Channel Normalization in Convolutional Neural Networks avoids Vanishing Gradients

Zhenwei Dai[†] and Reinhard Heckel[⋆,*]

[*]Dept. of Electrical and Computer Engineering and Dept. of Statistics[†], Rice University
[⋆]Dept. of Electrical and Computer Engineering, Technical University of Munich

June 28, 2019

Abstract

Normalization layers are widely used in deep neural networks to stabilize training. In this paper, we consider the training of convolutional neural networks with gradient descent on a single training example. This optimization problem arises in recent approaches for solving inverse problems such as the deep image prior or the deep decoder. We show that for this setup, channel normalization, which centers and normalizes each channel individually, avoids vanishing gradients, whereas without normalization, gradients vanish which prevents efficient optimization. This effect prevails in deep single-channel linear convolutional networks, and we show that without channel normalization, gradient descent takes at least exponentially many steps to come close to an optimum. Contrary, with channel normalization, the gradients remain bounded, thus avoiding exploding gradients.

1 Introduction

Deep learning and in particular convolutional neural networks have significantly improved the state-of-the-art in computer vision, image generation, and computational imaging, among many other fields. Deep neural networks are typically trained using first order methods such as gradient descent and the stochastic gradient method. However, the corresponding loss function is non-convex and therefore, depending on the initialization, convergence to an optimum is not guaranteed, and first order methods sometimes suffer from unstable training and/or vanishing or exploding gradients.

Normalization layers are widely used to avoid vanishing or exploding gradients, stabilize training, and enable learning with higher rates and faster convergence. The perhaps most popular normalization technique is batch normalization [IS15]; but a number of (often closely related) variations and alternatives have been proposed such as layer normalization [LB+16], weight normalization [SK16], and instance normalization [Uly+16].

A variety of recent works have proposed different explanations for the success of normalization layers. The original batch normalization paper [IS15] suggested that batch normalization aids optimization by reducing a quantity called internal covariate shift. In contrast, Santurkar et al. [San+18] reason that batch normalization reparameterizes the underlying optimization problem and thereby make its landscape significantly smoother. Kohler et al. [Koh+19] linked batch normalization to weight normalization [SK16], and pointed out that batch normalization accelerates the training process by splitting the optimization task into optimizing the length and direction of the parameters separately, and Bjorck [Bjo+18] argues that (batch) normalization enables training with larger training rates. We add that whether normalization layers are useful or not depends strongly on the architecture and initialization. For example carefully initialized deep residual networks can be trained without any normalization layers [Zha+19].

In this paper, we study channel normalization, which is a special case of a number of the above mentioned normalization techniques, in the context of a convolutional generator network. Channel normalization standardizes each channel in a convolutional neural network, individually for each

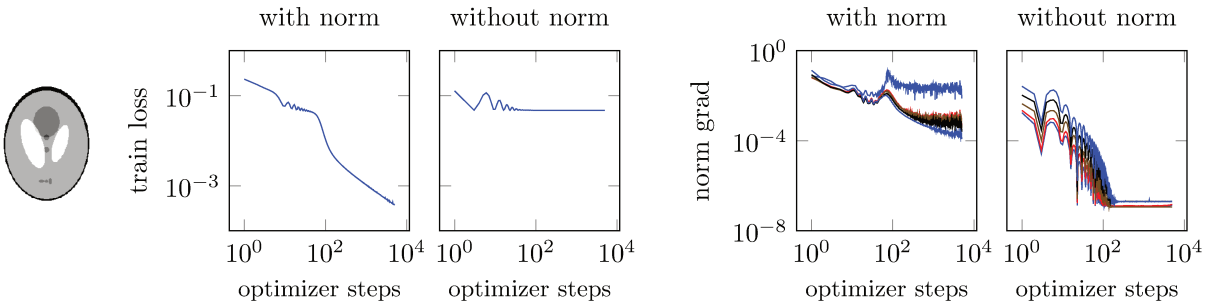

Figure 1: The train loss and the norm of the gradients in each of the 5 layers of a convolutional neural generator network with and without channel normalization for fitting the phantom MRI image: Without normalization the gradients vanish before gradient descent reaches a good solution.

training example, and scales and shifts the resulting vector with a (trainable) scalar. Channel normalization is equivalent to instance normalization [Uly+16] and to batch normalization for a single training example (then the batch size is one).

We first train a convolutional network with gradient descent on a single training example, a problem that occurs in solving inverse problems without training data [HH19; Uly+18], and demonstrate that channel normalization avoids exploding and vanishing gradients and enables reaching a close-to-optimal point. Contrary, without channel normalization, gradient descent does not converge to an optimum in a reasonable number of iterations. We then show analytically, for a special case of linear convolutional networks, that without channel normalization, gradient descent requires at least exponentially many steps to converge under mild initialization conditions.

The aforementioned works [IS15; San+18; Koh+19] have studied normalization techniques by focusing on shallow networks (i.e., networks with one hidden layer), since analytical gradient expressions for deep networks with non-linearities are almost intractable. Here, we sidestep this hurdle by exploring a simpler model, specifically a linear convolutional network with a single channel. Studying such a simple model is justified by observing that even for this simple model, normalization is critical for fast convergence.

2 Channel normalization

We start by introducing channel normalization and then show empirically that it is critical for running gradient descent efficiently on a convolutional generators trained on a single example.

The channel normalization operation normalizes each channel of a convolutional network individually. Let $\mathbf{z}_{ij}$ be the input of the j-th channel and the i-th layer. Channel normalization performs the transformation

$$\mathbf{z}'_{ij} = \frac{\mathbf{z}_{ij} - \text{mean}(\mathbf{z}_{ij})}{\sqrt{\text{var}(\mathbf{z}_{ij}) + \epsilon}} \gamma_{ij} + \beta_{ij},$$

where mean and var compute the empirical mean and variance, $\gamma_{i,j}$ and β_{ij} are parameters learned independently for each channel, and ϵ is a fixed small constant added for numerical stability.

