Towards Understanding the Generalization Bias of Two Layer Convolutional Linear Classifiers with Gradient Descent

Yifan Wu
Carnegie Mellon University
yw4@cs.cmu.edu

Barnabás Póczos
Carnegie Mellon University
bapoczos@cs.cmu.edu

Aarti Singh
Carnegie Mellon University
aarti@cs.cmu.edu

Abstract

A major challenge in understanding the generalization of deep learning is to explain why (stochastic) gradient descent can exploit the network architecture to find solutions that have good generalization performance when using high capacity models. We find simple but realistic examples showing that this phenomenon exists even when learning linear classifiers — between two linear networks with the same capacity, the one with a convolutional layer can generalize better than the other when the data distribution has some underlying spatial structure. We argue that this difference results from a combination of the convolution architecture, data distribution and gradient descent, all of which are necessary to be included in a meaningful analysis. We analyze of the generalization performance as a function of data distribution and convolutional filter size, given gradient descent as the optimization algorithm, then interpret the results using concrete examples. Experimental results show that our analysis is able to explain what happens in our introduced examples.

1 Introduction

It has been shown that the capacities of successful deep neural networks are typically large enough such that they can fit random labelling of the inputs in a dataset (Zhang et al., 2016). Hence an important problem is to understand why gradient descent (and its variants) is able to find the solutions that generalize well on unseen data. Another key factor, besides gradient descent, in achieving good generalization performance in deep neural networks is architecture design with weight sharing (e.g. Convolutional Neural Networks (CNNs) (LeCun et al., 1998) and Long Short Term Memories (LSTMs) (Hochreiter and Schmidhuber, 1997)). To the best of our knowledge, none of the existing work on analyzing the generalization bias of gradient descent takes these specific architectures into formal analysis. One may conjecture that the advantage of weight sharing is caused by reducing the network capacity compared with using fully connected layers without talking about gradient descent. However, as we will show later, there is a joint effect between network architectures and gradient descent on the generalization performance even if the model capacity remains unchanged. In this work we try to analyze the generalization bias of two layer CNNs together with gradient descent, as one of the initial steps towards understanding the generalization performance of deep learning in practice.

CNNs have proven to be successful in learning tasks where the data distribution has some underlying spatial structure such as image classification (Krizhevsky et al., 2012; He et al., 2016), Atari games (Mnih et al., 2013) and Go (Silver et al., 2017). A common view of how CNNs work is that convolutional filters extract high level features while pooling exploits spatial translation invariance (Goodfellow et al., 2016). Pooling, however, is not always used, especially in reinforcement learning (RL) tasks (see the networks used in Atari games (Mnih et al., 2013), and Go (Silver et al., 2017)) even if exploiting some level of spatial invariance is desired for good generalization. For example, if we are training a robot arm to pick up an apple from a table, one thing we are expecting is that the robot learns to move its arm to the left if the apple is on its left and vice versa. If we use a policy network to decide "left" or "right", we expect the network to be able to generalize without being trained with all of the pixel level combinations of the (arm, apple) location pair. In order to see whether stacking up convolutional filters

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

and fully connected layers without pooling can still exploit the spatial invariance in the data distribution, we design the following tasks, which are the simplified 1-D version of 2-D image based classification and control tasks:

- **Binary classification (Task-Cls):** Suppose we are trying to classify object A v.s. B given a 1-D d-pixel "image" as input. We assume that only one of the two objects appears on each image and the object occupies exactly one pixel. In pixel level inputs $x \in \{-1, 0, +1\}^d$, we use $+1$ to represent object A, -1 for object B and 0 for nothing. We use label $y = +1$ for object A and $y = -1$ for object B. The resulting dataset looks as follows:

$$x = [0, ..., 0, -1, 0, ..., 0] \rightarrow y = -1\,;$$
$$x = [0, ..., 0, +1, 0, ..., 0] \rightarrow y = +1\,.$$

  The entire possible dataset contains $2d$ samples.

- **First-person vision-based control (Task-1stCtrl):** Suppose we are doing first-person view control in 3-D environments with visual images as input, e.g. robot navigation, and the task is to go to the proximity of object A. One decision the robot has to make is to turn left if the object is on the left half of the image and turn right if it is on the right half. We consider the simplified 1-D version, where each input $x \in \{0, 1\}^d$ contains only one non-zero element $x_i = 1$ with $y = -1$ if the object is on the left half ($i \leq d/2$) and $y = +1$ if the object is on the right half ($i > d/2$). The resulting dataset looks as follows:

$$x = [0, ...1.., 0, ..., 0] \rightarrow y = -1\,;$$
$$x = [0, ..., 0, ...1.., 0] \rightarrow y = +1\,.$$

  The entire possible dataset contains d samples.

- **Third-person vision-based control (Task-3rdCtrl):** We consider the fixed third-person view control, e.g. controlling a robot arm, and the task is to control the agent (arm), denoted by object B and represented by -1, to touch the target object A, represented by $+1$, in the scene. Again we want to move the arm B to the left ($y = -1$) if A is on the left of B and move it to the right ($y = +1$) if A is on the right. The resulting dataset looks as follows:

$$x = [0, ... + 1.... - 1....., 0] \rightarrow y = -1\,;$$
$$x = [0, ... - 1... + 1..., 0] \rightarrow y = +1\,.$$

  The entire possible dataset contains $d(d-1)$ samples.

Although all of the three tasks we described above have a finite number of samples in the whole dataset we still seek good generalization performance on these tasks when learning from only a subset of them. We do not want the learner to see almost all possible pixel-wise appearance of the objects before it is able to perform well. Otherwise the sample complexity will be huge when the resolution of the image becomes higher and the number of objects involved in the task grows larger.

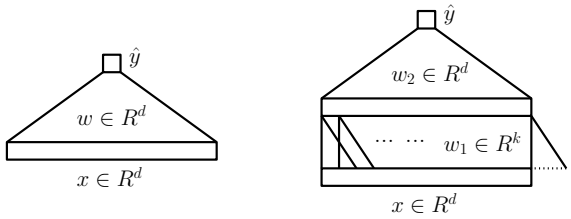
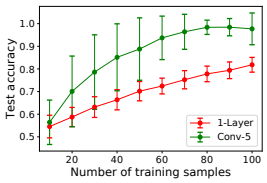


Figure 1: Left: $\hat{y} = \text{sign}\left(w^T x\right)$ — **Model-1-Layer**. Right: $\hat{y} = \text{sign}\left(w_2^T \text{Conv}(w_1, x)\right)$ — **Model-Conv-k**.

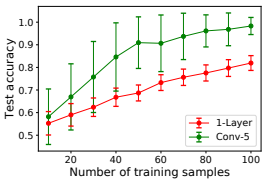
One key property of all these three tasks we designed is that the data distribution is linearly separable even without introducing the bias term. That is, for each of the tasks, there exist at least one $w \in \mathbb{R}^d$ such that $y = \text{sign}\left(w^T x\right)$ for all (x, y) in the whole dataset. This property gives superior convenience in both experiment control and theoretical analysis, while several important aspects , as we will explain later in this section, in analyzing the generalization of deep networks are still preserved: The linear separator w for a training set is not unique and the interesting question is why different algorithms (architecture plus optimization routine) can find solutions that generalize better or worse on unseen samples.

Since the data distribution is linearly separable the immediate learning algorithm one would try on these tasks is to train a single layer linear classifier using logistic regression, SVM, etc. However, this seems to be not exploiting the spatial structure of the data distribution and we are interested in seeing whether adding convolutional layers could help. More specifically, we consider adding a convolution layer between the input x and the output layer, where the convolution layer contains only one size-k filter (output_channel = 1) with stride = 1 and without non-linear activation functions. Figure 1 shows the two models we are comparing. We also experimented with the fully-connected two layer linear classifier where the first layer has weight matrix $W_1 \in \mathbb{R}^{d \times d}$ without non-linear activation. This model has the same generalization behavior as a single-layer linear classifier in all of our experiments so we will not show these results separately. This phenomenon, however, may also exhibit an interesting problem to study.

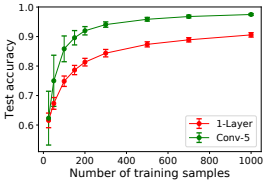
It is worth noting that Model-1-Layer and Model-



(a) Task-Cls



(b) Task-1stCtrl



(c) Task-3rdCtrl

Figure 2: Comparing the generalization performance between single layer and two layer convolutional linear classifiers with different sizes of training data. Training samples are uniformly sampled from the whole dataset ($d = 100$) with replacement. The models are trained by minimizing the hinge loss $(1 - yf(x))_+$ using full-batch gradient descent. Training stops when training reaches 0. Trained models are then evaluated on the whole dataset (including the training samples). For convolution layer we use a single size-5 filter with stride 1 and padding with 0. Each plotted point is based on repeating the same configuration for 100 times.

