Batch Normalization Orthogonalizes Representations in Deep Random Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper underlines a subtle property of batch-normalization (BN): Successive batch normalizations with random linear updates make samples increasingly orthogonal. We establish a non-asymptotic characterization of the interplay between depth, width, and the orthogonality of deep representations. More precisely, under a mild assumption, we prove that the deviation of the representations from orthogonality rapidly decays with depth up to a term inversely proportional to the network width. This result has two main implications: 1) Theoretically, as the depth grows, the distribution of the outputs contracts to a Wasserstein-2 ball around an isotropic Gaussian distribution. Furthermore, the radius of this Wasserstein ball shrinks with the width of the network. 2) Practically, the orthogonality of the representations directly influences the performance of stochastic gradient descent (SGD). When representations are initially aligned, we observe SGD wastes many iterations to orthogonalize representations before the classification. Nevertheless, we experimentally show that starting optimization from orthogonal representations is sufficient to accelerate SGD, with no need for BN.

1 Introduction

Batch Normalization (BN) (Ioffe and Szegedy, 2015) enhances training across a wide range of deep network architectures and experimental setups (He et al., 2016; Huang et al., 2017; Silver et al., 2017). The practical success of BN has inspired research into the underlying benefits of BN (Santurkar et al., 2018; Karakida et al., 2019; Arora et al., 2018b; Bjorck et al., 2018). All these studies investigated the connection between BN and gradient-based optimization. However, the benefits of BN go beyond its effect in optimization. For example, Frankle et al. (2020) shows that BN networks with random weights also achieve surprisingly high performance after only minor adjustments of weights. This striking result motivates us to study the representational power of random networks with BN.

Consider a batch of samples passing through consecutive BN and linear layers with Gaussian weights. These samples are perturbed by each random linear update, followed by a non-linear transformation with BN. While the representations appear unpredictable at first glance, they become incrementally orthogonal through these transformations. We introduce the notion of "orthogonality gap", defined in Section 4, to quantify the deviation of representations from a perfectly orthogonal representation. We prove that the orthogonality gap decays exponentially with the network depth and stabilizes around a term inversely related to the network width. More precisely

$$\mathbb{E}\left[\text{orthogonality gap}\right] = \mathcal{O}\left((1-\alpha)^{\text{depth}} + \frac{\text{batch size}}{\alpha\sqrt{\text{width}}}\right),$$

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

where the expectation is taken over the random weights of the network, and $\alpha > 0$ is an absolute constant under a mild assumption. In probability theoretic terms, we prove stochastic stability of the Markov chain of hidden representations (Kushner, 1967; Kushner and Yin, 2003; Khasminskii, 2011). The orthogonality of deep representations allows us to prove that the distribution of the representations contracts to a Wasserstein-2 ball around isotropic Gaussian distribution as the network depth grows. Moreover, the radius of the ball is inversely proportional to the network width. Omitting details, we prove the following bound holds:

$$\text{Wasserstein}_2(\text{representations}, \text{Gaussian})^2 = \mathcal{O}\left((1-\alpha)^{\text{depth}}\,(\text{batch size}) + \frac{\text{batch size}}{\alpha\sqrt{\text{width}}}\right).$$

The above equation shows how depth, width, and batch size, interact with the law of the representations. Since the established rate is exponential with depth, the distribution of the representations stays in a Wasserstein ball around isotropic Gaussian distribution after a few layers. Thus, BN not only stabilizes the distribution of the representations, which is its main promise (Ioffe and Szegedy, 2015), but also enforces Gaussian isotropic distribution in deep layers.

While deep neural networks are linked to Gaussian processes in the *asymptotic* regime of infinite width (Matthews et al., 2018), to the best of our knowledge, this is the first *non-asymptotic* Gaussian approximation for the representations of deep neural networks. Many studies rely on deep Gaussian representations in an infinite width setting (Yang et al., 2019; Schoenholz et al., 2016; Pennington et al., 2018; Klambauer et al., 2017; De Palma et al., 2018). Therefore, our error bound can be incorporated into their analysis to extend these results to the regime of finite width.

