

Therefore, the population tensor $\boldsymbol{T}$ is the outer product of the convolutional kernels whose entries are masked with another tensor $\boldsymbol{V}$. If the entries of the tensor $\boldsymbol{V}$ were all the same the population tensor would be exactly rank one with the factors revealing the convolutional kernel. One natural choice for approximating the population tensor with a rank one matrix is to replace the masking tensor $\boldsymbol{V}$ with a scalar. That is, use the approximation $\boldsymbol{T} \approx c \bigotimes_{\ell=1}^{D} \boldsymbol{k}^{(\ell)}$. Recall that $\boldsymbol{L}_{\text{CNN}} := \alpha_{\text{CNN}} \bigotimes_{\ell=1}^{D} \boldsymbol{k}^{(\ell)}$ is exactly such an approximation with c set to α_{CNN} given by

$$\alpha_{\text{CNN}} = \prod_{\ell=1}^{D} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I})}[\phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell})].$$

To characterize the quality of this approximation note that

$$\begin{aligned}
\left|\!\left|\!\left| \boldsymbol{T} - \boldsymbol{L}_{\text{CNN}} \right|\!\right|\!\right| &\overset{(a)}{\leq} \left\| \boldsymbol{T} - \boldsymbol{L}_{\text{CNN}} \right\|_F, \\
&\overset{(b)}{=} \left\| \boldsymbol{K} \odot (\boldsymbol{V} - \alpha_{\text{CNN}}) \right\|_F, \\
&\overset{(c)}{=} \left\| \boldsymbol{k} \odot (\boldsymbol{v} - \alpha_{\text{CNN}}) \right\|_{\ell_2}, \\
&\overset{(d)}{\leq} \left\| \boldsymbol{k} \right\|_{\ell_2} \left\| \boldsymbol{v} - \alpha_{\text{CNN}} \right\|_{\ell_\infty}, \\
&\overset{(e)}{=} \prod_{\ell=1}^{D} \left\| \boldsymbol{k}^{(\ell)} \right\|_{\ell_2} \left\| \boldsymbol{v} - \alpha_{\text{CNN}} \right\|_{\ell_\infty}.
\end{aligned}$$

Here, (a) follows from the fact for a tensor, its spectral norm is smaller than its Frobenius norm, (b) from the definitions of $\boldsymbol{T}$ and $\boldsymbol{L}_{\text{CNN}}$, (c) from the fact that $\boldsymbol{K} = \mathcal{T}(\boldsymbol{k})$ and $\boldsymbol{V} = \mathcal{T}(\boldsymbol{v})$, (d) from the fact that $\|\boldsymbol{a} \odot \boldsymbol{v}\|_{\ell_2} \leq \|\boldsymbol{a}\|_{\ell_2} \|\boldsymbol{b}\|_{\ell_\infty}$, and (e) from the fact that the Euclidean norm of the kronecker product of of vectors is equal to the product of the Euclidean norm of the indivisual vectors. As a result of the latter inequality to prove Theorem 4.7 it suffices to show that

$$\left\| \boldsymbol{v} - \alpha_{\text{CNN}} \right\|_{\ell_\infty} \leq \sqrt{8\pi} \mu S \cdot \sup_{\ell} \prod_{i=1}^{\ell} \left\| \boldsymbol{k}^{(i)} \right\|_{\ell_2} \cdot \frac{D}{\sqrt{\min_{\ell} d_\ell}}. \tag{9.16}$$

In the next lemma we prove a stronger statement.

Lemma 9.13 *Assume the activations are S-smooth. Also consider a vector $\boldsymbol{v} \in \mathbb{R}^p$ with entries $\boldsymbol{v}_i = E_{\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I})}[\phi'_{\mathbf{i}(i)}(\boldsymbol{x})]$ and $\alpha_{CNN} = \prod_{\ell=1}^{D} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I})}[\phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell})]$ we have*

$$|\boldsymbol{v}_i - \alpha_{CNN}| \leq \kappa_i := \sqrt{8\pi} S \sum_{\ell=1}^{D} \left| \boldsymbol{k}^{(\ell)}_{mod(\mathbf{i}_{\ell-1}, d_\ell)} \right| \prod_{i=1}^{\ell-1} \|\boldsymbol{k}^{(i)}\|_{\ell_2}.$$

Here, $\boldsymbol{i}$ is the vector path that starts at input feature i.

Before proving this lemma let us explain how (9.16) follows from this lemma. To show this we use the kernel diffuseness assumption introduced in Definition 4.6. This definition implies that $\left|\boldsymbol{k}^{(\ell)}_{\mathrm{mod}(\mathbf{i}_{\ell-1},d_\ell)}\right| \le \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_\infty} \le \frac{\mu}{\sqrt{d_\ell}}\left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2}$. Thus we have