Conv-k represent exactly the same set of functions. That is, for any w in Model-1-Layer we are able to find (w_1, w_2) in Model-Conv-k such that they represent the same function, and vice versa. Therefore, both of the two models have the same capacity and any difference in the generalization performance cannot be explained by the difference of capacity. We compare the generalization performance of the two models in Figure 1 on all of the three tasks we have introduced. As shown in Figure 2, Model-Conv-k outperforms Model-1-Layer on all of the three tasks. The rest of our paper is motivated by explaining the generalization behavior of Model-Conv-k.

Explaining our empirical observations requires a generalization analysis that depends on data distribution, convolution structure and gradient descent. Any of these three factors cannot be isolated from the analysis for the following reasons:

- **Data distribution:** If we randomly flip the label for each data point independently then all models will have the same generalize performance on unseen samples.
- **Convolution structure:** The network structure is the main factor that we are trying to analyze. We further argue that we should explain the advantage of convolution and not just depth. This is because, as we mentioned earlier, adding a fully connected layer does not provide any advantage compared with a single-layer model in all of our experiments.
- **Gradient descent:** The analysis should also include the optimization algorithm since the function classes represented by the two models we are comparing are equivalent. For example, the Model-Conv-k can be optimized in the way that we first find a solution w by optimizing Model-1-Layer then let $w_1 = [1, 0, ..., 0]$ and $w_2 = w$, which also gives a solution for Model-Conv-k but has no generalization advantage. Therefore, analyzing how gradient descent is able to exploit the convolution structure is necessary to explain the generalization advantage in our experiments.

In this paper we provide a data dependent analysis on the generalization performance of two layer convolutional linear classifiers given gradient descent as the optimizer. In Section 3 we first give a general analysis then interpret our results using specific examples. In Section 4 we empirically verify that our analysis is able to explain the observations in our experiments in Figure 2. Due to space constraints, proofs are relegated to the appendix.

Our main contribution can be highlighted as follows: (i) We design simple but realistic examples that are theory-friendly while preserving important challenges in understanding what is happening in practice. (ii) We are the first to provide a formal generalization analysis that considers the interaction among data distribution, convolution, and gradient descent, which is necessary to provide meaningful understanding for deep networks. (iii) We derive a closed form weight dynamics under gradient descent with a modified hinge loss and relate the generalization performance to the first singular vector pair of some matrix computed from the training samples. (iv) We interpret the results with one of our concrete examples using Perron-Frobenius Theorem for non-negative matrices, which shows how much sample complexity we can save by adding a convolution layer. (v) Our result reveals an interesting difference between the generalization bias of ConvNet and that of traditional regularizations — The bias itself requires some training samples to be built up. (vi) Our experiments show that our analysis is able to explain what happens in our examples. More specifically, we show that the performance under our modified hinge loss is strongly correlated with the performance under the real hinge loss.

2 Preliminaries

2.1 Learning Binary Classifiers

We consider learning binary classifiers $\hat{y} = \text{sign}(f_w(x))$ with a function class f parameterized by w, in order to predict the real label $y \in \{-1, +1\}$ given an input $x \in \mathbb{R}^d$. A random label is predicted with equal chance if $f_w(x) = 0$. In a single layer linear classifier (Model-1-Layer) we have $w \in \mathbb{R}^d$ and $f_w(x) = w^T x$. In the two layer convolutional linear classifiers (Model-Conv-k) we have $w = (w_1, w_2)$ where the convolution filter $w_1 \in \mathbb{R}^k$ and the output layer $w_2 \in \mathbb{R}^d$ ($k \le d$). f_w can be written as $f_w(x) = \sum_{i=1}^{d} w_{2,i} \sum_{j=1}^{k} w_{1,j} x_{i+j-1}$ where every term whose index is out of range is treated as zero.

We denote the entire data distribution as $\mathcal{D}$, which one can sample data points (x, y)s from. We say drawing a training set $D \sim \mathcal{D}$ when we independently sample $n = |D|$ data points from $\mathcal{D}$ with replacement and take the collection as D. For finite datasets, e.g. in the tasks we introduced, we assume the data distribution is uniform over all data points. In this paper we only consider the case where there is no noise in the label, i.e. the true label y is always deterministic given an input x, so that we can write $(x, y) \in D$ or $x \in D$ interchangeably.

Given a training set $D_{\text{tr}} \sim \mathcal{D}$ with n_{tr} samples and a model f_w, we learn the classifier by minimizing the empirical hinge loss $\mathcal{L}(w; D_{\text{tr}}) = \frac{1}{n_{\text{tr}}} \sum_{(x,y)\in D_{\text{tr}}} (1 - y f_w(x))_+$ using full-batch gradient descent with learning rate $\alpha > 0$:

$$\begin{aligned} w^{t+1} &= w^t - \alpha \nabla_w \mathcal{L}(w^t; D_{\text{tr}}) \\ &= w^t + \frac{\alpha}{n_{\text{tr}}} \sum_{(x,y)\in D_{\text{tr}}} \mathbb{I}\{y f_{w^t}(x) < 1\}\, y \nabla_w f_{w^t}(x)\,. \end{aligned}$$

Given a classifier f_w and data distribution $\mathcal{D}$, the generalization error can be written as

$$\begin{aligned} &\mathcal{E}(w; \mathcal{D}) = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\hat{y}}\left[\mathbb{I}\{\hat{y} \neq y\}\right]\right] \\ &= \mathbb{E}_{\mathcal{D}}\left[\mathbb{I}\{y f_w(x) < 0\} + \frac{1}{2}\mathbb{I}\{y f_w(x) = 0\}\right] \\ &= \mathbb{E}_{\mathcal{D}}\left[\bar{\mathcal{E}}(y f_w(x))\right], \end{aligned} \tag{1}$$

where we define function $\bar{\mathcal{E}} : \mathbb{R} \mapsto \{0, \frac{1}{2}, 1\}$ as $\bar{\mathcal{E}}(x) = \mathbb{I}\{x < 0\} + \frac{1}{2}\mathbb{I}\{x = 0\}$ which is non-increasing and satisfies $\bar{\mathcal{E}}(\alpha x) = \bar{\mathcal{E}}(x)$ for any $\alpha > 0$.

2.2 An Alternative Form for Two Layer ConvNets

For the convenience of analysis, we use an alternative form to express Model-Conv-k. Let $A_x \in \mathbb{R}^{d\times k}$ be $[x, x_{\leftarrow_1}, ..., x_{\leftarrow_{k-1}}]$, where k is the size of the filter and $x_{\leftarrow_l}$ is defined as the input vector left-shifted by l positions: $x_{\leftarrow_l, i} = x_{i+l}$ (pad with 0 if out of range). Then $f_w(x)$ can be written as $f_w(x) = w_1^T A_x^T w_2$. The definition of A_x is visualized in Figure 3.

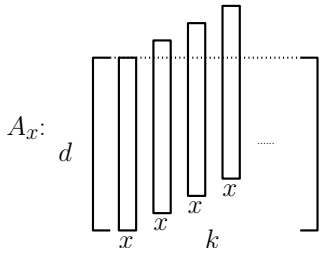


Figure 3: Matrix $A_x \in \mathbb{R}^{d\times k}$ given $x \in \mathbb{R}^d$.

Further define $M_{x,y} = y A_x$ then we have $y f_w(x) = w_1^T M_{x,y}^T w_2$. We can write the empirical loss as $\mathcal{L}(w; D_{\text{tr}}) = \frac{1}{n_{\text{tr}}} \sum_{(x,y)\in D_{\text{tr}}} (1 - w_1^T M_{x,y}^T w_2)_+$ and the generalization error as $\mathcal{E}(w; \mathcal{D}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\bar{\mathcal{E}}\left(w_1^T M_{x,y}^T w_2\right)\right]$.

3 Theoretical Analysis

In this section we analyze the generalization behavior of Model-Conv-k when training with gradient descent. We introduce a modified version of the hinge loss which enables a closed-form expression for the weight dynamics . Based on the closed-form we show that the weights converge to some specific directions as $t \to \infty$. Plugging the asymptotic weights back to the generalization error gives the observation that the generalization performance depends on the first singular vector pair of the average $M_{x,y}$ over the training samples. We interpret our result under Task-Cls, which shows that our analysis is well aligned with the empirical observations and quantifies how much ($\approx 2k-1$ times) sample complexity can be saved by adding a convolution layer.