We consider a variant of the deep decoder introduced in [HH19]. The network works well for image compression and for regularizing a variety of inverse problems, when trained or fitted to a single image only. Specifically, we consider an extremely simple convolutional generator consisting of $d = 5$ many 3x3 convolutional layers, followed by channel normalization and ReLU activation functions. Each layer has $k = 32$ channels, and the last layer is a 1x1 convolutional layer mapping the k channels to a single, 256x256 grayscale output image. The input to the network is a

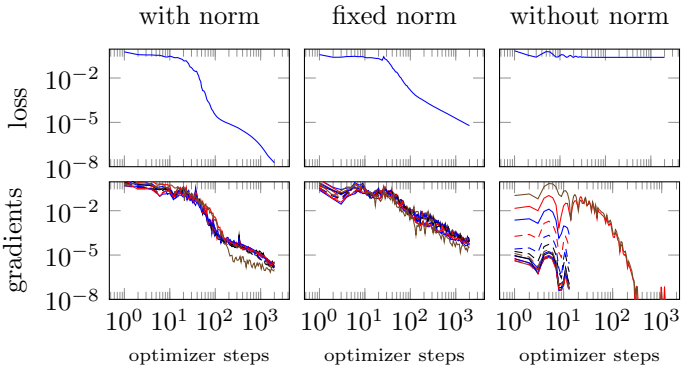

Figure 2: The training error and the norms of the gradients of each layer verses the number of gradient descent steps: Both channel normalization with fixed scale and bias parameters and learned ones enable efficiently finding an optimum.

32x256x256 volume that is chosen randomly and is fixed (i.e., we do not optimize over the input). Given an image $\mathbf{x}^*$ we then fit the parameters of the network (i.e., the weights) by minimizing the loss $L(\mathbf{C}) = \left\|G(\mathbf{C}) - \mathbf{x}^*\right\|_2^2$ with respect to the network parameters $\mathbf{C}$ using plain gradient descent with fixed stepsize.

Figure 1 shows the results for the phantom MRI image for a network with and without channel normalization. With channel normalization, the training loss converges rapidly and the gradients do not vanish. Contrary, without normalization, the network does not converge to a small error (even though the network has the capacity to represent the image), and the gradients vanish. This effect is not specific to the image (we have reproduced it using 100 randomly chosen images from imagenet), and it is also reproducible for a number of related convolutional generators, for example networks including upsampling operations.

3 Isolating the effect of channel normalization

We next show that to achieve the stabilizing effect of channel normalization, the trainable coefficients γ_{ij} and β_{ij} do not need to be learned and can be set to one and zero, respectively. We also demonstrate that even for linear networks, channel normalization is critical to avoid vanishing gradients. This justifies our theoretical study of linear networks in the next section.

Multiple Channels CNN: We first consider a one-dimensional convolutional network, again only consisting of convolutional layers followed by channel normalization and ReLU activation functions. We set the dimension of input/output vector to $n = 256$, the number of channels to $k = 4$, number of hidden layers to $d = 12$, and convolutional kernel size to 3. The entries of the input vector $\mathbf{x}$ are sampled from a standard uniform distribution. As before, we minimize the least squares loss with respect to the weight parameters using gradient descent with fixed step size. We consider three different normalization operations: the original channel normalization, a variant where $\gamma_{ij} = 1$ and $\beta_{ij} = 0$ (called fixed norm), and no normalization. We consider the problem of fitting a simple step function. The results reported in Figure 2 show that both normalization versions enable efficient optimization with gradient descent (the training error is near-zero), whereas without channel normalization the training error does not improve after a few iterations and the gradients vanish.

Single Channel Linear CNN: Next, we consider an even simpler network with only one channel and without activation functions. Without normalization, the network is linear. We set the dimension of input/output vector to $n = 64$, number of hidden layers $d = 10$, and convolution

kernel size $k = 9$. The results are very similar to the previous experiment in Figure 2 (see Figure 5 in the appendix), and demonstrates the critical role of channel normalization.

We also evaluated the loss function landscape around the point at convergence (Figure 3). For both multi-channel CNNs and linear CNNs, without normalization, the loss function becomes very flat, in comparison to the more steep loss surface pertaining to the case with channel normalization. This indicates that the gradients around the point of convergence are close to 0, and gradient descent makes little to no progress if the iterates fall into such flat regions.

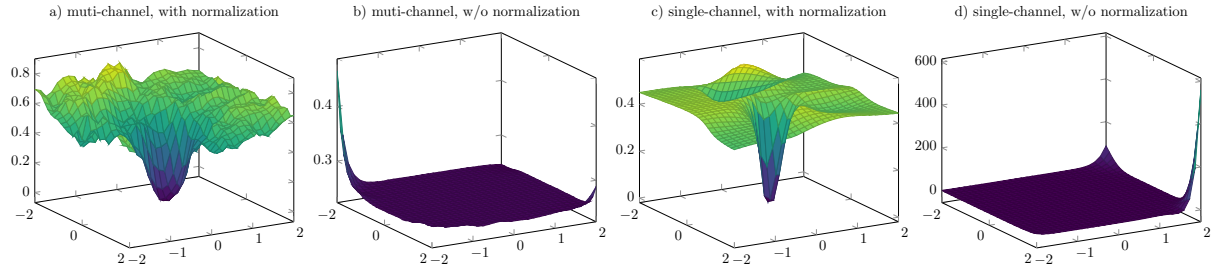

Figure 3: Panels a and b show the landscape around the point of convergence with and without channel normalization for multi-channel CNNs with $d = 10$, filter size 9, and number of channels 4; panels c and d contain the same plots but for a single channel linear CNN with $d = 10$ and filter size 9.

4 Theoretical analysis

From the previous section, we know that channel normalization avoids vanishing gradients even for linear, one-layer convolutional neural networks, and that the scale and shift parameters can be set to one and zero. In this section, we provide theoretical justification for the difficulty of optimization in the absence of normalization, and justification for the stabilizing effect of channel normalization.