$$\begin{aligned}
\kappa_i :=& \sqrt{8\pi} S \sum_{\ell=1}^{D} |\boldsymbol{k}^{(\ell)}_{\mathrm{mod}(\mathbf{i}_{\ell-1},d_\ell)}| \prod_{i=1}^{\ell-1} \|\boldsymbol{k}^{(i)}\|_{\ell_2}, \\
\le& \sqrt{8\pi}\mu S \sum_{\ell=1}^{D} \frac{1}{\sqrt{d_\ell}} \left\|\boldsymbol{k}^{(\ell)}\right\|_{\ell_2} \prod_{i=1}^{\ell-1} \|\boldsymbol{k}^{(i)}\|_{\ell_2}, \\
=& \sqrt{8\pi}\mu S \sum_{\ell=1}^{D} \frac{1}{\sqrt{d_\ell}} \prod_{i=1}^{\ell} \|\boldsymbol{k}^{(i)}\|_{\ell_2}, \\
\le& \sqrt{8\pi}\mu D S \cdot \sup_\ell \frac{\prod_{i=1}^{\ell} \|\boldsymbol{k}^{(i)}\|_{\ell_2}}{\sqrt{d_\ell}}, \\
\le& \sqrt{8\pi}\mu S \cdot \sup_\ell \prod_{i=1}^{\ell} \left\|\boldsymbol{k}^{(i)}\right\|_{\ell_2} \cdot \frac{D}{\sqrt{\min_\ell d_\ell}}.
\end{aligned}$$

This completes the proof of Theorem 4.7. All that remains is to prove Lemma 9.13 which is the subject of the next section.

9.3.1 Proof of Lemma 9.13

To bound the difference between $\boldsymbol{v}_i$ and α_{CNN} consider the path $\mathbf{i} = \mathbf{i}(i)$ and define the variables $\{a_i\}_{i=0}^{D}$ as

$$a_i = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{I})}[\prod_{\ell=1}^{i} \phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell})] \prod_{\ell=i+1}^{D} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{I})}[\phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell})].$$

Note that $a_D = \boldsymbol{v}_i$ and $a_0 = \alpha_{\mathrm{CNN}}$. To bound the difference $\boldsymbol{v}_i - \alpha_{\mathrm{CNN}} = a_D - a_0$ we use a telescopic sum

$$|a_D - a_0| \le \sum_{\ell=0}^{D-1} |a_{\ell+1} - a_\ell|. \tag{9.17}$$

We thus focus on bounding each of the summands $|a_\ell - a_{\ell-1}|$. Setting $\gamma_\ell = \prod_{i=\ell}^{D} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{I})}[\phi'_i(\bar{\boldsymbol{h}}^{(i)}_{\mathbf{i}_i})]$, this can be written as

$$a_\ell - a_{\ell-1} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{I})}[(\phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell}) - \mathbb{E}[\phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell})])\prod_{i=1}^{\ell-1} \phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell})]\gamma_{\ell+1}.$$

Using $|\gamma_\ell| \le 1$ (which follows from the assumption that the activations are 1-Lipschitz), it suffices to bound

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{I})}[(\phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell}) - \mathbb{E}[\phi'_\ell(\bar{\boldsymbol{h}}^{(\ell)}_{\mathbf{i}_\ell})])\theta_{\ell-1}] \quad \text{where} \quad \theta_\ell = \prod_{i=1}^{\ell} \phi'_i(\bar{\boldsymbol{h}}^{(i)}_{\mathbf{i}_i}). \tag{9.18}$$

To this aim we state two useful lemmas whose proofs are deferred to Sections 9.3.1.1 and 9.3.1.2.

Lemma 9.14 *Let X, Y, Z be random variables where X is independent of Z. Let f be an L-Lipschitz function. Then*

$$|\mathbb{E}[f(X+Y)Z] - \mathbb{E}[f(X+Y)]\,\mathbb{E}[Z]| \le L\,\mathbb{E}[|\mathbf{zm}(Y)|(|Z| + |\mathbb{E}[Z]|)]. \tag{9.19}$$

Furthermore if $|Z| \le 1$, then

$$|\mathbb{E}[f(X+Y)Z] - \mathbb{E}[f(X+Y)]\,\mathbb{E}[Z]| \le 2L\,\mathbb{E}[|\mathbf{zm}(Y)|]. \tag{9.20}$$

Lemma 9.15 *$\bar{\boldsymbol{h}}_i^{(\ell)}(\boldsymbol{x})$ (and $\boldsymbol{h}_i^{(\ell)}(\boldsymbol{x})$) is a deterministic function of the entries of $\boldsymbol{x}$ indexed by $\mathrm{set}_\ell(i)$. In other words, there exists a function f such that $\bar{\boldsymbol{h}}_i^{(\ell)}(\boldsymbol{x}) = f(\boldsymbol{x}_{set_\ell(i)})$.*