3.1 The Extreme Hinge Loss (X-Hinge)

We consider minimizing a linear variant of the hinge loss $\ell(w; x, y) = -y f_w(x)$. We call it *the extreme hinge loss* because the gradient of this loss is the same as the gradient of the hinge loss $\ell(w; x, y) = (1 - y f_w(x))_+$ when $y f_w(x) < 1$. Then the training loss becomes $\mathcal{L}(w; D_{\text{tr}}) = -w_1^T M_{\text{tr}}^T w_2$ where we define $M_{\text{tr}} = \frac{1}{n_{\text{tr}}} \sum_{(x,y)\in D} M_{x,y} \in \mathbb{R}^{d\times k}$. Note that minimizing this loss will lead to $\mathcal{L} \to -\infty$. However, since the normal hinge loss can be viewed as a fit-then-stop version of X-hinge, considering loss $\mathcal{L} \to -\infty$ in our cases gives interesting insights about the generalization bias of Conv-k under the normal hinge loss. In the next section we will further verify the correlation between X-hinge and the normal hinge loss through

experiments, which can be summarized as follows:

(i) Under X-hinge (w_1, w_2) converges to a limit direction which brings superior generalization advantage. (ii) Conv-k generalizes better under normal hinge because the weights tend to converge to this limit direction (but stopped when training loss reaches 0). (iii) The variance (due to different initialization) in the generalization performance of Conv-k comes from how close the weights are to this limit direction when training stops.

3.2 An Asymptotic Analysis

The full-batch gradient descent update for minimizing X-hinge with learning rate α is $w_1^{t+1} = w_1^t + \alpha M_{\text{tr}}^T w_2^t$ and $w_2^{t+1} = w_2^t + \alpha M_{\text{tr}} w_1^t$. For the simplicity of writing our analysis we let $w_2^0 = 0$, which does not affect our theoretical conclusion. Now we try to analyze the generalization error when $t \to \infty$. (See Appendix for a finite-time closed form expression of w^t.) First we will show that the weight converge to a specific direction as $t \to \infty$ given fixed w_1^0:

Lemma 1. *For any training set D_{tr} let $M_{\text{tr}} = U\Sigma V^T$ be (any of) its SVD and $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_k \geq 0$ be the diagonal of Σ with $\sigma_1 > 0$.[1] Denote $1 \leq m \leq k$ be the largest number such that $\sigma_1 = \sigma_m$, then we have*

$$w_1^\infty \doteq \lim_{t\to+\infty} \frac{2w_1^t}{(1+\alpha\sigma_1)^t} = V_{:m}V_{:m}^T w_1^0\,,$$
$$w_2^\infty \doteq \lim_{t\to+\infty} \frac{2w_2^t}{(1+\alpha\sigma_1)^t} = U_{:m}V_{:m}^T w_1^0\,, \qquad (2)$$

where $A_{:m}$ denotes the first m columns of a matrix A.

Let $w^\infty = (w_1^\infty, w_2^\infty)$ be a random variable that depends on (D_{tr}, w_1^0) and $\mathcal{F}^\infty(x, y, w_1^0, D_{\text{tr}}) \doteq y f_{w^\infty}(x) = {w_1^\infty}^T M_{x,y}^T w_2^\infty = {w_1^0}^T V_{:m}V_{:m}^T M_{x,y}^T U_{:m}V_{:m}^T w_1^0$. We define the *asymptotic generalization error* for Model-Conv-k with gradient descent on data distribution $\mathcal{D}$ as [2]

$$\mathcal{E}_{\text{Convk}}^\infty(\mathcal{D}) \doteq \mathbb{E}_{D_{\text{tr}}, w_1^0}\left[\mathcal{E}(w^\infty, \mathcal{D})\right] = \mathbb{E}_{w_1^0, D_{\text{tr}}, (x,y)}\left[\bar{\mathcal{E}}\left(\mathcal{F}^\infty(x, y, w_1^0, D_{\text{tr}})\right)\right]. \qquad (3)$$

One can further remove the dependence on w_1^0 when using Gaussian initialization:

Theorem 2. *Consider training Model-Conv-k by gradient descent with initialization $w_1^0 \sim \mathcal{N}(0, b^2 I_k)$ for some $b > 0$ and $w_2^0 = 0$. Let UV_1^M denote the set of left-right singular vector pairs corresponding to the largest singular value σ_1 for a given matrix M. The asymptotic generalization error in (3) can be upper bounded by $\mathcal{E}_{\text{Convk}}^\infty(\mathcal{D}) \leq \mathbb{E}_{D_{\text{tr}},(x,y)}\left[\bar{\mathcal{E}}\left(\min_{(u,v)\in UV_1^{M_{\text{tr}}}} v^T M_{x,y}^T u\right)\right]$.*

When the first singular vector pair of M_{tr} is unique (which is always true when n_{tr} is not too small in our experiments), denoted by (u, v), we have $m = 1$ and Lemma 1 says that w_1^t converges to the same direction as v while w_2^t converges to the same direction as u. In this case Theorem 2 holds with equality and we can remove the min operator . The asymptotic generalization performance is characterized by how many data points in the whole dataset can be correctly classified by Model-Conv-k with the first singular vector pair of M_{tr} as its weights. Later on we will show that this quantity is highly correlated with the real generalization performance in practice where we use the original hinge loss but not the extreme one.

3.3 Interpreting the Result with Task-Cls

We will use our previously introduced task Task-Cls to show that the quantity in Theorem 2 is non-vacuous: it saves approximately $2k-1$ times samples over Model-1-Layer in Task-Cls.

3.3.1 Decomposing the generalization error

Notation. For any $l \in [d] = \{1, ..., d\}$ define $e_l \in \{0,1\}^d$ to be the vector that has 1 in its l-th position and 0 elsewhere. Then the set of inputs x in Task-Cls is the set of e_l and $-e_l$ for all l. Note that in Task-Cls $y_{(-x)} = -y_{(x)}$ and $M_{-x,-y} = M_{x,y}$ so each pair of data points e_l and $-e_l$ can be treated equivalently during training and test. Thus we can think of sampling from $\mathcal{D}$ as sampling from the d positions. Let $\mathcal{U}[d]$ denote the uniform distribution over $[d]$. Given a training set D_{tr} define $S_{\text{tr}} = \{l \in [d] : e_l \in D_{\text{tr}} \vee -e_l \in D_{\text{tr}}\}$ to be the set of non-zero positions that appear in D_{tr}.

To analyze the quantity in Theorem 2 we notice that all elements in M_{tr} are non-negative for any D_{tr}. By applying the Perron-Frobenius theorem (Frobenius, 1912) which characterizes the spectral property for non-negative matrices we can further decompose it into two parts. We first introduce the following definition[3]:

Definition 3. Let $A \in \mathbb{R}^{k\times k}$ be a non-negative square matrix. A is *primitive* if there exists a positive integer t such that $A_{ij}^t > 0$ for all i, j.

[1] We implicitly assume that the data distribution $\mathcal{D}$ satisfies $\Pr(M_{\text{tr}} = 0) = 0$ for any $n_{\text{tr}} > 0$, which is true in all of our examples.

[2] Note that $\mathcal{E}(w^\infty, \mathcal{D}) = \lim_{t\to\infty} \mathcal{E}(w^t, \mathcal{D})$ may not hold due to the discontinuity of $\mathbb{I}\{\cdot\}$.

[3] See appendix for what a primitive matrix looks like and what it indicates.

Now we are ready to state the following theorem:

Theorem 4. *Let $\Omega(A)$ be the event that A is primitive and $\Omega^c(A)$ be its complement. Consider training Model-Conv-k with gradient descent on Task-Cls. The asymptotic generalization error defined in (3) can be upper bounded by* $\mathcal{E}^{\infty}_{\text{Convk}} \leq \Pr\left(\Omega^c(M_{\text{tr}}^T M_{\text{tr}})\right) + \frac{1}{2}\mathbb{E}_{l\sim\mathcal{U}[d]}\left[\Pr\left(\forall l' \in S_{\text{tr}}, |l'-l| \geq k\right)\right]$.

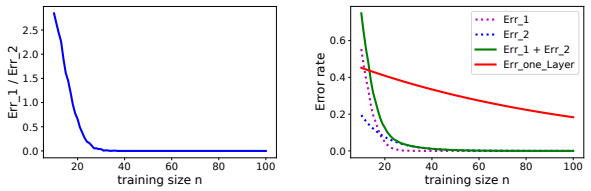
The message delivered by Theorem 4 is that the upper bound of the asymptotic generalization error depends on whether $M_{\text{tr}}^T M_{\text{tr}}$ is primitive and (if yes) how much of the whole dataset is covered by the k-neighborhoods of the points in the training set. Next we will discuss the two quantities in Theorem 4 separately.