Throughout this section, we consider a single channel linear convolutional neural network with d layers, with output given as $f(\mathbf{x}, \mathbf{w}) = \prod_{i=1}^{d} \mathbf{W}_i \mathbf{x}$, where $\mathbf{W}_i \in \mathbb{R}^{n \times n}$ are circulant matrices implementing the convolution operation, and $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d)$ is the set of weights or convolutional filters, given by the first columns of the respective circulant matrices $\mathbf{W}_i, \ldots, \mathbf{W}_d$ (which define all other entries of the matrices). We study gradient descent applied to the squared loss function $L(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \frac{1}{2}\big\|\mathbf{y} - f(\mathbf{x}, \mathbf{w})\big\|_2^2$. We start by showing that without channel normalization gradient descent needs at least exponentially many steps to converge under a standard initialization scheme.

Theorem 1. *Suppose that the signal* $\mathbf{y}$ *doesn't vanish, i.e.,* $\|\mathbf{y}\|/\|\mathbf{x}\| \geq dn^{d/2}\tau$*, where* τ *is a constant. Moreover, suppose that* $\mathbf{y}$ *is in the range of the generator* f *and that the initial weights are Gaussian random variables with zero mean and covariance matrix* $(1/n^2)\mathbf{I}$*. Then gradient descent with constant stepsize* $\eta \leq \exp(cd)$ *runs at least for* $\exp(\Omega(d))$ *steps until it reaches a point that is* c' *close to optimal with probability larger than* $1 - \exp(-\Omega(d))$*. Here,* c *and* c' *are constants independent of* d*.*

The proof, deferred to the appendix, relies on diagonalizing the circulant matrices $\mathbf{W}_i$ using the Fourier transform. Then, the optimization problem reduces to n one-dimensional problems, and we can build on results by Shamir [Sha18] on the hardness of optimizing one-dimensional deep neural networks.

Theorem 1 shows that the number of steps to come close to the optimum is at least exponential in the network depth d, even when the stepsize is large (exponential in d). This can be interpreted

as a case of gradient vanishing. Also note that if we initialize the weight away from 0 with another initialization scheme, e.g., $\mathbf{w}_k$ is initialized following Gaussian distribution with covariance matrix $\mathbf{I}$, then the norms of the gradients increase exponentially fast with the network depth, and the network becomes difficult to optimize due to exploding gradients.

Next, we evaluate the effect of channel normalization on the gradients. Since our experiments have shown that fixing scale and shift parameters or learning them yields comparable performance, we focus on the case where they are fixed to $\gamma_{ij} = 1$ and $\beta_{ij} = 0$. Suppose that the input $\mathbf{x}$ has zero mean. Then the gradients pertaining to the loss function with normalized loss are (see Appendix C):

$$\nabla_{\mathbf{w}_k} L_N(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \mathbf{X}_k^T \frac{w_{d+1}}{\|\mathbf{X}_k \mathbf{w}_k\|} \left(\mathbf{I} - \mathbf{X}_k \mathbf{w}_k \mathbf{w}_k^T \mathbf{X}_k^T / \|\mathbf{X}_k \mathbf{w}_k\|^2 \right) \mathbf{y}, \quad \mathbf{X}_k = \prod_{i \neq k} \mathbf{W}_i \mathbf{X},$$

where $\mathbf{X}$ is the circulant matrix with first column $\mathbf{x}$, and w_{d+1} is a scale parameter that we optimize over and necessary so that the range of the network can exhaust $\mathbb{R}^n$. By this expression, the gradients are obtained by projecting $\mathbf{y}$ onto the orthogonal complement of the estimate at the k-th iteration, $\mathbf{X}_k \mathbf{w}_k$, followed by multiplication with $\mathbf{X}_k^T / \|\mathbf{X}_k \mathbf{w}_k\|$. In contrast, the norm of the un-normalized gradients is given by $\nabla_{\mathbf{w}_k} L(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \mathbf{X}_k^T (\mathbf{y} - \mathbf{X}_k \mathbf{w}_k)$. In Figure 4 Panel (a) and (b) we plot the distribution with and without normalization at initialization for a network with $n = 100$ and $d = 6$ layers. Note that the loss typically diverges at the first few iterations, which justifies considering the gradients at initialization. The results show that the normalization leads to the gradients to be significantly better behaved, i.e., the distribution does have a significantly smaller tail.

In Figure 4 Panel (c) and (d) we plot the distribution for a multi-channel CNN with ReLU activation functions, and likewise, the results shows that without channel normalization, the tail is significantly larger. Thus, without normalization for a given stepsize the network is much more susceptible to vanishing or exploding gradients.

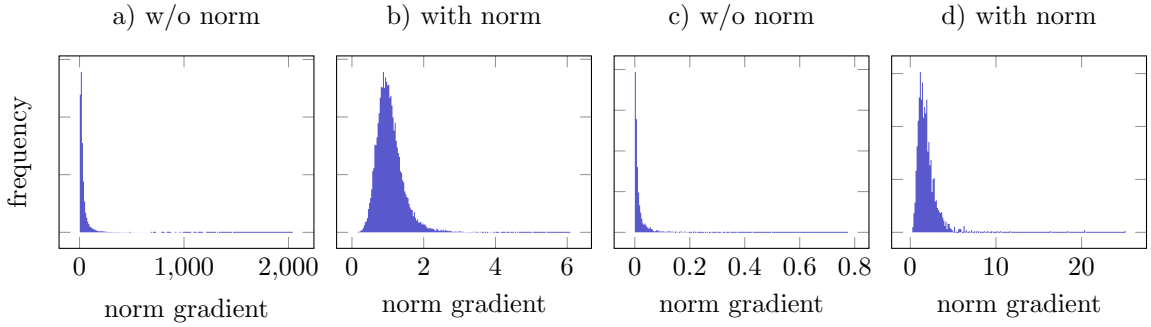

Figure 4: Panel a) and b) show the distribution of the gradients at initialization with normalization and without normalization for a single channel linear CNN with $n = 100$ and $d = 6$; c) and d) show that for a multiple channel CNN with $n = 64$, $d = 6$, filter size 9 and number of channels 4.

Code

Code to reproduce the experiments is available at github.com/reinhardh/normalization_dnns.