With these lemmas in-hand we return to bounding (9.18). To this aim we decompose $\bar{\boldsymbol{h}}_{\mathbf{i}_\ell}^{(\ell)}$ as follows

$$\bar{\boldsymbol{h}}_{\mathbf{i}_\ell}^{(\ell)} = \sum_{i=1}^{d_\ell} \boldsymbol{k}_i^{(\ell)} \boldsymbol{h}_{d_\ell(\mathbf{i}_\ell-1)+i}^{(\ell-1)} = \boldsymbol{k}_{\mathrm{mod}(\mathbf{i}_{\ell-1},d_\ell)}^{(\ell)} \boldsymbol{h}_{\mathbf{i}_{\ell-1}}^{(\ell-1)} + \mathbf{r},$$

where the $\mathbf{r}$ term is the contribution of the entries of $\boldsymbol{h}^{(\ell-1)}$ other than $\mathbf{i}_{\ell-1}$. By the non-overlapping assumption, $\mathbf{r}$ is independent of $\theta_{\ell-1}$ as well as $\boldsymbol{h}_{\mathbf{i}_{\ell-1}}^{(\ell-1)}$ (see Lemma 9.15). In particular, $\boldsymbol{h}_{\mathbf{i}_{\ell-1}}^{(\ell-1)}$ and $\theta_{\ell-1}$ is a function of $\boldsymbol{x}_{\mathrm{set}_{\ell-1}(\mathbf{i}_{\ell-1})}$ whereas $\mathbf{r}$ is a function of the entries over the complement $\mathrm{set}_\ell(\mathbf{i}_\ell) - \mathrm{set}_{\ell-1}(\mathbf{i}_{\ell-1})$. With these observations, applying Lemma 9.14 with $f = \phi_\ell', X = \mathbf{r}, Y = \boldsymbol{k}_{\mathrm{mod}(\mathbf{i}_{\ell-1},d_\ell)}^{(\ell)} \boldsymbol{h}_{\mathbf{i}_{\ell-1}}^{(\ell-1)}, Z = \theta_{\ell-1}$ and using the fact that $|\theta_{\ell-1}| \le 1$ which holds due to 1-Lipschitzness of σ_ℓ's, we conclude that

$$|\,\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{I})}[(\phi_\ell'(\bar{\boldsymbol{h}}_{\mathbf{i}_\ell}^{(\ell)}) - \mathbb{E}[\phi_\ell'(\bar{\boldsymbol{h}}_{\mathbf{i}_\ell}^{(\ell)})])\theta_{\ell-1}]| \le 2S\,\mathbb{E}\,|\mathbf{zm}(Y)|.$$

Here, S is the smoothness of σ_ℓ and Lipschitz constant of ϕ_ℓ'. To conclude, we need to assess the $\mathbb{E}\,|\mathbf{zm}(Y)|$ term. Now note that starting from $\boldsymbol{x}$, each entry of $\boldsymbol{h}^{(\ell-1)}$ is obtained by applying a sequence of inner products with $\{\boldsymbol{k}^{(i)}\}_{i=1}^{\ell-1}$ and activations $\sigma_\ell(\cdot)$, which implies $\boldsymbol{h}_{\mathbf{i}_{\ell-1}}^{(\ell-1)}$ is a $\prod_{i=1}^{\ell-1}\|\boldsymbol{k}^{(i)}\|_{\ell_2}$-Lipschitz function of $\mathrm{set}_{\ell-1}(\mathbf{i}_{\ell-1})$. This implies Y is a Lipschitz function of a Gaussian vector with Lipschitz constant $L_Y = |\boldsymbol{k}_{\mathrm{mod}(\mathbf{i}_{\ell-1},d_\ell)}^{(\ell)}|\prod_{i=1}^{\ell-1}\|\boldsymbol{k}^{(i)}\|_{\ell_2}$. Hence, $\mathbf{zm}(Y)$ obeys the tail bound

$$\mathbb{P}(|\mathbf{zm}(Y)| \ge t) \le 2\exp(-\frac{t^2}{2L_Y^2}).$$

Using a standard integration by parts argument the latter implies that

$$\mathbb{E}\,|\mathbf{zm}(Y)| \le \sqrt{2\pi}L_Y.$$

Thus,

$$|a_\ell - a_{\ell-1}| \le |\boldsymbol{k}_{\mathrm{mod}(\mathbf{i}_{\ell-1},d_\ell)}^{(\ell)}|\prod_{i=1}^{\ell-1}\|\boldsymbol{k}^{(i)}\|_{\ell_2}.$$

concluding the upper-bound on each summand of (9.17). Combining such upper bounds (9.17) implies

$$|a_D - a_0| \le \sqrt{8\pi}S\sum_{\ell=1}^{D}|\boldsymbol{k}_{\mathrm{mod}(\mathbf{i}_{\ell-1},d_\ell)}^{(\ell)}|\prod_{i=1}^{\ell-1}\|\boldsymbol{k}^{(i)}\|_{\ell_2} := \kappa_{\mathbf{i}}.$$

This concludes the proof of Lemma 9.13.