First consider the second term. Let $X_{\text{tr}} = \{x_1, x_2, ..., x_n\}$ be the collection of xs in the training set D_{tr} with $n_{\text{tr}} = n$ and $L_{\text{tr}} = \{l_1, l_2, ..., l_n\}$ be the corresponding non-zero positions of X_{tr}, which are i.i.d. samples from $\mathcal{U}[d]$. Therefore $\Pr\left(\forall l' \in S_{\text{tr}}, |l'-l| \geq k\right) = \Pr\left(\bigcap_{i=1}^n |l_i - l| \geq k\right) = \left(\frac{d-k-\min\{k,l,d-l+1\}+1}{d}\right)^n$. The second quantity now can be exactly calculated by averaging over all $l \in [d]$. To get a cleaner form that is independent of l, we can either further upper bound it by $\left(\frac{d-k}{d}\right)^n$ or approximate it by $\left(\frac{d-2k+1}{d}\right)^n$ if $k \ll d$.

Now come back to the first quantity , which is the probability that $M_{\text{tr}}^T M_{\text{tr}}$ is not primitive. Exactly calculating or even tightly upper bounding this quantity seems hard so we derive a sufficient condition for $M_{\text{tr}}^T M_{\text{tr}}$ to be primitive so that the probability of its complement can be used to upper bound the probability that $M_{\text{tr}}^T M_{\text{tr}}$ is not primitive:

Lemma 5. *Let $\tilde{\Omega}_{\text{tr}}$ be the event that there exists $k \leq i \leq d$ such that both $i-1, i \in S_{\text{tr}}$. If $\tilde{\Omega}_{\text{tr}}$ happens then $M_{\text{tr}}^T M_{\text{tr}}$ is primitive.*

Lemma 5 says that $M_{\text{tr}}^T M_{\text{tr}}$ is primitive if there exist two training samples with adjacent non-zero positions and the positions should be after k due to shifting/padding issues. Thus we have $\Pr\left(\Omega^c(M_{\text{tr}}^T M_{\text{tr}})\right) \leq \Pr\left(\tilde{\Omega}^c_{\text{tr}}\right)$. Calculating the quantity $\Pr\left(\tilde{\Omega}^c_{\text{tr}}\right)$ which is the probability that no adjacent non-zero positions after k appear in a randomly sampled training set with size n, however, is still hard so we empirically estimate $\Pr\left(\tilde{\Omega}^c_{\text{tr}}\right)$. Figure 4(a) shows that $\Pr\left(\tilde{\Omega}^c_{\text{tr}}\right)$ has a lower order than the quantity $\left(\frac{d-2k+1}{d}\right)^n$ as n goes larger so the second term in Theorem 4 becomes dominating in the generalization bound. We give a rough intuition about this: let s_n be the expected number of unique samples when we uniformly draw n samples from $1, ..., d$, then $s_n \to d$ as $n \to \infty$. The available slots for the $n+1$-th sample not creating adjacent pairs is at most $d - s_n$. So the total probability of not having adjacent pairs can be roughly upper bounded by $\prod_{i=1}^n \frac{d-s_n}{d}$. Taking the ratio to $\left(\frac{d-2k+1}{d}\right)^n$ gives $\prod_{i=1}^n \frac{d-s_n}{d-2k+1}$, which goes to zero as $n \to \infty$ and $s_n \to d$.



(a) Ratio Err_1/Err_2 (b) Conv-5 v.s. 1-Layer

Figure 4: Visualizing the calculated/estimated generalization errors. Err_1 denotes the estimate of $\Pr\left(\tilde{\Omega}^c_{\text{tr}}\right)$, Err_2 denotes $\frac{1}{2}\left(\frac{d-2k+1}{d}\right)^n$ and Err_one_Layer denotes $\frac{1}{2}\left(\frac{d-1}{d}\right)^n$, where we set $d = 100$, $k = 5$, and each estimate for $\Pr\left(\tilde{\Omega}^c_{\text{tr}}\right)$ is based on repeatedly sampling n points from $\mathcal{U}[d]$ for 10,000 times.

3.3.2 Comparing with Model-1-Layer

We compare the second term of Theorem 4, which is approximately $\frac{1}{2}\left(\frac{d-2k+1}{d}\right)^n$, with the generalization error of Model-1-Layer. Assume all elements in the single layer weights are initialized independently with some distribution centering around 0. In each step of gradient descent w_i is updated only if $x = \pm e_i$ is in the training set. So for any (x, y) in the whole dataset, it is guaranteed to be correctly classified only if $\pm e_i$ appears in the training set, otherwise it has only a half chance to be correctly classified due to random initialization. Then the generalization error can be written as $\mathcal{E}_{\text{1Layer}} = \frac{1}{2}\mathbb{E}_{l\sim\mathcal{U}[d]}\left[\Pr\left(\forall l' \in S_{\text{tr}}, l' \neq l\right)\right] = \frac{1}{2}\left(\frac{d-1}{d}\right)^n$ The two error rates are the same when $k = 1$, which is expected, and $\frac{1}{2}\left(\frac{d-2k+1}{d}\right)^n$ is smaller when $k > 1$.

To see how much we save on the sample complexity by using Model-Conv-k to achieve a certain error rate ϵ we let $\frac{1}{2}\left(\frac{d-2k+1}{d}\right)^n = \epsilon$, which gives $n = \frac{1}{\log d - \log(d-2k+1)} \log\frac{1}{2\epsilon}$ and $\lim_{d\to\infty} \frac{n}{d} = \frac{1}{2k-1}\log\frac{1}{2\epsilon}$. So the sample complexity for using Model-Conv-k is approximately $\frac{d}{2k-1}\log\frac{1}{2\epsilon}$ when $k \ll d$ while we need $d\log\frac{1}{2\epsilon}$ samples for Model-1-Layer. Model-Conv-k requires approximately $2k-1$ times fewer samples when $k \ll d$ and ϵ is small enough such that the first part in Theorem 4 is negligible.

Now take the first term in Theorem 4 into consideration by adding up the empirically estimated $\Pr\left(\tilde{\Omega}^c_{\text{tr}}\right)$ and $\frac{1}{2}\left(\frac{d-2k+1}{d}\right)^n$ as an upper bound for $\mathcal{E}^{\infty}_{\text{Convk}}$ then compare the sum with $\mathcal{E}_{\text{1Layer}}$. Figure 4(b) shows that the estimated upper bound for $\mathcal{E}^{\infty}_{\text{Conv5}}$ is clearly smaller than $\mathcal{E}_{\text{1Layer}}$ when n is not too small. This difference is well aligned with our empirical observation in Fig-

ure 2(a) where the two models perform similarly when n is small and Model-Conv-k outperforms Model-1-Layer when n grows larger.

Theorem 4 and Lemma 5 show that when there exist $l, l' \in S_{\text{tr}}$ such that $|l - l'| = 1$ then the training samples in D_{tr} generalize to their k-neighbors. We argue that this generalization bias itself requires some samples to be built up, which means that achieving k-neighbors generalization requires some condition hold for S_{tr}. Having $l, l' \in S_{\text{tr}}$ such that $|l - l'| = 1$ is a sufficient condition but not a requirement. Now we derive a necessary condition for this generalization advantage:

Proposition 6. *If for all $l \in S_{\text{tr}}$ we have $l \geq k$ and for any $l, l' \in S_{\text{tr}}$ we have $|l - l'| \geq 2k$ then this k-neighbor generalization does not hold for Conv-k in Task-Cls. Actually, under this condition and $w_1^0 \sim \mathcal{N}(0, b^2 I)$, there is no generalization advantage for Model-Conv-k compared to Model-1-Layer.*

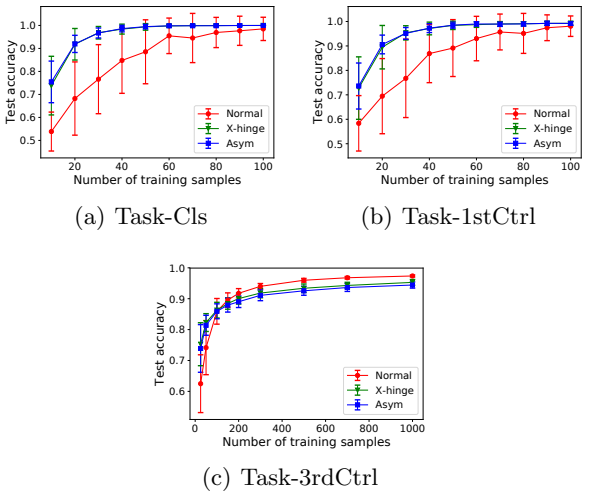
Proposition 6 states that when the training samples are too sparse Model-Conv-k provides the same generalization performance as Model-1-Layer. Together with Theorem 4 and Lemma 5 our results reveal a very interesting fact that, unlike traditional regularization techniques, the generalization bias here requires a certain amount of training samples before saving the sample complexity effectively.