Since optimization starts from networks with random weights, properties of random neural networks directly influence optimization. This impact motivated theoretical studies to explore properties of random deep neural networks (Daneshmand et al., 2020; Bjorck et al., 2018; Frankle et al., 2020; Schoenholz et al., 2016; Saxe et al., 2013; Bahri et al., 2020). In particular, it is shown that hidden representations in random networks *without BN* become correlated as the network grows in depth, thereby drastically slowing optimization (Daneshmand et al., 2020; He et al., 2016; Bjorck et al., 2018; Saxe et al., 2013). On the contrary, we prove that deep representations in networks *with BN* are almost orthogonal.

We experimentally validate that orthogonal representations can save training time that would otherwise be needed to orthogonalize them. We propose an initialization for weights that ensures the orthogonality of hidden representations. Our experiments show such an initialization effectively avoids the training slowdown with depth for vanilla networks, with no need for BN. This observation further motivates studying the inner workings of BN to replace or improve it in deep neural networks.

Theoretically, we made the following contributions:

1. For MLPs with batch normalization, linear activation, and Gaussian i.i.d. weights, we prove that representations across layers become increasingly orthogonal up to a constant inversely proportional to the network width.
2. Leveraging the orthogonality, we prove that distributions of outputs of networks contract to a Wasserstein ball around a Gaussian distribution as the depth grows. Up to the best of our knowledge, this the first *non-asymptotic* Gaussian approximation for outputs of deep neural networks with finite width.

Experimentally, we made the following contribution:

3. We propose a novel weight initialization for MLPs with ReLU activations that ensure orthogonal representations without BN. Experimentally, we show that this initialization effectively avoids training slowdown with depth in the absence of BN.

2 Related work

There is growing interest in bridging the gap between neural networks, as the most successful parametric methods for learning, and Gaussian processes and kernel methods, as well-understood classical models for learning (Jacot et al., 2018; Matthews et al., 2018; Lee et al., 2017; Bietti and Mairal,

2019; Huang et al., 2014). In particular, a single-layer network resembles a Gaussian process as its width goes to infinity (Neal, 1996). However, increasing the depth may significantly shift the distribution of the representations away from Gaussian (Ioffe and Szegedy, 2015). This distributional shift clouds the connection between Gaussian processes and deep neural networks. Matthews et al. (2018) show increasing the width of the network proportional to the network depth ensures Gaussian outputs in an infinite width regime. Here, we prove BN allows us to approximate the distribution of the network outputs by isotopic Gaussian distribution for networks with finite width.

Recent studies report that BN accelerates optimization by avoiding the rank collapse in deep representation (Daneshmand et al., 2020), direction-length decoupling of optimization (Kohler et al., 2018), influencing the learning rate of the steepest descent (Arora et al., 2018b; Bjorck et al., 2018), and smoothing the optimization objective function (Santurkar et al., 2018; Karakida et al., 2019). Frankle et al. (2020) show that deep BN networks with only minor adjustments to the initial random weights can classify images surprisingly well. This interesting result motivates us to investigate hidden representations in deep random BN networks.

3 Preliminaries

3.1 Notations

Borrowing the setting of Daneshmand et al. (2020), we focus on a Multi-Layer Perceptron (MLP) with linear activation. Theoretical studies of linear networks is a growing research area (Saxe et al., 2013; Daneshmand et al., 2020; Bartlett et al., 2019; Arora et al., 2018a). When weights initialized randomly, linear and non-linear networks share similar properties (Daneshmand et al., 2020). We use n to denote batch size, and d to denote the width across all layers, which we further assume is larger than the batch size $d \geq n$. Let $H_\ell \in \mathbb{R}^{d\times n}$ denote n sample representations in layer ℓ, with $H_0 \in \mathbb{R}^{d\times n}$ corresponding to n input samples in the batch with d features. Successive representations are connected by Gaussian weight matrices $W_\ell \sim \mathcal{N}(0, I_d/d)$ as