9.3.1.1 Proof of Lemma 9.14

Using the independence of X, Z, we can write

$$\begin{aligned}\mathbb{E}[f(X+Y)Z] &= \mathbb{E}[f(X+\mathbb{E}[Y])Z] + \mathbb{E}[(f(X+Y) - f(X+\mathbb{E}[Y]))Z] \\ &= \mathbb{E}[f(X+\mathbb{E}[Y])]\,\mathbb{E}[Z] + \mathbb{E}[(f(X+Y) - f(X+\mathbb{E}[Y]))Z] \\ &= \mathbb{E}[f(X+Y)]\,\mathbb{E}[Z] + \mathbb{E}[f(X+\mathbb{E}[Y]) - f(X+Y)]\,\mathbb{E}[Z] + \mathbb{E}[(f(X+Y) - f(X+\mathbb{E}[Y]))Z].\end{aligned}$$

This implies

$$\mathbb{E}[f(X+Y)Z] - \mathbb{E}[f(X+Y)]\,\mathbb{E}[Z] = \mathbb{E}[f(X+\mathbb{E}[Y]) - f(X+Y)]\,\mathbb{E}[Z] + \mathbb{E}[(f(X+Y) - f(X+\mathbb{E}[Y]))Z].$$

Now, using Lipschitzness of f, we deterministically have that $|f(X+\mathbb{E}[Y])-f(X+Y)| \le L|Y-\mathbb{E}[Y]| = L|\mathbf{zm}(Y)|$. Similarly, $|(f(X+Y)-f(X+\mathbb{E}[Y]))Z| \le L|\mathbf{zm}(Y)Z|$. Taking absolute values we arrive at

$$|\mathbb{E}[f(X+\mathbb{E}[Y])-f(X+Y)]\,\mathbb{E}[Z]+\mathbb{E}[(f(X+Y)-f(X+\mathbb{E}[Y]))Z]| \le L\,\mathbb{E}[|\mathbf{zm}(Y)|(|Z|+|\mathbb{E}[Z]|)].$$

This immediately implies (9.19). If $|Z| \le 1$ almost surely, we have $\mathbb{E}[|\mathbf{zm}(Y)Z|] \le \mathbb{E}[|\mathbf{zm}(Y)|]$ and $|\mathbb{E}[Z]| \le 1$ which yields the $2L\,\mathbb{E}[|\mathbf{zm}(Y)|]$ upper bound.

9.3.1.2 Proof of Lemma 9.15

Informally, this lemma is obvious via the tree visualization. To formally prove this lemma we use an induction argument. For $\bar{\boldsymbol{h}}^{(1)}$ the result is trivial because $\bar{\boldsymbol{h}}_i^{(1)} = \left\langle \boldsymbol{k}^{(1)}, \boldsymbol{x}(i) \right\rangle$ which is a weighted sum of entries corresponding to $\mathrm{set}_1(i)$. Suppose the claim holds for all layers less than or equal to $\ell-1$ and $\bar{\boldsymbol{h}}_i^{(\ell-1)} = f_{\ell-1}(\boldsymbol{x}_{\mathrm{set}_{\ell-1}(i)})$. For layer ℓ, we can use the fact that $\mathrm{set}_\ell(i) = \bigcup_{j=1}^{d_\ell} \mathrm{set}_{\ell-1}((i-1)d_\ell + j)$ to conclude that

$$\begin{aligned}\bar{\boldsymbol{h}}_i^{(\ell)} &= \sum_{j=1}^{d_\ell} \boldsymbol{k}_j^{(\ell)} \sigma_{\ell-1}(\bar{\boldsymbol{h}}_{(i-1)d_\ell+j}^{(\ell-1)}) \\ &= \sum_{j=1}^{d_\ell} \boldsymbol{k}_j^{(\ell)} \sigma_{\ell-1}(f_{\ell-1}(\boldsymbol{x}_{\mathrm{set}_{\ell-1}((i-1)d_\ell+j)})) := f_\ell(\boldsymbol{x}_{\mathrm{set}_\ell(i)}).\end{aligned}$$

The latter is clearly a deterministic function of $\boldsymbol{x}_{\mathrm{set}_\ell(i)}$. Also it is independent of entry i because it simply chunks the vector $\boldsymbol{x}_{\mathrm{set}_\ell(i)}$ into d_ℓ sub-vectors and returns a sum of weighted functions of these sub-vectors. Here, the weights are the entries of $\boldsymbol{k}^{(\ell)}$ and the functions are given by $\sigma_{\ell-1}(f_{\ell-1}(\cdot))$ (also note that the activation output is simply $\boldsymbol{h}_i^{(\ell)} = \sigma_\ell(f_\ell(\boldsymbol{x}_{\mathrm{set}_\ell(i)}))$).

9.4 Proofs for learning the convolutional kernels (Proof of Theorem 4.8)

The first part of the theorem follows trivially by combining Theorems 4.2 and 4.7. To translate a bound on the tensor spectral norm of $\boldsymbol{T}_n - \boldsymbol{L}_{\mathrm{CNN}}$ to a bound on learning the kernels, requires a perturbation argument for tensor decompositions. This is the subject of the next lemma.