4 Experiments

In this section we empirically investigate the relationship between our analysis and the actual performance in experiments (Figure 2). Recall that we made two major surrogates during our analysis: (i) We consider the extreme hinge loss $\ell(w; x, y) = -y f_w(x)$ instead of the typically used $\ell(w; x, y) = (1 - y f_w(x))_+$. (ii) We consider the asymptotic weights w^∞ instead of w^t. Now we study the difference caused by these surrogates. We compare the following three quantities: (a) the empirical estimate for the asymptotic error $\mathcal{E}^\infty_{\text{Convk}}$ using Theorem 2 by computing SVD of sampled M_{tr}s, (b) the test errors by training with the extreme hinge loss and (c) the real hinge loss. [4] The results are shown in Figure 5.

It can be seen from Figure 5 that there is not much difference between the estimated quantity in Theorem 2 by SVD and the actual test error by training with the extreme hinge loss $\ell(w; x, y) = -y f_w(x)$, which verifies our derivation in Section 3. It is also shown that, especially in Task-Cls and Task-1stCtrl, the asymptotic quantity can be viewed as an upper confidence bound

[4] We also tried cross entropy loss with sigmoid and found no much difference from using the hinge loss.



(a) Task-Cls (b) Task-1stCtrl

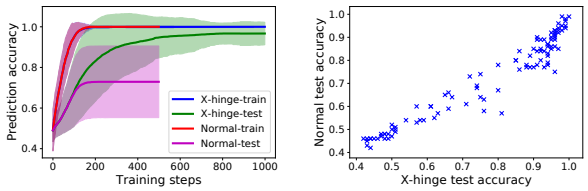
(c) Task-3rdCtrl

Figure 5: Comparing estimated asymptotic error (Asym) v.s. finite time extreme hinge loss (X-hinge) v.s. normal hinge loss (Normal) with different sizes of training data. For using normal hinge loss training stops when training loss goes to 0 while for extreme hinge loss we train the model for 1000 steps. The other settings remain the same as the experiments shown in Figure 2.

for the actual performance with the normal hinge loss. The asymptotic quantity has a much lower variance which only comes from the randomization of the training set so the high variance with the normal hinge loss is caused by random initialization and good initializations would perform closer to the asymptotic quantity than the bad ones. To verify this, we fix the training set and compare the performance of the two losses at each training step t with difference initial weights w^0. Figure 6(a) [5] shows the convergence of training/test accuracies with difference losses. With the normal hinge loss, the test performance remains the same once the training loss reaches 0. With the extreme hinge loss, the test performance is still changing even after the training data is fitted and eventually converges to $\mathcal{E}^\infty_{\text{Convk}}$. As we can see, there is a difference in how fast the direction of weight w^t converges (in terms of test accuracy) to its limit w^∞ defined in Lemma 1 with different initialization when using the extreme hinge loss. We further argue that this variance is strongly correlated with the variance in the generalization performance under the normal hinge loss, as shown in Figure 6(b), from which we can see how well a model trained using the normal hinge loss with some w^0 generalizes depends on how fast w^t converges to its limit direction using the extreme hinge loss.

One may wonder that whether X-hinge always generalizes better than the normal hinge under gradient

[5] Results for the other two tasks are put in the appendix.



(a) Convergence with t. (b) Correlation at $t = 150$.

Figure 6: The effect of weight initialization in Task-Cls. We fix $d = 100$, $n = 30$ and train Model-Conv-5 with 100 different random initializations using both losses. w^0 is uniformly sampled from $[-b, b]^{d+k}$.

descent as in Task-Cls. However, this is not true in Task-3rdCtrl, where the limit direction is better when n_{tr} is small but worse when n_{tr} is large, according to Figure 5(c). The reason is that the limit direction w^∞ may not be able to separate the training set[6]. This indicates that the potential generalization "benefit" from the convolution layer may actually be a bias.

5 Related Work

Among all recent attempts that try to explain the behavior of deep networks our work is distinct in the sense that we study the generalization performance that involves the interaction between gradient descent and convolution. For example, Du et al. (2017) study how gradient descent learns convolutional filters but they focus on optimization instead of generalization. Several recent works study the generalization bias of gradient descent (Hardt et al., 2015; Dziugaite and Roy, 2017; Brutzkus et al., 2017; Soudry et al., 2017) but they are not able to explain the advantage of convolution in our examples. Hardt et al. (2015) bounds the stability of stochastic gradient descent within limited number of training steps. Dziugaite and Roy (2017) proposes a non-vacuous bound that relies on the stochasticity of the learning process. Neither limited number of training steps or stochasticity is necessary to achieve better generalization in our examples. Similarly to our work, Soudry et al. (2017) study the convergence of $w/\left\|w\right\|_2$ under gradient descent. However, their work is limited to single layer logistic regression and their result shows that the linear separator converges to the max-margin one, which does not indicate good generalization in our cases. Gunasekar et al. (2018) also study the limit directions of multi-layer linear convolutional classifiers under gradient descent. Their result is not directly applicable to ours since they consider loopy convolutional filters with full width $k = d$ while we consider filters with $k \ll d$ and padding with 0. Our setting of filters is closer to what people use in practice. Moreover, Gunasekar et al. (2018) does not provide any generalization analysis while we show that the limit direction of the convolutional linear classifier provides significant generalization advantage on some specific tasks. Brutzkus et al. (2017) shows that optimizing an overparametrized 2-layer network with SGD can generalize on linearly separable data. Their work is limited to only training the first fully connected layer while we study jointly training two layers with convolution. Another thread of work (Bartlett et al., 2017; Neyshabur et al., 2017b,a) tries to develop novel complexity measures that are able to characterize the generalization performance in practice. These complexities are based on the margin, norm or the sharpness of the learned model on the training samples. Taking Task-Cls as an example, the linear classifier with the maximum margin or minimum norm will place 0 on the weights where there are no training samples, which is undesirable in our case, while the sharpness of the learned model in terms of training loss contains no information about how it behaves on unseen samples. So none of these measures can be applied to our scenario. (Fukumizu, 1999; Saxe et al., 2013; Pasa and Sperduti, 2014; Advani and Saxe, 2017) study the dynamics of linear network but these results do not apply in our case due to difference loss and network architecture: (Fukumizu, 1999; Saxe et al., 2013; Advani and Saxe, 2017) study fully connected networks with L2 regression loss while Pasa and Sperduti (2014) considers recurrent networks with reconstruction loss.

6 Conclusion

We analyze the generalization performance of two layer convolutional linear classifiers trained with gradient descent. Our analysis is able to explain why, on some simple but realistic examples, adding a convolution layer can be more favorable than just using a single layer classifier even if the data is linearly separable. Our work can be a starting point for several interesting future direction: (i) Closing the gaps in normal hinge loss v.s. the extreme one as well as asymptotic analysis v.s. finite time analysis. The latter may be able to characterize how good a weight initialization is. (ii) Another interesting question is how we can interpret the generalization bias as a prior knowledge. We conjecture that the jointly trained filter works as a data adaptive bias as it requires a certain amount of data to provide the generalization bias (supported by Proposition 6. (iii) Other interesting directions include studying the choice of k, making practical suggestions based on our analysis and bringing in more factors such as feature extraction, non-linearity and pooling.

[6]See appendix for an example.

References

Advani, M. S. and Saxe, A. M. (2017). High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6241–6250.

Brutzkus, A., Globerson, A., Malach, E., and Shalev-Shwartz, S. (2017). Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*.

Du, S. S., Lee, J. D., and Tian, Y. (2017). When is a convolutional filter easy to learn? *arXiv preprint arXiv:1709.06129*.

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

Frobenius, G. F. (1912). *Über Matrizen aus nicht negativen Elementen*. Königliche Akademie der Wissenschaften.

Fukumizu, K. (1999). Generalization error of linear neural networks in unidentifiable cases. In *International Conference on Algorithmic Learning Theory*, pages 51–62. Springer.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. (2018). Implicit bias of gradient descent on linear convolutional networks. *arXiv preprint arXiv:1806.00468*.

Hardt, M., Recht, B., and Singer, Y. (2015). Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017a). Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5949–5958.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017b). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*.