$$H_{\ell+1} = \frac{1}{\sqrt{d}} BN(W_\ell H_\ell), \quad BN(M) = \text{diag}(MM^\top)^{-1/2} M, \tag{1}$$

where $\text{diag}(M)$ zeros out off-diagonal elements of its input matrix, and the scaling factor $1/\sqrt{d}$ ensures that all matrices $\{H_k\}_k$ have unit Frobenius norm (see Appendix 3). The BN function in Eq. (1) differs slightly from the commonly used definition for BN as the mean correction is omitted. However, Daneshmand et al. (2020) showes this difference does not change the network properties qualitatively. Readers can find all the notations in Appendix Table 2.

3.2 The linear independence of hidden representations

Daneshmand et al. (2020) observe that if inputs are linearly independent, then their hidden representations remain linearly independent in all layers as long as $d = \Omega(n^2)$. Under technical assumptions, Daneshmand et al. (2020) establishes a lower-bound on the average of the rank of hidden representations over infinite layers. Based on this study, we assume that the linear independence holds and build our analysis upon that. This avoids restating the technical assumptions of Daneshmand et al. (2020) and also further technical refinements of their theorems. The next assumption presents the formal statement of the linear independence property.

Assumption $\mathcal{A}_1(\alpha, \ell)$**.** *There exists an absolute positive constant α such that the minimum singular value of H_k is greater than (or equal to) α for all $k = 1, \ldots, \ell$.*

The linear independence of the representations is a shared property across all layers. However, the representations constantly change when passing through random layers. In this paper, we mathematically characterize the dynamics of the representations across layers.

4 Orthogonality of deep representations

4.1 A warm-up observation

To illustrate the difference between BN and vanilla networks, we compare hidden representations of two input samples across the layers of these networks. Figure 1 plots the absolute value of cosine

similarity of these samples across layers. This plot shows a stark contrast between vanilla and BN networks: While representations become increasingly orthogonal across layers of a BN network, they become increasingly aligned in a vanilla network. We used highly correlated inputs for the BN network and perfectly orthogonal inputs for the Vanilla network to contrast their differences. While the hidden representation of vanilla networks has been theoretically studied (Daneshmand et al., 2020; Bjorck et al., 2018; Saxe et al., 2013), up to the best of our knowledge, there is no theoretical analysis of BN networks. In the following section, we formalize and prove this orthogonalizing property for BN networks.

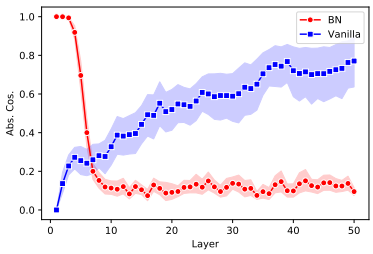


Figure 1: **Orthogonality: BN vs. vanilla networks.** The horizontal axis shows the number of layers, and the vertical axis shows the absolute value of cosine similarity between two samples across the layers ($d = 32$). Mean and 95% confidence intervals of 20 independent runs.

4.2 Theoretical analysis

The notion of orthogonality gap plays a central role in our analysis. Given the hidden representation $H \in \mathbb{R}^{d\times n}$, matrix $H^\top H$ constitutes inner products between representations of different samples. Note that $H^\top H \in \mathbb{R}^{n\times n}$ is different form the covariance matrix $HH^\top/n \in \mathbb{R}^{d\times d}$. The orthogonality gap of H is defined as the deviations of $H^\top H$ from identity matrix, after proper scaling. More precisely, define $V : \mathbb{R}^{d\times n} \setminus \mathbf{0} \to \mathbb{R}_+$ as

$$V(H) := \left\| \left(\frac{1}{\|H\|_F^2}\right) H^\top H - \left(\frac{1}{\|I_n\|_F^2}\right) I_n \right\|_F. \tag{2}$$

The following theorem establishes a bound on the orthogonality of representation in layer ℓ.