Lemma 9.16 *Let $\boldsymbol{L} = \gamma \bigotimes_{\ell=1}^{D} \boldsymbol{v}_\ell$ be a rank one tensor with $\{\boldsymbol{v}_i\}_{i=1}^D$ vectors of unit norm. Also assume $\boldsymbol{E}$ is a perturbation tensor obeying $\|\boldsymbol{E}\| \le \delta$. Set*

$$\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D = \underset{\bar{\boldsymbol{u}}_1, \bar{\boldsymbol{u}}_2, \ldots, \bar{\boldsymbol{u}}_D}{\arg\max} \left\langle \boldsymbol{L}, \bigotimes_{\ell=1}^{D} \bar{\boldsymbol{u}}_\ell \right\rangle \quad \textit{subject to} \quad \|\bar{\boldsymbol{u}}_1\|_{\ell_2} = \|\bar{\boldsymbol{u}}_2\|_{\ell_2} = \ldots = \|\bar{\boldsymbol{u}}_D\|_{\ell_2} = 1. \tag{9.21}$$

Then we have

$$\prod_{i=1}^{D} |\boldsymbol{u}_i^* \boldsymbol{v}_i| \ge 1 - 2\delta/\gamma.$$

The proof of Theorem 4.8 is complete by applying Lemma 9.16 above with $\boldsymbol{v}_\ell = \boldsymbol{k}^{(\ell)}$, $\boldsymbol{u}_\ell = \hat{\boldsymbol{k}}^{(\ell)}$, $\gamma = \alpha_{\mathrm{CNN}}$ and $\boldsymbol{E} = \boldsymbol{T}_n - \boldsymbol{L}_{\mathrm{CNN}}$. All that remains is to prove Lemma 9.16 which is the subject of the next section.

9.4.1 Proof of Lemma 9.16

To prove this lemma first note that for any two rank one tensors we have

$$\left\langle \bigotimes_{\ell=1}^{D} \boldsymbol{u}_\ell, \bigotimes_{\ell=1}^{D} \boldsymbol{v}_\ell \right\rangle = \prod_{i=1}^{D} \langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle .$$

Using this equality together with the fact that the vectors $\{\boldsymbol{u}_\ell\}_{\ell=1}^D$ are a maximizer for (9.21) we conclude that

$$\begin{aligned}\left\langle \boldsymbol{L}+\boldsymbol{E}, \bigotimes_{\ell=1}^D \boldsymbol{v}_\ell \right\rangle &\le \left\langle \boldsymbol{L}+\boldsymbol{E}, \bigotimes_{\ell=1}^D \boldsymbol{u}_\ell \right\rangle, \\ &\le \left| \left\langle \boldsymbol{L}, \bigotimes_{\ell=1}^D \boldsymbol{u}_\ell \right\rangle \right| + \left| \left\langle \boldsymbol{E}, \bigotimes_{\ell=1}^D \boldsymbol{u}_\ell \right\rangle \right|, \\ &\le \gamma \prod_{i=1}^D |\langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle| + \delta. \end{aligned} \tag{9.22}$$

Furthermore, note that

$$\left\langle \boldsymbol{L}+\boldsymbol{E}, \bigotimes_{\ell=1}^D \boldsymbol{v}_\ell \right\rangle = \gamma + \left\langle \boldsymbol{E}, \bigotimes_{\ell=1}^D \boldsymbol{v}_\ell \right\rangle \ge \gamma - \delta. \tag{9.23}$$

Combining (9.22) and (9.23) we conclude that

$$\gamma \prod_{i=1}^D |\langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle| + \delta \ge \gamma - \delta,$$

concluding the proof.

9.5 Generalization bounds for CERM

In this section we prove a generalization bound for Centered Empirical Risk Minimization (CERM) (5.1). The following theorem shows that using a finite sample size n, CERM is guaranteed to choose a function close to population's minimizer. For the sake of this section $\|f\|_{L_\infty}$ will be the Lipschitz constant of a function.

Theorem 9.17 *Let $\mathcal{F}$ be a class of functions $f : \mathbb{R}^p \to \mathbb{R}$. Suppose $\sup_{f\in\mathcal{F}} \|f\|_{L_\infty} \le R$. Let $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \sim (\boldsymbol{x}, y)$ be i.i.d. data points where $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_p)$ and y is so that $y - \mathbb{E}[y]$ is K subgaussian. Suppose $\mathcal{F}$ has $\|\cdot\|_{L_\infty}$, δ-covering bound obeying $\log N_\delta \le s \log \frac{C}{\delta}$ for some constants $s \ge 1, C \ge R$. Given $\varepsilon \le \bar{K} = K + R$, suppose $n \ge \mathcal{O}(\max\{\varepsilon^{-1}, \varepsilon^{-2}\}\bar{K}^2 s \log \frac{C' p \bar{K}}{\varepsilon})$ for some $C' > 0$. Then the CERM output (5.1) obeys*

$$\mathbb{E}[\big(\hat{f}(\boldsymbol{x}) - y - \mathbb{E}[(\hat{f}(\boldsymbol{x}) - y)]\big)^2] \le \min_{f\in\mathcal{F}} \mathbb{E}[(f(\boldsymbol{x}) - y - \mathbb{E}[(f(\boldsymbol{x}) - y)])^2] + \varepsilon. \tag{9.24}$$

with probability $1 - \exp(-\mathcal{O}(n)) - 4n\exp(-p)$.