Pasa, L. and Sperduti, A. (2014). Pre-training of recurrent neural networks via linear autoencoders. In *Advances in Neural Information Processing Systems*, pages 3572–3580.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Soudry, D., Hoffer, E., and Srebro, N. (2017). The implicit bias of gradient descent on separable data. *arXiv preprint arXiv:1710.10345*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

A Proof of Lemma 1

We first introduce the following Lemma, which shows that w_1^t and w_2^t can be written in closed-forms in terms of $(w_1^0, w_2^0, M_{\rm tr}, \alpha, t)$:

Lemma 7. *Let $M_{\rm tr} = U\Sigma V^T$ be (any of) its SVD such that $U \in \mathbb{R}^{d\times k}, \Sigma \in \mathbb{R}^{k\times k}, V \in \mathbb{R}^{k\times k}$, $U^TU = V^TV = VV^T = I$. Then for any $t \geq 0$*

$$\begin{aligned} w_1^t &= \frac{1}{2}V\left(\Lambda^{+,t}V^Tw_1^0 + \Lambda^{-,t}U^Tw_2^0\right), \\ w_2^t &= \frac{1}{2}U\left(\Lambda^{-,t}V^Tw_1^0 + \Lambda^{+,t}U^Tw_2^0\right) - UU^Tw_2^0 + w_2^0. \end{aligned} \tag{4}$$

where we define $\Lambda^{+,t} = (I+\alpha\Sigma)^t + (I-\alpha\Sigma)^t$ and $\Lambda^{-,t} = (I+\alpha\Sigma)^t - (I-\alpha\Sigma)^t$.

Proof. We start with stating the following facts for $\Lambda^{+,t}$ and $\Lambda^{-,t}$:

$\Lambda^{+,0} = 2I$, $\Lambda^{-,0} = 0$ and for any $t \geq 0$

$$\begin{aligned} \Lambda^{+,t+1} &= \Lambda^{+,t} + \alpha\Sigma\Lambda^{-,t}, \\ \Lambda^{-,t+1} &= \Lambda^{-,t} + \alpha\Sigma\Lambda^{+,t}. \end{aligned}$$

Now we prove (4) by induction. When $t = 0$, $w_1^0 = VV^Tw_1^0$ and $w_2^0 = UU^Tw_2^0 - UU^Tw_2^0 + w_2^0$ so (4) holds for $t = 0$. Assume Lemma (4) holds for t then consider the next step $t+1$:

$$\begin{aligned} w_1^{t+1} &= w_1^t + \alpha M_{\rm tr}^T w_2^t \\ &= \frac{1}{2}V\left(\Lambda^{+,t}V^Tw_1^0 + \Lambda^{-,t}U^Tw_2^0\right) \\ &+ \alpha V\Sigma U^T\left(\frac{1}{2}U\left(\Lambda^{-,t}V^Tw_1^0 + \Lambda^{+,t}U^Tw_2^0\right)\right. \\ &\left.-UU^Tw_2^0 + w_2^0\right) \\ &= \frac{1}{2}V\left(\Lambda^{+,t}V^Tw_1^0 + \Lambda^{-,t}U^Tw_2^0\right. \\ &\left.+\alpha\Sigma\Lambda^{-,t}V^Tw_1^0 + \alpha\Sigma\Lambda^{+,t}U^Tw_2^0\right) \\ &= \frac{1}{2}V\left(\Lambda^{+,t+1}V^Tw_1^0 + \Lambda^{-,t+1}U^Tw_2^0\right). \end{aligned}$$

Similarly, we can show

$$\begin{aligned} w_2^{t+1} &= w_2^t + \alpha M_{\rm tr} w_1^t \\ &= \frac{1}{2}U\left(\Lambda^{-,t+1}V^Tw_1^0 + \Lambda^{+,t+1}U^Tw_2^0\right) \\ &- UU^Tw_2^0 + w_2^0. \end{aligned}$$

Thus (4) holds for all $t \geq 0$. □

Proof of Lemma 1. Taking $w_2^0 = 0$ in Lemma 7 we can write $w_1^t = \frac{1}{2}V\Lambda^{+,t}V^Tw_1^0$ and $w_2^t = \frac{1}{2}U\Lambda^{-,t}V^Tw_1^0$

For $1 \leq i \leq m$, $\sigma_i = \sigma_1$ thus

$$\lim_{t\to+\infty}\frac{(1+\alpha\sigma_i)^t}{(1+\alpha\sigma_1)^t} = 1. \tag{5}$$

For $m < i \leq k$, $\sigma_i < \sigma_1$ thus

$$\lim_{t\to+\infty}\frac{(1+\alpha\sigma_i)^t}{(1+\alpha\sigma_1)^t} = 0. \tag{6}$$

For any $1 \leq i \leq k$, we have $\frac{1-\alpha\sigma_i}{1+\alpha\sigma_1} \leq \frac{1}{1+\alpha\sigma_1} < 1$ and $\frac{1-\alpha\sigma_i}{1+\alpha\sigma_1} \geq \frac{1-\alpha\sigma_1}{1+\alpha\sigma_1} = -1 + \frac{2}{1+\alpha\sigma_1} > -1$ thus

$$\lim_{t\to+\infty}\frac{(1-\alpha\sigma_i)^t}{(1+\alpha\sigma_1)^t} = 0. \tag{7}$$

Applying (5)—(7) to compute the limits in (2) gives the result in Lemma 1. □

B Proof of Theorem 2

Proof. For any vector $z \in \mathbb{R}^m$ such that $\|z\|_2 = 1$, we have

$$\begin{aligned} M_{\rm tr}V_{:m}z &= U\Sigma V^TV_{:m}z = \sigma_1U_{:m}z, \\ M_{\rm tr}^TU_{:m}z &= V\Sigma U^TU_{:m}z = \sigma_1V_{:m}z, \end{aligned}$$

Since

$$\begin{aligned} \|V_{:m}z\|_2^2 &= z^TV_{:m}^TV_{:m}z = 1, \\ \|U_{:m}z\|_2^2 &= z^TU_{:m}^TU_{:m}z = 1 \end{aligned}$$

we know that $(U_{:m}z, V_{:m}z)$ is also a pair of left-right singular vectors with singular value σ_1. Therefore, when $V_{:m}^Tw_1^0 \in \mathbb{R}^m$ is non-zero $\left(\frac{U_{:m}V_{:m}^Tw_1^0}{\|V_{:m}^Tw_1^0\|_2}, \frac{V_{:m}V_{:m}^Tw_1^0}{\|V_{:m}^Tw_1^0\|_2}\right)$ is also such a pair. Following (3) we have

$$\begin{aligned} \frac{\mathcal{F}^\infty(x,y,w_1^0,D_{\rm tr})}{\|V_{:m}^Tw_1^0\|_2^2} &= \left(\frac{V_{:m}V_{:m}^Tw_1^0}{\|V_{:m}^Tw_1^0\|_2}\right)^T M_{x,y}^T\left(\frac{U_{:m}V_{:m}^Tw_1^0}{\|V_{:m}^Tw_1^0\|_2}\right) \\ &\geq \min_{(u,v)\in UV_1^{M_{\rm tr}}} v^TM_{x,y}^Tu \end{aligned} \tag{8}$$

for any w_1^0 such that $V_{:m}^Tw_1^0 \neq \vec{0}$.

When $w_1^0 \sim \mathcal{N}(0, b^2I_k)$, for any fixed $V_{:m}$ satisfying $V_{:m}^TV_{:m} = I_m$, the random variable $V_{:m}^Tw_1^0$ also follows a normal distribution:

$$\mathbb{E}\left[V_{:m}^Tw_1^0(V_{:m}^Tw_1^0)^T\right] = V_{:m}^T\mathbb{E}\left[w_1^0w_1^{0T}\right]V_{:m} = b^2I_m$$

hence $V_{:m}^Tw_1^0 \sim \mathcal{N}(0, b^2I_m)$.