Acknowledgements

RH and ZD are partially supported by NSF award IIS-1816986.

References

[Bjo+18] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger. "Understanding batch normalization". In: *Advances in Neural Information Processing Systems*. 2018.

[HH19] R. Heckel and P. Hand. "Deep Decoder: Concise image representations from untrained non-convolutional networks". In: *International Conference on Learning Representations*. 2019.

[IS15] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[Koh+19] J. Kohler, H. Daneshmand, A. Lucchi, T. Hofmann, M. Zhou, and K. Neymeyr. "Exponential convergence rates for Batch Normalization: The power of length-direction decoupling in non-convex optimization". In: *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*. 2019.

[LB+16] J. Lei Ba, J. R. Kiros, and G. E. Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[SK16] T. Salimans and D. P. Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *Advances in Neural Information Processing Systems*. 2016.

[San+18] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. "How does batch normalization help optimization?" In: *Advances in Neural Information Processing Systems*. 2018.

[Sha18] O. Shamir. "Exponential convergence time of gradient descent for one-dimensional deep linear neural networks". In: *arXiv preprint arXiv:1809.08587* (2018).

[Uly+18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep image prior". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[Uly+16] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint arXiv:1607.08022* (2016).

[Zha+19] H. Zhang, Y. N. Dauphin, and T. Ma. "Fixup initialization: Residual learning without normalization". In: *International Conference on Learning Representations*. 2019.

Appendix

A Convergence for linear single layer convolutional networks

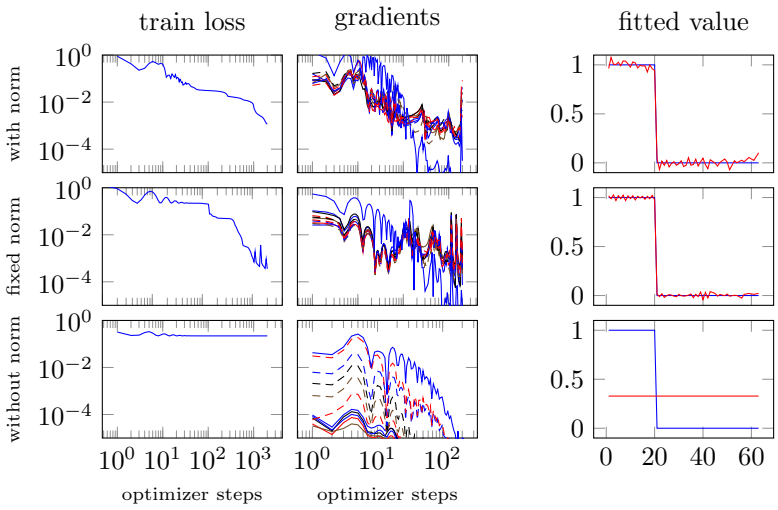

Figure 5: The left panel shows the training error and gradients of different layers over the number of gradient descent steps for optimizing a linear network (no activation functions) with a single channel in each layer only. The right panel shows the fitted response (red) and true response (blue). The results show that normalization is critical for reaching a good minima, and that both normalization with trained and fixed scale and shift parameters works similarly well.

B Proof of Theorem 1

Our proof relies on diagonalizing the circulant matrices implementing the convolutions with the Fourier transform.

B.1 Linear single channel CNNs in the Fourier domain

Since the matrices $\mathbf{W}_i$ are circulant, they can be diagonalized with the Fourier transformation. As a consequence, the loss function of a linear CNN becomes a sum of loss functions of one dimensional single channel deep linear neural networks.

With $\mathbf{W}_i = \mathbf{F}\text{diag}(\sqrt{n}\mathbf{F}^H\mathbf{w}_i)\mathbf{F}^H$, where $\mathbf{F}$ is the unitary $n \times n$ discrete Fourier transform matrix and $\mathbf{w}_i$ is the first column of $\mathbf{W}_i$, the network's output without normalization can be expressed as

$$f(\mathbf{x}, \mathbf{W}) = \prod_{i=1}^{d} \mathbf{W}_i \mathbf{x} = \prod_{i=1}^{d} \mathbf{F}\text{diag}(\sqrt{n}\mathbf{F}^H\mathbf{w}_i)\mathbf{F}^H\mathbf{x} = n^{d/2}\mathbf{F}\left(\prod_{i=1}^{d}\text{diag}(\mathbf{F}^H\mathbf{w}_i)\right)\mathbf{F}^H\mathbf{x}.$$

With this expression, the loss function becomes

$$
\begin{aligned}
L(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \frac{1}{2}\left\|\mathbf{y} - f(\mathbf{x}, \mathbf{W})\right\|^2 &= \frac{1}{2}\left\|\mathbf{y} - n^{d/2}\mathbf{F}\left(\prod_{i=1}^{d}\operatorname{diag}(\mathbf{F}^H\mathbf{w}_i)\right)\mathbf{F}^H\mathbf{x}\right\|^2 \\
&= \frac{1}{2}\left\|\mathbf{F}^H\mathbf{y} - n^{d/2}\left(\prod_{i=1}^{d}\operatorname{diag}(\mathbf{F}^H\mathbf{w}_i)\right)\mathbf{F}^H\mathbf{x}\right\|^2 \\
&= \frac{1}{2}\sum_{j=1}^{n}\left|\mathbf{f}_j^H\mathbf{y} - n^{d/2}\left(\prod_{i=1}^{d}\mathbf{f}_j^H\mathbf{w}_i\right)\mathbf{f}_j^H\mathbf{x}\right|^2,
\end{aligned}
$$

where $\mathbf{f}_j$ is the j-th column of the Fourier matrix $\mathbf{F}$.

B.2 Proof of Theorem 1

In this section, we show that gradient descent takes exponentially many steps to converge under Xavier initialization, a standard initialization scheme. The proof follows a similar line of arguments as a very related result by Shamir [Sha18] on the hardness of optimizing one-dimensional deep neural networks.