Proof Consider the centered empirical loss that can alternatively be written in the form

$$\begin{aligned} E(f) &= \frac{1}{n}\sum_{i=1}^n \mathbf{zm}(y_i - f(\boldsymbol{x}_i))^2 - \frac{1}{n^2}(\sum_{i=1}^n \mathbf{zm}(y_i - f(\boldsymbol{x}_i)))^2 - \mathbb{E}[\mathbf{zm}(f(\boldsymbol{x}) - y)^2]. \\ &= \frac{1}{n}\sum_{i=1}^n \mathbf{zm}(\mathbf{zm}(y_i - f(\boldsymbol{x}_i))^2) - \frac{1}{n^2}(\sum_{i=1}^n \mathbf{zm}(y_i - f(\boldsymbol{x}_i)))^2 \\ &:= T_1 + T_2 \end{aligned} \tag{9.25}$$

To prove the theorem, we will simply bound the supremum $\sup_{f\in\mathcal{F}} |E(f)| \le \sup_{f\in\mathcal{F}} |T_1 + T_2|$. Pick a δ covering $\mathcal{F}_\delta$ of $\mathcal{F}$ with size $s \log \frac{C}{\delta}$ where δ will be determined later in this proof. We first bound $E(f)$ for all $f \in \mathcal{F}_\delta$. Given a fixed f, observe that $\|\mathbf{zm}(\mathbf{zm}(y_i - f(\boldsymbol{x}_i))^2)\|_{\psi_1} \le \mathcal{O}(\bar{K})^2 = \mathcal{O}(K + R)^2$ which follows from $\|\mathbf{zm}(y_i - f(\boldsymbol{x}_i))\|_{\psi_2} \le \mathcal{O}(\bar{K}) = \mathcal{O}(K + R)$. Applying subexponential concentration, since T_1 is sum of i.i.d. subexponentials, we have

$$\mathbb{P}(|T_1| \ge \varepsilon) \le \exp(-\mathcal{O}(n \min\{\varepsilon^2/\bar{K}^2, \varepsilon/\bar{K}\})). \tag{9.26}$$

Next, since $\|\mathbf{zm}(y_i - f(\boldsymbol{x}_i))\|_{\psi_2} \leq \mathcal{O}(\bar{K})$, we can conclude that $\|\frac{1}{n}\sum_{i=1}^n \mathbf{zm}(y_i - f(\boldsymbol{x}_i))\|_{\psi_2} \leq \mathcal{O}(\bar{K})/\sqrt{n} \implies \|T_2\|_{\psi_1} \leq \mathcal{O}(\bar{K})^2/n$. Using (9.26) for T_1 and the subexponential tail bound for T_2 holds when $\varepsilon \leq \bar{K}$, and assuming the number of samples n obeys $n \geq \mathcal{O}(\max\{\varepsilon^{-1}, \varepsilon^{-2}\}\bar{K}^2 s \log \frac{C}{\delta})$, then for all cover elements

$$|T_1| + |T_2| \leq 2\varepsilon.$$

holds with probability at least $1 - \exp(-\mathcal{O}(n))$. To conclude the proof, we need to move from the cover $\mathcal{F}_\delta$ to $\mathcal{F}$. Pick $f \in \mathcal{F}$ and its δ neighbor $f_\delta \in \mathcal{F}_\delta$. Utilizing the deterministic relation $|\mathbf{zm}(X)| \leq |X| + |\mathbb{E}[X]|$ and using the fact that f_δ is in a δ neighborhood of f, we arrive at the following bounds

$$|\mathbf{zm}(f(\boldsymbol{x}) - f_\delta(\boldsymbol{x}))| \leq \delta(\|\boldsymbol{x}\|_{\ell_2} + \mathbb{E}[\|\boldsymbol{x}\|_{\ell_2}]). \tag{9.27}$$

$$|\mathbf{zm}(f(\boldsymbol{x}) + f_\delta(\boldsymbol{x}) - 2y)| \leq 2R(\|\boldsymbol{x}\|_{\ell_2} + \mathbb{E}[\|\boldsymbol{x}\|_{\ell_2}]) + 2K|\mathbf{zm}(y)|. \tag{9.28}$$

Next observe that, with probability at least $1 - 4n\exp(-p)$, all $\boldsymbol{x}_i, y_i$ obey $\|\boldsymbol{x}_i\|_{\ell_2} \leq \mathcal{O}(\sqrt{p}), |\mathbf{zm}(y_i)| \leq \mathcal{O}(K\sqrt{p})$. Combining this with (9.27), we conclude that for all $1 \leq i \leq n$

$$|\mathbf{zm}(f(\boldsymbol{x}_i) - y_i)^2 - \mathbf{zm}(f_\delta(\boldsymbol{x}_i) - y_i)^2| \leq \mathcal{O}(\bar{K}\delta p). \tag{9.29}$$

Expanding the square differences in the same way, an identical argument shows the following two deviation bounds,

$$|\mathbb{E}[\mathbf{zm}(f(\boldsymbol{x}_i) - y_i)^2 - \mathbf{zm}(f_\delta(\boldsymbol{x}_i) - y_i)^2]| \leq \mathcal{O}(\bar{K}\delta p), \tag{9.30}$$

$$\frac{1}{n^2}|(\sum_{i=1}^n \mathbf{zm}(y_i - f(\boldsymbol{x}_i)))^2 - (\sum_{i=1}^n \mathbf{zm}(y_i - f_\delta(\boldsymbol{x}_i)))^2| \leq \mathcal{O}(\bar{K}\delta p).$$