Applying the fact that $\bar{\mathcal{E}}(\cdot) \leq 1$ is non-increasing and $\bar{\mathcal{E}}(\alpha x) = \bar{\mathcal{E}}(x)$ for any $\alpha > 0$ we can upper bound (3) by

$\mathcal{E}_{\rm Convk}^\infty(\mathcal{D})$

$$
\begin{aligned}
&= \mathbb{E}_{w_1^0, D_{\text{tr}}, (x,y)} \left[\bar{\mathcal{E}} \left(\mathcal{F}^\infty(x, y, w_1^0, D_{\text{tr}}) \right) \right] \\
&= \mathbb{E}_{D_{\text{tr}}, (x,y)} \left[\mathbb{E}_{w_1^0} \left[\bar{\mathcal{E}} \left(\mathcal{F}^\infty(x, y, w_1^0, D_{\text{tr}}) \right) \right] \right] \\
&= \mathbb{E}_{D_{\text{tr}}, (x,y)} \left[\Pr \left(V_{:m}^T w_1^0 = \vec{0} \right) \mathbb{E}_{w_1^0} \left[\bar{\mathcal{E}} \left(\mathcal{F}^\infty \right) | V_{:m}^T w_1^0 = \vec{0} \right] \right. \\
&\left. + \Pr \left(V_{:m}^T w_1^0 \neq \vec{0} \right) \mathbb{E}_{w_1^0} \left[\bar{\mathcal{E}} \left(\mathcal{F}^\infty \right) | V_{:m}^T w_1^0 \neq \vec{0} \right] \right] \\
&= \mathbb{E}_{D_{\text{tr}}, (x,y)} \left[\mathbb{E}_{w_1^0} \left[\bar{\mathcal{E}} \left(\frac{\mathcal{F}^\infty}{\| V_{:m}^T w_1^0 \|_2^2} \right) | V_{:m}^T w_1^0 \neq \vec{0} \right] \right] \\
&\leq \mathbb{E}_{D_{\text{tr}}, (x,y)} \left[\bar{\mathcal{E}} \left(\min_{(u,v) \in U V_1^{M_{\text{tr}}}} v^T M_{x,y}^T u \right) \right] .
\end{aligned}
$$

□

C Perron-Frobenius Theorem

Let $A \in \mathbb{R}^{k \times k}$ be a non-negative square matrix[7]:

- **Definition:** A is *primitive* if there exists a positive integer t such that $A_{ij}^t > 0$ for all i, j.
- **Definition:** A is *irreducible* if for any i, j there exists a positive integer t such that $A_{ij}^t > 0$.
- **Definition:** Its *associated graph* $\mathcal{G}_A = (V, E)$ is defined to be a directed graph with $V = \{1, ..., k\}$ and $(i, j) \in E$ iff $A_{ij} \neq 0$. $\mathcal{G}_A$ is said to be *strongly connected* if for any i, j there is path from i to j.
- **Property:** A is irreducible iff $\mathcal{G}_A$ is strongly connected.
- **Property:** If A is irreducible and has at least one non-zero diagonal element then A is primitive.
- **Property:** If A is primitive then its first eigenvalue is unique ($\lambda_1 > \lambda_2$) and the corresponding eigenvector is all-positive (or all-negative up to sign flipping).

D Proof of Theorem 4

Proof. Following (3) and let

$$
\hat{\mathcal{E}}(x, y, D_{\text{tr}}) = \bar{\mathcal{E}} \left(\min_{(u,v) \in U V_1^{M_{\text{tr}}}} v^T M_{x,y}^T u \right) \leq 1
$$

we have

$$
\begin{aligned}
&\mathcal{E}_{\text{Convk}}^\infty(\mathcal{D}) \leq \mathbb{E}_{D_{\text{tr}}, (x,y)} \left[\hat{\mathcal{E}}(x, y, D_{\text{tr}}) \right] \\
&= \mathbb{E}_{D_{\text{tr}}, (x,y)} \left[\left(\mathbb{I} \left\{ \Omega^c(M_{\text{tr}}^T M_{\text{tr}}) \right\} + \mathbb{I} \left\{ \Omega(M_{\text{tr}}^T M_{\text{tr}}) \right\} \right) \right. \\
&\left. \hat{\mathcal{E}}(x, y, D_{\text{tr}}) \right] \\
&\leq \mathbb{E}_{D_{\text{tr}}} \left[\mathbb{I} \left\{ \Omega^c(M_{\text{tr}}^T M_{\text{tr}}) \right\} \right] \\
&+ \mathbb{E}_{D_{\text{tr}}, (x,y)} \left[\mathbb{I} \left\{ \Omega(M_{\text{tr}}^T M_{\text{tr}}) \right\} \hat{\mathcal{E}}(x, y, D_{\text{tr}}) \right]
\end{aligned}
$$

[7] https://en.wikipedia.org/wiki/Perron-Frobenius_theorem .

$$
\begin{aligned}
&= \Pr \left(\Omega^c(M_{\text{tr}}^T M_{\text{tr}}) \right) \\
&+ \mathbb{E}_{D_{\text{tr}}, l \sim \mathcal{U}[d]} \left[\mathbb{I} \left\{ \Omega(M_{\text{tr}}^T M_{\text{tr}}) \right\} \hat{\mathcal{E}}(e_l, 1, D_{\text{tr}}) \right] \qquad (9)
\end{aligned}
$$

Now look at the second term in (9). If $M_{\text{tr}}^T M_{\text{tr}}$ is primitive then its first eigenvalue $\lambda_1 = \sigma_1^2$ is unique ($\sigma_1 > \sigma_2$) and the corresponding eigenvector v is all positive (or all negative if we flip the sign of v and u, which does not change the sign of $v^T M^T u$ thus it is safe to assume $v > 0$). $u = M_{\text{tr}} v / \sigma_1$ gives that u is also unique and non-negative. Since $M_{x,y}$ is also non-negative we have $v^T M_{x,y}^T u \geq 0$ for any x, y. Therefore,

$$
\begin{aligned}
\hat{\mathcal{E}}(x, y, D_{\text{tr}}) &= \bar{\mathcal{E}} \left(v^T M_{x,y}^T u \right) \\
&= \mathbb{I} \left\{ v^T M_{x,y}^T u < 0 \right\} + \frac{1}{2} \mathbb{I} \left\{ v^T M_{x,y}^T u = 0 \right\} \\
&= \frac{1}{2} \mathbb{I} \left\{ v^T M_{x,y}^T u = 0 \right\} .
\end{aligned}
$$

From $u = M_{\text{tr}} v / \sigma_1$ and $v > 0$ we know that $u_i > 0$ iff there exists $1 \leq j \leq k$ such that $(M_{\text{tr}})_{i,j} > 0$, which is equivalent to that there exists $i \leq l < i + k$ such that $l \in S_{\text{tr}}$. Also for $x = e_l$ ($y = 1$), according to the definition of $M_{x,y}$ and the fact that $v > 0$ we have $v^T M_{e_l,1}^T u > 0$ iff there exists $l - k < i \leq l$ such that $u_i > 0$. So we have

$$
v^T M_{e_l,1}^T u > 0 \iff \exists l' \in \bigcup_{l-k<i\leq l} [i, i+k) \text{ s.t. } l' \in S_{\text{tr}}
$$

Since $v^T M_{e_l,1}^T u \geq 0$ and $\bigcup_{l-k<i\leq l} [i, i+k) = (l-k, l+k)$ we have

$$
v^T M_{e_l,1}^T u = 0 \iff \forall l' \in S_{\text{tr}}, |l' - l| \geq k .
$$

Now we have proved that, if $M_{\text{tr}}^T M_{\text{tr}}$ is primitive then

$$
\hat{\mathcal{E}}(e_l, 1, D_{\text{tr}}) = \frac{1}{2} \mathbb{I} \left\{ \forall l' \in S_{\text{tr}}, |l' - l| \geq k \right\} ,
$$

which means that

$$
\mathbb{I} \left\{ \Omega(M_{\text{tr}}^T M_{\text{tr}}) \right\} \hat{\mathcal{E}}(e_l, 1, D_{\text{tr}}) \leq \frac{1}{2} \mathbb{I} \left\{ \forall l' \in S_{\text{tr}}, |l' - l| \geq k \right\}
$$

holds for any D_{tr}. Therefore

$$
\begin{aligned}
&\mathbb{E}_{D_{\text{tr}}, l \sim \mathcal{U}[d]} \left[\mathbb{I} \left\{ \Omega(M_{\text{tr}}^T M_{\text{tr}}) \right\} \hat{\mathcal{E}}(e_l, 1, D_{\text{tr}}) \right] \\
&\leq \frac{1}{2} \mathbb{E}_{D_{\text{tr}}, l \sim \mathcal{U}[d]} \left[\mathbb{I} \left\{ \forall l' \in S_{\text{tr}}, |l' - l| \geq k \right\} \right] \\
&= \frac{1}{2} \mathbb{E}_{l \sim \mathcal{U}[d]} \left[\Pr \left(\forall l' \in S_{\text{tr}}, |l' - l| \geq k \right) \right]
\end{aligned}
$$

which concludes the proof. □

E Proof of Lemma 5

Proof. If $k \leq i \leq d$ and $i - 1, i \in S_{\text{tr}}$ then for any $1 \leq j \leq k$ we have $(M_{\text{tr}})_{i-j,j} > 0$ and $(M_{\text{tr}})_{i-j+1,j} > 0$,

which also means that for any $1 \leq j < k$ we have $(M_{\text{tr}})_{i-j,j} > 0$ and $(M_{\text{tr}})_{i-j,j+1} > 0$. Since every two adjacent columns have at least one common non-zero position what we have is $(M_{\text{tr}}^T M_{\text{tr}})_{j,j+1} > 0$ and $(M_{\text{tr}}^T M_{\text{tr}})_{j+1,j} > 0$ for all $1 \leq j < k$. So its associated graph $\mathcal{G}_{M_{\text{tr}}^T M_{\text{tr}}}$ is strongly connected thus $M_{\text{tr}}^T M_{\text{tr}}$ is irreducible. It is also true that all diagonal elements of $M_{\text{tr}}^T M_{\text{tr}}$ are positive since every column of M_{tr} must contain at least one non-zero element. Now we have proved that $M_{\text{tr}}^T M_{\text{tr}}$ is primitive because it is irreducible and has at least one non-zero element on its diagonal. □