Assume the kernel size of the i-th convolution layer is $p \geq 1$. Thus, $\mathbf{w}_i \in \mathbb{S}_p$, where $\mathbb{S}_p = \{\mathbf{z} = (z_1, z_2, \ldots, z_n) | z_1, \ldots, z_p \in \mathbb{R}, z_{p+1}, \ldots, z_n = 0\}$.

Assumption 1. *Assume $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ are drawn independently and the first p entries of $\mathbf{w}_i$ are drawn i.i.d from a distribution that satisfies*

$$
\begin{aligned}
&\mathrm{P}\left[\|\mathbf{w}_i\| \leq t\right] \leq c_1 t \quad \textit{and} \\
&\mathbb{E}\left[\|\mathbf{w}_i\|\right] \leq \frac{1}{\sqrt{n}}(1 - c_2),
\end{aligned}
$$

where the constants $c_1, c_2 > 0$ are independent of d.

The assumption holds for some widely used initialization distributions, like the distribution $\mathcal{N}(0, \frac{1}{n^2})$ ($\mathbb{E}\left[\|\mathbf{w}_i\|\right] = \frac{\sqrt{2}}{n}\frac{\Gamma(n+1/2)}{\Gamma(n/2)} \leq \frac{1}{\sqrt{n}}$) in the statement of the theorem.

Next, we show that with the initialization satisfying Assumption 1, gradient descent takes at least exponentially many gradient descent iterations to reach a close-to-optimal point with high probability.

The key idea of the proof of Theorem 1 is to show that if we start from such a random initialization $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d)$, then gradient descent must take exponentially many steps to escape a ball of radius r around $\mathbf{w}$, defined as

$$
\mathcal{B}(\mathbf{w}, r) = \left\{\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d) \middle| \sum_{i=1}^{d}\|\mathbf{v}_i - \mathbf{w}_i\|^2 \leq r^2\right\}.
$$

We then show that the loss function value evaluated at any point inside the ball is sub-optimal, i.e., for $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$, $L(\mathbf{v}) \geq c'$, where c' is a constant. This will establish the proof.

To show that there exist a radius $r > 0$ such that gradient descent takes at least exponentially many steps to escape the ball $\mathcal{B}(\mathbf{w}, r)$, we first note that the number of iterations required to escape

a ball of radius r is at least $r/(\eta \sup_{\mathbf{v}\in\mathcal{B}(\mathbf{w},r)} \left\|\nabla L(\mathbf{v})\right\|)$, since at each step, gradient descent can at most move by $\eta \sup_{\mathbf{v}\in\mathcal{B}(\mathbf{w},r)} \left\|\nabla L(\mathbf{v})\right\|$ away from the initialization.

The following lemma provides an upper bound on $\left\|\nabla L(\mathbf{v})\right\|$, which enables us to show that the number of iterations must be large.

Lemma 1. *Suppose that the initial point* $\mathbf{w}$ *satisfies, for some* $\alpha, \delta > 0$, *i)* $\max_k \left(\prod_{i\neq k} \|\mathbf{w}_i\|\right) \leq \alpha$ *and ii)* $\min_i \|\mathbf{w}_i\| \geq \delta$. *Then, there exists a radius* r *such that for all* $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$, *it holds that* $\prod_{i=1}^d \|\mathbf{v}_i\| \leq \alpha \exp\left(\frac{\sqrt{d}r}{\delta}\right) \max_k \|\mathbf{v}_k\|$ *and* $\left\|\nabla L(\mathbf{v})\right\| \leq \left\|D(\mathbf{v})\right\| \|\mathbf{x}\| \alpha\sqrt{d} \exp\left(\frac{\sqrt{d}}{\delta} r\right)$, *where* $L(\mathbf{v}) = \frac{1}{2}\left\|D(\mathbf{v})\right\|^2$ *and* $D(\mathbf{v}) = \mathbf{F}^H\mathbf{y} - n^{d/2} \prod_{i=1}^d \operatorname{diag}(\mathbf{F}^H\mathbf{v}_i)\mathbf{F}^H\mathbf{x}$.

Lemma 1 guarantees that in a ball $\mathcal{B}(\mathbf{w}, r)$ around the initialization, the gradients of the loss function are strictly upper bounded. So, provided the stepsize is not too large, the progress made in each step of gradient descent is also upper bounded. Then, we show with high probability, that there is a radius r that is much larger than the updates in each step.

Evoking Lemma 1, given a constant stepsize η, the number of steps required to escape the ball $\mathcal{B}(\mathbf{w}, r)$ is at least $\frac{r}{\Omega(\eta n^{d/2}\alpha\sqrt{d}\exp(\frac{\sqrt{d}}{\delta}r))}$ if $\left\|D(\mathbf{v})\right\| \|\mathbf{x}\|$ is upper bounded by a numerical constant. Next, we show that conditions i and ii from Lemma 1 hold with high probability for $\alpha = \frac{\exp(-2cd)}{n^{d/2}}$ and $\delta = \exp(-cd)$ given Assumption 1. We then show that there is a radius r in which $\left\|D(\mathbf{v})\right\|$ is upper bounded and gradient descent takes at least exponentially many steps to escape the ball $\mathcal{B}(\mathbf{w}, r)$.

We start by showing that conditions i and ii from Lemma 1 hold with high probability for $\alpha = \frac{\exp(-2cd)}{n^{d/2}}$ and $\delta = \exp(-cd)$.

Lemma 2. *Suppose* $\mathbf{w}$ *is initialized satisfying Assumption 1. With probability at least* $1-\Omega(de^{-cd})$, *the conditions*

$$i) \max_k \prod_{i\neq k} \|\mathbf{w}_i\| \leq \frac{\exp(-2cd)}{n^{d/2}},\ ii) \min_i \|\mathbf{w}_i\| \geq \exp(-cd),\ \text{and}\ iii) \max_i \|\mathbf{w}_i\| \leq \exp(cd)$$

hold simultaneously, where c *is a constant independent of* d.