Combining these three inequalities ((9.29) and (9.30)) and substituting them in (9.25), we conclude that for all neighbors f_δ, f,

$$|E(f) - E(f_\delta)| \leq \mathcal{O}(\bar{K}\delta p).$$

Next we set $\delta = c\varepsilon/(p\bar{K})$ for a sufficiently small constant $c > 0$, to find that with probability at least $1 - \exp(-n)$, $\sup_{f\in\mathcal{F}} |E(f)| \leq \varepsilon$ holds as long as the number of samples n obeys $n \geq \mathcal{O}(\max\{\varepsilon^{-1}, \varepsilon^{-2}\}\bar{K}^2 s \log \frac{Cp\bar{K}}{c\varepsilon})$. We define $\mathcal{L}_{erm}(f) = \frac{1}{n}\sum_{i=1}^n (y_i - f(\boldsymbol{x}_i) - r_{\text{avg}}(f))^2$ and $\mathcal{L}_{pop}(f) = \mathbb{E}[\mathbf{zm}(f(\boldsymbol{x}) - y)^2]$. We also denote the CERM minimizer $f_{erm} = \arg\min_{f\in\mathcal{F}} \mathcal{L}(f)$ and population minimizer $f_{pop} = \min_{f\in\mathcal{F}} \mathcal{L}_{pop}(f)$. Inequality (9.24) follows from the facts that we simultaneously have $|E(f_{erm})| \leq \mathcal{O}(\varepsilon)$ and $|E(f_{pop})| \leq \mathcal{O}(\varepsilon)$ which implies that

$$\mathcal{L}_{pop}(f_{erm}) \leq \mathcal{L}_{erm}(f_{erm}) + \mathcal{O}(\varepsilon) \leq \mathcal{L}_{erm}(f_{pop}) + \mathcal{O}(\varepsilon) \leq \mathcal{L}_{pop}(f_{pop}) + \mathcal{O}(\varepsilon),$$

concluding the proof. ■

9.6 Proof of Theorem 5.1

In this section we will show how Theorem 5.1 follows from Theorem 9.17. To this aim we need to show that $\mathcal{F}_{\hat{\boldsymbol{k}},B}$ has a small Lipscshitz covering number. We construct the following cover $\mathcal{F}'$ for the set $\mathcal{F}_{\hat{\boldsymbol{k}},B}$. Let $B' = B^{1/D}$. Pick a $\delta \leq B'$ ℓ_2 cover $\mathcal{C}$ of the interval $[-B', B']$ which has size $2B'/\delta$. Let $\mathcal{C}_i$ be identical copies of $\mathcal{C}$. We set

$$\mathcal{F}' = \{f_{\text{CNN}}(\beta_\ell \hat{\boldsymbol{k}}^{(\ell)}) \;\big|\; \beta_\ell \in \mathcal{C}_\ell,\ 1 \leq \ell \leq D\}.$$

In words, we construct CNNs by picking numbers from the cartesian product $\mathcal{C}_1 \times \ldots \mathcal{C}_D$ and scaling $\{\hat{\boldsymbol{k}}^{(\ell)}\}_{\ell=1}^D$ with them. We now argue that $\mathcal{F}'$ provides a cover of $\mathcal{F}$. Given $f \in \mathcal{F}$ with scalings β_ℓ, there exists $f' \in \mathcal{F}'$ which uses scalings β'_ℓ such that $|\beta_\ell - \beta'_\ell| \leq \delta$. Now, let f_ℓ be the function with scalings β'_i until $i = \ell$ and β_i for $i > \ell$. Note that $f_0 = f$, $f_D = f'$. With this, we write

$$\|f - f'\|_{L_\infty} \leq \sum_{i=1}^D \|f_{i+1} - f_i\|_{L_\infty}.$$

Observe that f_{i-1} and f_i have equal layers except the ith layer. Let g_1 be the function of the first $i-1$ layers and g_2 be the function of layers $i+1$ to D. We have that $f_{i+1}(\boldsymbol{x}) - f_i(\boldsymbol{x}) = g_2(\phi(\boldsymbol{K}_i(g_1(\boldsymbol{x})))) - g_2(\phi(\boldsymbol{K}_i'(g_1(\boldsymbol{x}))))$ where $\boldsymbol{K}_i, \boldsymbol{K}_i'$ differ in the ith layer kernels of f and f' created from $\beta_i\hat{\boldsymbol{k}}^{(i)}$ and $\beta_i'\hat{\boldsymbol{k}}^{(i)}$ respectively. Also, observe that g_1 is B'^{i-1} Lipschitz and $g_2(\phi(\cdot))$ is B'^{D-i} Lipschitz functions. Hence,

$$|g_2(\phi(\boldsymbol{K}_i(g_1(\boldsymbol{x})))) - g_2(\phi(\boldsymbol{K}_i'(g_1(\boldsymbol{x}))))| \le B'^{D-i}|\boldsymbol{K}_i(g_1(\boldsymbol{x})) - \boldsymbol{K}_i'(g_1(\boldsymbol{x}))| \le B'^{D-i}\delta B'^{i-1} \le \delta B'^{D-1}. \quad (9.31)$$