F Proof of Proposition 6

Proof. Let $n = |S_{\text{tr}}|$. Then given the conditions in this proposition we can see that any column in M_{tr} has exactly n non-zero entries with value $1/n$ and any two columns in M_{tr} has no overlapping non-zero positions. Hence we have $M_{\text{tr}}^T M_{\text{tr}} = \frac{1}{n} I_k$ so that $m = k$ in Lemma 1 and $VV^T = I$. Applying Lemma 1 we have $w_1^\infty = w_1^0$ and $w_2^\infty = nM_{\text{tr}} w_1^0$. Then for any $x = e_l$ we have

$$y f_{w^\infty}(x) = w_1^{\infty T} M_{x,y}^T w_2^\infty = n w_1^{0T} A_x^T M_{\text{tr}} w_1^0 .$$

For x to be correctly classified we need $y f_{w^\infty}(x) > 0$. We will show that this is guaranteed only when $l \in S_{\text{tr}}$, i.e. x or $-x \in D_{\text{tr}}$.

Since for any $l, l' \in S_{\text{tr}}$, $|l - l'| \geq 2k$ we know that there exist at most one $l' \in S_{\text{tr}}$ such that $|l - l'| < k$.

If there does not exist such l' then $A_x^T M_{\text{tr}} = 0$ and $y f_{w^\infty}(x) = 0$, which means x is classified randomly.

If there exists a unique l' such that $|l - l'| < k$ and let $s = |l - l'|$, we have that

$$y f_{w^\infty}(x) = n w_1^{0T} A_x^T M_{\text{tr}} w_1^0 = \sum_{i=1}^{k-s} w_{1,i}^0 w_{1,i+s}^0 .$$

When $l \in S_{\text{tr}}$, which means $s = 0$, we have $y f_{w^\infty}(x) = w_1^{0T} w_1^0 > 0$ when $w_1^0 \neq 0$ (which holds almost surely).

When $0 < s < k$ it is not guaranteed that $\sum_{i=1}^{k-s} w_{1,i}^0 w_{1,i+s}^0 > 0$ under $w_1^0 \sim \mathcal{N}(0, b^2 I)$. Actually we can show that the distribution of this quantity is symmetric around 0: For any s we can draw a graph with k nodes and every $(i, i+s)$ forms an edge. This graph contains s independent chains so we can choose a set of nodes $S \subset [k]$ such that for any edge exactly one of the two nodes is contained in S. Now for any w_1^0 if we flip the sign at the positions that belong to S then the sign of $\sum_{i=1}^{k-s} w_{1,i}^0 w_{1,i+s}^0$ is also flipped. With $w_1^0 \sim \mathcal{N}(0, b^2 I)$ this indicates that $P(\sum_{i=1}^{k-s} w_{1,i}^0 w_{1,i+s}^0 > 0) = 1/2$.

Now we have shown that, under the condition in this proposition, a data sample is correctly classified by Conv-k with w^∞ if and only if this sample appears in the training set. Otherwise it has only a half change to be correctly classified. This generalization behavior is exactly the same as Model-1-Layer in Task-Cls, which concludes the proof.

□

G A Supporting Evidence for Interpreting Conv-Filters as a Data Adaptive Bias

We have shown that, different from typical regularizations, the bias itself may require some samples to be built up (see Figure 4(b)). We conjecture that convolution layer adds a data adaptive bias: The set of possible filters forms a set of biases. With a few number of samples gradient descent is able to figure out which bias(filter) is more suitable for the dataset. Then the identified bias can play as a prior knowledge to reduce the sample complexity. We provide another evidence for this: Let the dataset contains all $e_l, l \in [d]$ while $y_{e_l} = +1$ if l is odd and -1 is l is even. Model-Conv-k is still able to outperform Model-1-Layer on this task (see Figure 7). We observe that the sign of the learned filter looks like (+, -, +, -, ...) in contrast to the ones learned in our three tasks, which are likely to be all positive or all negative. This indicates that, besides spatial shifting invariance, jointly training the convolutional filter can exploit a broader set of structures and be adaptive to different data distributions.

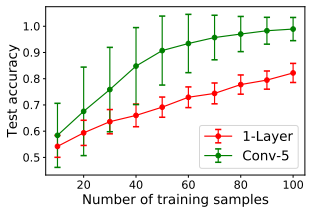
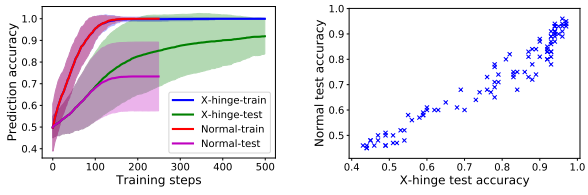


Figure 7: Classifying even v.s. odd non-zero position. Settings are the same as in Figure 2.

H Correlation Between Normal-hinge and X-hinge under Different Initializations

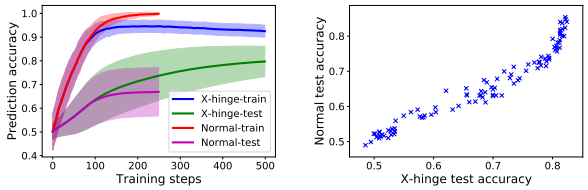
Figure 8 and 9 shows the variance introduced by weight initialization is also strongly correlated under two losses in Task-1stCtrl and Task-3rdCtrl. Figure 9(a) looks a bit different from the other two tasks because the extreme hinge loss is biased and w^∞ may

not able to separate the training samples in Task-3rdCtrl. But the strong correlation between the normal hinge loss and the extreme hinge loss under different weight initializations still holds.



(a) Convergence with t. (b) Correlation at $t = 150$.

Figure 8: The effect of weight initialization in Task-1stCtrl. We fix $d = 100$, $n = 30$ and train Model-Conv-k with 100 different random initializations using both losses.



(a) Convergence with t. (b) Correlation at $t = 150$.

Figure 9: The effect of weight initialization in Task-3stCtrl. We fix $d = 100$, $n = 50$ and train Model-Conv-k with 100 different random initializations using both losses.

I The bias of X-Hinge in Task-3rdCtrl and Potential Practical Indications

In Figure 9(a) we observe that running gradient descent may not be able to achieve 0 training error even if the samples are linearly separable. To explain this, simply consider a training set with 3 samples and $k = 1, d = 4$: $x_1 = [-1, 1, 0, 0], x_2 = [0, -1, 1, 0], x_3 = [0, 0, -1, 1]$. All labels are positive. Then $M_{\text{tr}} = [-1/3, 0, 0, 1/3]$. If we optimize the X-hinge loss then the network has no intent to classify x_2 correctly.

Notice that in Figure 9(a), under X-hinge, the generalization performance is still improving even after the training accuracy starts to decrease. We conjecture that this indicates a new way of interpreting the role of regularization in deep nets. On real datasets we typically use sigmoid with cross entropy loss which can be viewed and a smoothed version of the hinge loss. We say a data sample is *active* during training if $yf(x)$ is small so that the gradient for fitting (x, y) is salient since it is not well fit yet. With X-hinge all samples are "equality active". One message delivered by our observation is that having more samples to be "active" during training will make convolution filters have better generalization property, but may hurt with training data fitting. In practice we cannot recommend using X-hinge loss since the network will fail to fit the training set if we keep *all* samples to be equally "active". But we can view this as a trade off when using logistic loss: keeping more samples to be "active" during training with gradient descent will help with some generalization property (e.g. better Conv filters) but cause underfitting. For regularization we may want to keep as many samples to be active as possible while still be able to fit the training samples. This provides a new view of the role of regularization: Taking weight norm regularization as an example, traditional interpretation is that controlling the weight norm will reduce the capacity of neural nets, which may not be sufficient to explain non-overfitting in very large nets. The new potential interpretation is that, if we keep the weight norm to be small during training, the training samples are more "active" during gradient descent so that better convolution filters can be learned for generalization purposes. Verifying this conjecture on real datasets will be an interesting future direction.