Proof. With Markov's inequality and Assumption 1, we have, for $t > 0$, that

$$\mathrm{P}\left[\prod_{i\neq k} \|\mathbf{w}_i\| \geq t\right] \leq \frac{\prod_{i\neq k} \mathbb{E}\left[\|\mathbf{w}_i\|\right]}{t} \leq \frac{((1-c_2)/\sqrt{n})^{d-1}}{t}.$$

Let c be a constant so that $\exp(-4c) = 1 - c_2$ and set $t = \frac{\exp(-2cd)}{n^{d/2}}$. Then we obtain

$$\mathrm{P}\left[\prod_{i\neq k} \|\mathbf{w}_i\| \geq \frac{\exp(-2cd)}{n^{d/2}}\right] \leq \frac{\sqrt{n}\exp(-4c(d-1))}{\exp(-2cd)} = \Omega(\exp(-2cd)). \tag{1}$$

Thus, by the union bound, $\max_k \prod_{i\neq k} \|\mathbf{w}_i\| \leq \frac{\exp(-2cd)}{n^{d/2}}$ with probability at least $1-\Omega(d\exp(-2cd))$.

We next consider $\min_i \|\mathbf{w}_i\|$. Again by Assumption 1, it holds for all i that $\mathrm{P}\left[\|\mathbf{w}_i\| \leq \exp(-cd)\right] \leq \Omega(\exp(-cd))$. Then, by the union bound,

$$\mathrm{P}\left[\min_i \|\mathbf{w}_i\| \leq \exp(-cd)\right] \leq \sum_{i=1}^d \mathrm{P}\left[\|\mathbf{w}_i\| \leq \exp(-cd)\right] \leq \Omega(d\exp(-cd)). \tag{2}$$

Finally, by the union bound, Markov's inequality, and Assumption 1, we have that

$$\mathrm{P}\left[\max_i \|\mathbf{w}_i\| \geq \exp(cd)\right] \leq \sum_{i=1}^{d} \mathrm{P}\left[\|\mathbf{w}_i\| \geq \exp(cd)\right] \leq \Omega(d\exp(-cd)). \tag{3}$$

Thus, by a union bound, with probability at least

$$1 - \Omega(\exp(-2cd)) - \Omega(d\exp(-cd)) - \Omega(d\exp(-cd)) = 1 - \Omega(d\exp(-cd)),$$

the conditions i-iii are satisfied simultaneously, which concludes the proof. □

Next, we show that under conditions i-iii in Lemma 2, provided that d is large enough, $\left\|D(\mathbf{v})\right\|$ is upper bounded by a constant independent of d for radius $r = \Omega(\frac{\delta}{\sqrt{d}}) = \Omega(\frac{\exp(-cd)}{\sqrt{d}})$.

Evoking Lemma 1, we have

$$\begin{aligned}
\prod_{i=1}^{d} \|\mathbf{v}_i\| &\leq \alpha \exp\left(\frac{\sqrt{d}r}{\delta}\right) \max_k \|\mathbf{v}_k\| \\
&\leq \frac{\exp(-2cd)}{n^{d/2}} \Omega(\exp(cd)) \max_k \|\mathbf{v}_k\| \\
&\leq \frac{\Omega(1)}{n^{d/2}},
\end{aligned}$$

where the second inequality follows from our choice of α and $\exp(\frac{\sqrt{d}r}{\delta}) = \Omega(1)$, by our choice of δ, and the third inequality follows from $\max_k \|\mathbf{v}_k\| \leq \max_k \|\mathbf{w}_k\| + r = \Omega(\exp(cd))$.

Then, we can upper bound $\left\|D(\mathbf{v})\right\|$ when d is large,

$$\begin{aligned}
\left\|D(\mathbf{v})\right\| &= \left\|\mathbf{F}^H\mathbf{y} - n^{d/2}\prod_{i=1}^{d}\mathrm{diag}(\mathbf{F}^H\mathbf{v}_i)\mathbf{F}^H\mathbf{x}\right\| \leq \sum_{j=1}^{n}\left(\left|\mathbf{f}_j^H\mathbf{y}\right| + n^{d/2}\prod_{i=1}^{d}\left|\mathbf{f}_j^H\mathbf{v}_i\right|\left|\mathbf{f}_j^H\mathbf{x}\right|\right) \\
&\leq n\left(\|\mathbf{y}\| + n^{d/2}\prod_{i=1}^{d}\|\mathbf{v}_i\|\|\mathbf{x}\|\right) \leq n\left(\|\mathbf{y}\| + n^{d/2}\|\mathbf{x}\|\right).
\end{aligned}$$

Therefore, $\left\|D(\mathbf{v})\right\|$ is upper bounded by a constant (given $\mathbf{y}$ is constant, and $\mathbf{x}$ satisfies the assumption in theorem 1 independent of d.

Finally, we can prove that gradient descent takes at least exponentially many steps to escape $\mathcal{B}(\mathbf{w}, r)$ with $r = \Omega(\frac{\exp(-cd)}{\sqrt{d}})$. By lemma 2, with conditions i-iii satisfied, the number of steps in $\mathcal{B}(\mathbf{w}, r)$ is at least $\frac{r}{\left\|D(\mathbf{v})\right\|\|\mathbf{x}\|\Omega(\eta n^{d/2}\alpha\sqrt{d}\exp(\frac{\sqrt{d}}{\delta}r))}$. Since $\left\|D(\mathbf{v})\right\|$ can be upper bounded by a constant, and using that $\|\mathbf{x}\|$ is upper bounded by a constant (from $\|\mathbf{x}\| < \|\mathbf{y}\|$, with $\|\mathbf{y}\|$ upper bounded by a constant) and $\eta \leq \exp(\frac{d}{2})$ (by the assumptions in theorem 1), $\frac{r}{\left\|D(\mathbf{v})\right\|\|\mathbf{x}\|\Omega(\eta n^{d/2}\alpha\sqrt{d}\exp(\frac{\sqrt{d}}{\delta}r))} \geq \frac{r}{\Omega(\exp(-\frac{3}{2}cd + \frac{\sqrt{d}}{\delta}r))} = \Omega(\exp(\frac{1}{2}cd))$, which increases exponentially in d.