Theorem 1. *Under Assumption $\mathcal{A}_1(\alpha, \ell)$, the following holds:*

$$\mathbb{E}\left[V(H_{\ell+1})\right] \le 2\left(1 - \frac{2}{3}\alpha\right)^\ell + \frac{3n}{\alpha\sqrt{d}}. \tag{3}$$

Assumption $\mathcal{A}_1$ is studied by Daneshmand et al. (2020) who note that $\mathcal{A}_1(\alpha, \infty)$ holds as long as $d = \Omega(n^2)$. If $\mathcal{A}_1$ does not hold, one can still prove that there is a function of representations that decays with depth up to a constant (see Appendix C).

The above result implies that BN is an approximation algorithm for orthogonalizing the hidden representations. If we replace $\mathrm{diag}(M)^{-1}$ by $(M)^{-1}$ in BN formula, in Eq. (1), then all the hidden representation will become exactly orthogonal. However, computing the inverse of a *non-diagonal* $d \times d$ matrix is computationally expensive, which must repeat for all layers throughout training, and differentiation must propagate back through this inversion. The diagonal approximation in BN significantly reduces the computational complexity of the matrix inversion. Since the orthogonality gap decays at an exponential rate with depth, the approximate orthogonality is met after only a few layers. Interestingly, this yields a desirable cost-accuracy trade-off: For a larger width, the orthogonality is more accurate, due to term $1/\sqrt{d}$ in the orthogonality gap, and also the computational gain is more significant.

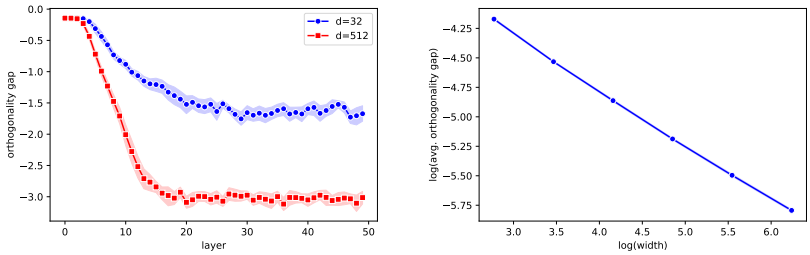
From a different angle, Theorem 1 proves the stochastic stability of the Markov chain of hidden representations. In expectation, stochastic processes may obey an inherent Lyapunov-type of stability

(Kushner, 1967; Kushner and Yin, 2003; Khasminskii, 2011). One can analyze the mixing and hitting times of Markov chains based on the stochastic stability of Markov chains,(Kemeny and Snell, 1976; Eberle, 2009). In our analysis, the orthogonality gap is a Lyapunov function characterizing the stability of the chain of hidden representations. This stability opens the door to more theoretical analysis of this chain, such as studying mixing and hitting times. Since these theoretical properties may not be of interest to the machine learning community, we focus on the implications of these results for understanding the underpinnings of neural networks.

It is helpful to compare the orthogonality gap in BN networks to studies on vanilla networks (Bjorck et al., 2018; Daneshmand et al., 2020; Saxe et al., 2013). The function implemented by a vanilla linear network is a linear transformation as the product of random weight matrices. The spectral properties of the product of i.i.d. random matrices are the subject of extensive studies in probability theory (Bougerol et al., 2012). Invoking these results, one can readily check that the orthogonality gap of the hidden representations in vanilla networks rapidly increases since the representations converge to a rank one matrix after a proper scaling (Daneshmand et al., 2020).

4.3 Experimental validations

Our experiments presented in Fig. 2a validate the exponential decay rate of V with depth. In this plot, we see that $\log(V_\ell)$ linearly decreases for $\ell = 1, \ldots, 20$, then it wiggles around a small constant. Our experiments in Fig. 2b suggest that the $\mathcal{O}(1/\sqrt{d})$ dependency on width is almost tight. Since $V(H_\ell)$ rapidly converges to a ball, the average of $V(H_\ell)$ over layers estimates the radius of this ball. This plot shows how the average of $V(H_\ell)$, over 500 layers, changes with the network width. Since the scale is logarithmic in this plot, slope is close to $-1/2$ in logarithmic scale, validating the $\mathcal{O}(1/\sqrt{d})$ dependency implied by Theorem 1.