Summing over all i, this implies that $\|f - f'\|_{L_\infty} \le D\delta B'^{D-1}$. Recalling $|\mathcal{F}'| \le (2B'/\delta)^D$ and setting $\delta = \varepsilon/(DB'^{D-1})$, the ε covering number of $\mathcal{F}_{\hat{\boldsymbol{k}},B}$ is $N_\varepsilon \le (2DB'^D/\varepsilon)^D = (2DB/\varepsilon)^D$ which implies $\log N_\varepsilon = D\log(\frac{2DB}{\varepsilon})$. Now, since all kernels have Euclidean norm bounded by B', we have $\|f_{\text{CNN}}()\|_{L_\infty} \le B$ and $\|f\|_{L_\infty} \le B$ for all $f \in \mathcal{F}$. This also implies $\|\mathbf{zm}(f_{\text{CNN}}(\boldsymbol{x}))\|_{\psi_2} = \mathcal{O}(B)$. Hence, we can apply Theorem 9.17 to conclude the proof of Theorem 5.1.

Acknowledgements

M.S. is supported by the Air Force Office of Scientific Research Young Investigator Program (AFOSR-YIP) under award number FA9550-18-1-0078, a Google Faculty Research Award and a grant by the Northrop Grumman Cybersecurity Research Consortium.

References

[1] A. Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with Gaussian inputs. *arXiv preprint arXiv:1702.07966*, 2017.

[2] 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, 2014.

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

[4] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592, 2014.

[5] Rasmus Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.

[6] Emmanuel J. Candes, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *arXiv preprint arXiv:1407.1065*, 2014.

[7] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728, 2016.

[8] Nadav Cohen and Amnon Shashua. Convolutional rectifier networks as generalized tensor decompositions. In *International Conference on Machine Learning*, pages 955–963, 2016.

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[10] Sjoerd Dirksen. Tail bounds via generic chaining. *arXiv preprint arXiv:1309.3522*, 2013.

[11] Simon S Du, Jason D Lee, and Yuandong Tian. When is a convolutional filter easy to learn? *arXiv preprint arXiv:1709.06129*, 2017.

[12] Simon S Du, Jason D Lee, Yuandong Tian, Barnabas Poczos, and Aarti Singh. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *arXiv preprint arXiv:1712.00779*, 2017.

[13] Haoyu Fu, Yuejie Chi, and Yingbin Liang. Local geometry of one-hidden-layer neural networks for logistic regression. *arXiv preprint arXiv:1802.06463*, 2018.

[14] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points – online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

[15] Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*, 2017.

[16] 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.

[17] Kishore Jaganathan, Samet Oymak, and Babak Hassibi. Recovery of sparse 1-d signals from the magnitudes of their fourier transform. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium On*, pages 1473–1477. IEEE, 2012.

[18] Kishore Jaganathan et al. Predicting splicing from primary sequence with deep learning. *preprint*, 2018.

[19] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015.

[20] Jean Kossaifi, Zachary C Lipton, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar. Tensor regression networks. *arXiv preprint arXiv:1707.08308*, 2017.

[21] Jean Kossaifi, Yannis Panagakis, and Maja Pantic. Tensorly: Tensor learning in python. *arXiv preprint arXiv:1610.09555*, 2016.

[22] 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.

[23] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, pages 597–607, 2017.

[24] Eran Malach and Shai Shalev-Shwartz. A provably correct algorithm for deep learning that actually works. *arXiv preprint arXiv:1803.09522*, 2018.

[25] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks. *arXiv preprint arXiv:1804.06561*, 2018.

[26] Marco Mondelli and Andrea Montanari. On the connection between learning two-layers neural networks and tensor decomposition. *arXiv preprint arXiv:1802.07301*, 2018.

[27] Samet Oymak. Learning compact neural networks with regularization. *arXiv preprint arXiv:1802.01223*, 2018.

[28] Samet Oymak and Mahdi Soltanolkotabi. Learning the input layer of a deep convolutional neural network via centered gradient descent. *preprint*, 2018.

[29] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

[30] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[31] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[32] Mahdi Soltanolkotabi. Learning ReLUs via gradient descent. *arXiv preprint arXiv:1705.04591*, 2017.

[33] Mahdi Soltanolkotabi. Structured signal recovery from quadratic measurements: Breaking sample complexity barriers via nonconvex optimization. *arXiv preprint arXiv:1702.06175*, 2017.

[34] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *arXiv preprint arXiv:1707.04926*, 2017.

[35] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

[36] Michel Talagrand. *The generic chaining: upper and lower bounds of stochastic processes*. Springer Science & Business Media, 2006.

[37] Michel Talagrand. Gaussian processes and the generic chaining. In *Upper and Lower Bounds for Stochastic Processes*, pages 13–73. Springer, 2014.

[38] Ryota Tomioka and Taiji Suzuki. Spectral norm of random tensors. *arXiv preprint arXiv:1407.1870*, 2014.

[39] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[40] Kai Zhong, Zhao Song, and Inderjit S Dhillon. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017.

[41] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. *arXiv preprint arXiv:1706.03175*, 2017.