It remains to prove that for any $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$ (with r as chosen above), the loss function $L(\mathbf{v})$ is lower bounded away from the global minimum. We have proved that when d is large, $\prod_{i=1}^{d}\left|\mathbf{f}_j^H\mathbf{v}_i\right| \leq \prod_{i=1}^{d}\|\mathbf{v}_i\| \leq \Omega(1)$. From the assumption $\|\mathbf{y}\| - \|\mathbf{x}\|dn^{d/2} \geq \tau$, there exists $t \in [n]$

such that $\left|\mathbf{f}_t^H\mathbf{y}\right| - n^{d/2}\left|\mathbf{f}_t^H\mathbf{x}\right| \geq \tau$. So, we have

$$\begin{aligned}\left|\mathbf{f}_t^H\mathbf{y} - n^{d/2}\prod_{i=1}^{d}\left(\mathbf{f}_t^H\mathbf{v}_i\right)\mathbf{f}_t^H\mathbf{x}\right| &\geq \left|\mathbf{f}_t^H\mathbf{y}\right| - \left|n^{d/2}\prod_{i=1}^{d}\left(\mathbf{f}_t^H\mathbf{v}_i\right)\mathbf{f}_t^H\mathbf{x}\right| \\ &\geq \left|\mathbf{f}_t^H\mathbf{y}\right| - n^{d/2}\prod_{i=1}^{d}\left|\mathbf{f}_t^H\mathbf{v}_i\right|\left|\mathbf{f}_t^H\mathbf{x}\right| \\ &\geq \left|\mathbf{f}_t^H\mathbf{y}\right| - n^{d/2}\left|\mathbf{f}_t^H\mathbf{x}\right| > \tau.\end{aligned}$$

Thus, for $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$, the loss function obeys $L(\mathbf{v}) = \sum_{j=1}^{n}\left|\mathbf{f}_j^H\mathbf{y} - n^{d/2}\prod_{i=1}^{d}\left(\mathbf{f}_j^H\mathbf{v}_i\right)\mathbf{f}_j^H\mathbf{x}\right| > \tau$ and is thus lower bounded away from zero-training error.

B.3 Proof of Lemma 1

Let $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$ as defined previously. We have

$$\begin{aligned}\left\|\nabla L(\mathbf{v})\right\|^2 &\leq \sum_{k=1}^{d}\left\|\nabla_{\mathbf{v}_k} L(\mathbf{v})\right\|^2 = \sum_{k=1}^{d} n^d \left\|D(\mathbf{v})\right\|^2 \left(\prod_{i\neq k}\|\mathbf{v}_i\|\right)^2 \|\mathbf{x}\|^2 \\ &\leq \sum_{k=1}^{d} n^d \left\|D(\mathbf{v})\right\|^2 \left(\max_k \prod_{i\neq k}\|\mathbf{v}_i\|\right)^2 \|\mathbf{x}\|^2. \qquad (4)\end{aligned}$$

Here, we used that

$$\left\|\nabla_{\mathbf{v}_k} L(\mathbf{v})\right\| = \left\|\left\|D(\mathbf{v})\right\| n^{d/2}\mathrm{diag}(\mathbf{F}^H\mathbf{x}_i)\prod_{i\neq k}\mathrm{diag}(\mathbf{F}^H\mathbf{v}_i)\right\| \leq n^{d/2}\left\|D(\mathbf{v})\right\|\|\mathbf{x}\|\prod_{i\neq k}\|\mathbf{v}_i\|.$$

The lemma now follows from $\max_k\left(\prod_{i\neq k}\|\mathbf{v}_i\|\right) \leq \left(\alpha\sqrt{d}\right)\exp\left(\frac{\sqrt{d}}{\delta}r\right)$.

Define $\mathbf{r}_i = \mathbf{v}_i - \mathbf{w}_i$, for notational convenience, and note that $\sum_{i=1}^{d}\|\mathbf{r}_i\|^2 \leq r^2$ since $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$. Then, we have

$$\begin{aligned}\prod_{i\neq k}\|\mathbf{v}_i\| &= \prod_{i\neq k}\|\mathbf{w}_i + \mathbf{r}_i\| \\ &\leq \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\left[\prod_{i\neq k}\left(1 + \frac{\|\mathbf{r}_i\|}{\|\mathbf{w}_i\|}\right)\right] \\ &\leq \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\left[\prod_{i\neq k}\left(1 + \frac{\|\mathbf{r}_i\|}{\delta}\right)\right] \\ &= \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\exp\left[\sum_{i\neq k}\log\left(1 + \frac{\|\mathbf{r}_i\|}{\delta}\right)\right] \leq \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\exp\left[\sum_{i\neq k}\frac{\|\mathbf{r}_i\|}{\delta}\right], \qquad (5)\end{aligned}$$

where the second inequality follows from the assumption $\min_i \|\mathbf{w}_i\| \geq \delta$. We also have

$$r^2 \geq \sum_{i=1}^{d} \|\mathbf{r}_i\|^2 \geq \frac{\left(\sum_{i=1}^{d} \|\mathbf{r}_i\|_1\right)^2}{d}. \tag{6}$$

Inserting equation (6) into equation (5), we get

$$\prod_{i\neq k} \|\mathbf{v}_i\| \leq \left(\prod_{i\neq k} \|\mathbf{w}_i\|\right) \exp\left(\frac{\sqrt{d}r}{\delta}\right) \leq \alpha \exp\left(\frac{\sqrt{d}r}{\delta}\right),$$

where the last inequality follows from the assumption $\max_k \left(\prod_{i\neq k} \|\mathbf{w}_i\|\right) \leq \alpha$. Therefore, $\prod_{i=1}^{d} \|\mathbf{v}_i\| \leq \alpha \exp\left(\frac{\sqrt{d}r}{\delta}\right) \cdot \max_k \|\mathbf{v}_k\|$. Application of this inequality in (4) yields $\left\|\nabla L(\mathbf{v})\right\| \leq \sqrt{d} \cdot \max_k \left\|\nabla_k L(\mathbf{v})\right\| \leq n^{d/2} \left\|D(\mathbf{v})\right\| \|\mathbf{x}\| \left(\alpha\sqrt{d}\right) \exp\left(\frac{\sqrt{d}}{\delta} r\right)$, which concludes the proof.