(a) Orthogonality vs. depth

(b) Orthogonality vs. width

Figure 2: **Orthogonality gap vs. depth and width.** Left: $\log(V(H_\ell))$ vertically versus ℓ horizontally . Right: $\log(\frac{1}{500}\sum_{\ell=1}^{500} V(H_\ell))$ vertically versus d horizontally. The slop of the right plot is -0.46. Mean and 95% confidence interval of 20 independent runs.

5 Gaussian approximation

5.1 Orthogonality yields Gaussian approximation

The established result on the orthogonality gap allows us to show that the distribution of output representation is approximately Gaussian. In standard BN MLPs, the output is a linear transformation of the hidden representation of the last layer. Let $W_\ell H_\ell$ denote the output representation. Note that if H_ℓ is orthogonal, the random matrix $W_\ell H_\ell$ is equal in distribution to a standard Gaussian matrix due to the invariance of standard Gaussian distribution under linear orthogonal transformations. We can formalize this notion of Gaussianity by bounding the Wasserstein-2 distance, denoted by $\mathcal{W}_2$, between the distribution of $W_\ell H_\ell$ and standard Gaussian distribution. The next lemma formally establishes the link between the orthogonality of representations and the distribution of the outputs.

Lemma 2. *Given $G \in \mathbb{R}^{d\times n}$ with i.i.d. zero-mean $1/d$-variance Gaussian elements, the following Gaussian approximation holds:*

$$\mathcal{W}_2\left(\mathrm{law}(W_\ell H_\ell), \mathrm{law}(G/\sqrt{n})\right)^2 \leq 2n\mathbb{E}\left[V(H_\ell)\right]. \tag{4}$$

Combining the above result with Theorem 1 yields the result presented in the next corollary.

Corollary 3. *For $G \in \mathbb{R}^{d\times n}$ with i.i.d. zero-mean $1/d$-variance Gaussian elements,*

$$\mathcal{W}_2\left(\mathrm{law}(W_\ell H_\ell), \mathrm{law}(G/\sqrt{n})\right)^2 \leq 4n\left(1-\frac{2}{3}\alpha\right)^\ell + \frac{6n^2}{\alpha\sqrt{d}} \tag{5}$$

holds under Assumption $\mathcal{A}_1(\alpha, \ell)$.

In other words, the distribution of the outputs contracts to a Wasserstein 2 ball around an isotropic Gaussian distribution as the depth grows. The radius of the Wasserstein 2 ball is at most $\mathcal{O}(1/\sqrt{\text{width}})$. As noted in the last section, $\mathcal{A}_1$ is extensively studied by Daneshmand et al. (2020) where it is shown that $\mathcal{A}_1(\alpha > 0, \infty)$ holds as long as $d = \Omega(n^2)$.

5.2 Deep neural networks as Gaussian processes

Leveraging BN, Corollary 3 establishes the first *non-asymptotic* Gaussian approximation for deep random neural networks. For vanilla networks, the Gaussianity is guaranteed only in the *asymptotic* regime of infinite width (Garriga-Alonso et al., 2018; Neal, 1996; Lee et al., 2017; Neal, 1996; Hazan and Jaakkola, 2015). Particularly, Matthews et al. (2018) links vanilla networks to Gaussian processes when their width is infinite and grows in successive layers, while our Gaussian approximation holds for networks with finite width across layers. Table 1 briefly compares Gaussian approximation for vanilla and BN networks.

Network	Width	Depth	Distribution of Outputs
Vanilla MLP	infinite	finite	Converges to Gaussian as width $\to \infty$
Vanilla Convnet	infinite	finite	Converges to Gaussian as width $\to \infty$
BN MLP (Cor. 3)	(in)finite	(in)finite	In a $\mathcal{O}(\text{width}^{-1/4})$-$\mathcal{W}_2$ ball around Gaussian

Table 1: **Distribution of outputs for random vanilla and BN networks.** For the convolutional network, the width refers to the number of channels. Results for Vanilla MLPs and Vanilla convolution networks are establish by Matthews et al. (2018), and Garriga-Alonso et al. (2018), respectively. Remarkably, Corollary 3 holds for MLP with linear activations.