C Channel normalization on a single channel linear CNN

We consider channel normalization with scale parameter fixed to $\gamma_{ij} = 1$ and the shift parameter fixed to $\beta_{ij} = 0$. Hence, merely varying the scale of the parameters does not change the output of the network with channel normalization $f_N(\mathbf{W}, \mathbf{x})$.

To simplify the derivation we assume the input vector $\mathbf{x}$ is centered, i.e., its entries sum to zero. Let $\mathbf{z}_i$ be the input of the i-th layer. Then, the output of the convolutional layer can be written as $\mathbf{W}_i \mathbf{z}_i$, where $\mathbf{W}_i$ is the circulant matrix implementing the convolution operation. Channel normalization centers the mean and adjusts the empirical variance to one. Since $\mathbf{W}_i$ is a convolution operation, given $\mathbf{z}_i$ is centered, $\mathbf{W}_i \mathbf{z}_i$ is centered as well. Thus, the effect of channel normalization in this setup is to normalize the scale of the vector.

It follows that the output of the network with channel normalization can be written as

$$f_N(\mathbf{W}, \mathbf{x}) = w_{d+1} \frac{\prod_{i=1}^{d} \mathbf{W}_i \mathbf{x}}{\left\|\prod_{i=1}^{d} \mathbf{W}_i \mathbf{x}\right\|}, \tag{7}$$

where w_{d+1} is a scale parameter that we introduced so that $f_N(\mathbf{W}, \mathbf{x})$ can exhaust $\mathbb{R}^n$.

To see that the output of the network can be written as in equation (7), note that the input of the first layer is $\mathbf{z}_1 = \mathbf{x}$, with $\mathbf{x}$ centered. The normalization of the first layer yields the input of the second layer by division by $\|\mathbf{W}_1 \mathbf{z}_1\|/\sqrt{n} = \|\mathbf{W}_1 \mathbf{x}\|/\sqrt{n}$, which gives

$$\mathbf{z}_2 = \frac{\mathbf{W}_1 \mathbf{x}}{\|\mathbf{W}_1 \mathbf{x}\|/\sqrt{n}}.$$

The normalization operation at the second layer divides by

$$\|\mathbf{W}_2 \mathbf{z}_2\|/\sqrt{n} = \left\|\frac{\sqrt{n}\mathbf{W}_2 \mathbf{W}_1 \mathbf{x}}{\|\mathbf{W}_1 \mathbf{x}\|}\right\|/\sqrt{n} = \frac{\|\mathbf{W}_2 \mathbf{W}_1 \mathbf{x}\|}{\|\mathbf{W}_1 \mathbf{x}\|},$$

which yields

$$\mathbf{z}_3 = \frac{\mathbf{W}_3 \mathbf{z}_2}{\|\mathbf{W}_2 \mathbf{W}_1 \mathbf{x}\|/\|\mathbf{W}_1 \mathbf{x}\|} = \frac{\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 \mathbf{x}}{\|\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 \mathbf{x}\|/\sqrt{n}}.$$

Continuing this logic yields equation (7), where we absorbed $\sqrt{n}$ in the parameter w_{d+1}.

C.1 Gradient analysis

Note that with $\mathbf{X} \in \mathbb{R}^{n \times n}$ the circulant matrix with first column equal to $\mathbf{x}$, we have that $\mathbf{W}_k \mathbf{x} = \mathbf{X}\mathbf{w}_k$ and the channel normalized output of the network becomes

$$f_N(\mathbf{W}, \mathbf{x}) = w_{d+1} \frac{\mathbf{X}_k \mathbf{w}_k}{\|\mathbf{X}_k \mathbf{w}_k\|},$$

where we defined $\mathbf{X}_k = \prod_{i \neq k} \mathbf{W}_i \mathbf{X}$ for notational convenience. We next compute the gradient

$$\nabla_{\mathbf{w}_k} L(\mathbf{W}, \mathbf{x}, \mathbf{y}), \quad L(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \left\|\mathbf{y} - f_N(\mathbf{W}, \mathbf{x})\right\|^2.$$

Towards this goal, first note that

$$\nabla_{\mathbf{z}} \left\| \mathbf{y} - \gamma \frac{\mathbf{z}}{\|\mathbf{z}\|} \right\|^2 = \frac{\gamma}{\|\mathbf{z}\|} \left(\mathbf{I} - \mathbf{z}\mathbf{z}^T / \|\mathbf{z}\|^2 \right) (\mathbf{y} - \gamma \mathbf{z}).$$

Thus, by the chain rule

$$\begin{aligned} \nabla_{\mathbf{w}_k} L(\mathbf{W}, \mathbf{x}, \mathbf{y}) &= w_{d+1} \mathbf{X}_k^T \frac{1}{\|\mathbf{X}_k \mathbf{w}_k\|} \left(\mathbf{I} - \mathbf{X}_k \mathbf{w}_k \mathbf{w}_k^T \mathbf{X}_k^T / \|\mathbf{X}_k \mathbf{w}_k\|^2 \right) (\mathbf{y} - w_{d+1} \mathbf{X}_k \mathbf{w}_k) \\ &= \mathbf{X}_k^T \frac{w_{d+1}}{\|\mathbf{X}_k \mathbf{w}_k\|} \left(\mathbf{I} - \mathbf{X}_k \mathbf{w}_k \mathbf{w}_k^T \mathbf{X}_k^T / \|\mathbf{X}_k \mathbf{w}_k\|^2 \right) \mathbf{y}. \end{aligned}$$

From direct computation,

$$\left\langle \nabla_{\mathbf{w}_k} L(\mathbf{W}, \mathbf{x}, \mathbf{y}), \mathbf{w}_k \right\rangle = 0,$$

thus the gradient is orthogonal to $\mathbf{w}_k$. Moreover, if $\mathbf{w}_k$ is multiplied with a positive scalar γ, then the gradient scales with $1/\gamma$.