The link between Gaussian processes and infinite-width neural networks has inspired several studies to rely on Gaussian representations in deep random networks (Klambauer et al., 2017; De Palma et al., 2018; Pennington et al., 2018; Schoenholz et al., 2016; Yang et al., 2019). Assuming the representations are Gaussian, Klambauer et al. (2017) designed novel activation functions that improve the optimization performance, De Palma et al. (2018) studies the sensitivity of random networks, Pennington et al. (2018) highlights the spectral universality in random networks, Schoenholz et al. (2016) studies information propagating through the network layers, and Yang et al. (2019) studies gradients propagation through the depth. Indeed, our analysis implies that including BN imposes the Gaussian representations required for these analyses. Although our result is established for linear activations, we conjecture a similar result holds for non-linear MLPs (see Appendix G).

6 The orthogonality and optimization

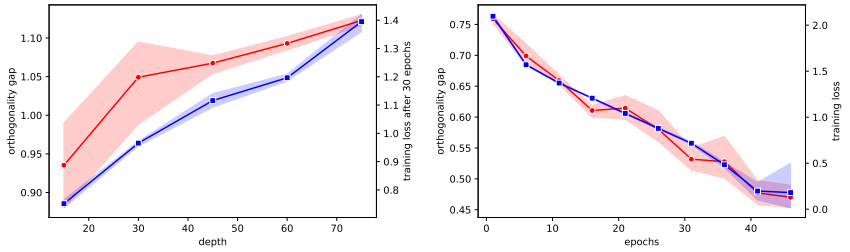
In the preceding sections, we elaborated on the theoretical properties of BN networks in controlled settings. In this section, we demonstrate the practical applications of our findings. In the first part, we focus on the relationship between depth and orthogonality. Increasing depth drastically slows the training of neural networks with BN. Furthermore, we observe that as depth grows, the training slowdown highly correlates with the orthogonality gap. This observation suggests that SGD needs to orthogonalize deep representations in order to start classification. This intuition leads us to the following question: If orthogonalization is a prerequisite for training, can we save optimization time by starting from orthogonal representations? To test this experimentally, we devised a weight initialization that guarantees orthogonality of representations. Surprisingly, even in a network without BN, our experiments showed that this initialization avoids the training slow down, affirmatively answering the question.

Throughout the experiments, we use vanilla MLP (without BN) with a width of 800 across all hidden layers, ReLU activation, and used Xavier's method for weights intialization (Glorot and Bengio,

2010). We use SGD with stepsize 0.01 and batch size 500 and for training. The learning task is classification with cross entropy loss for CIFAR10 dataset (Krizhevsky et al., 2009, MIT license). We use PyTorch (Paszke et al., 2019, BSD license) and Google Colaboratory platform with a single Tesla-P100 GPU with 16GB memory in all the experiments. The reported orthogonality gap is the average of the orthogonality gap of representation in the last layer.

6.1 Orthogonality correlates with optimization performance

In the first experiment, we show that the orthogonality of representations at the initialization correlates with optimization speed. For networks with 15, 30, 45, 60, and 75 widths, we register training loss after 30 epochs and compare it with the initial orthogonality gap. Figure 3a shows the training loss (blue) and the initial orthogonality gap (red) as a function of depth. We observe that representations are more entangled, i.e., orthogonal, when we increase depth, coinciding with the training slowdown. Intuitively, the slowdown is due to the additional time SGD must spend to orthogonalize the representations before classification. In the second experiment, we validate this intuitive argument by tracking the orthogonality gap during training. Figure 3b plots the orthogonality gap of output and training loss for a network with 20 layers. We observe that SGD updates are iteratively orthogonalizing representations, marked by the reduction in the orthogonality gap.



(a) gap and loss vs. depth (b) gap and loss during training

Figure 3: **Orthogonality and Optimization** Left: the orthogonality gap at initialization (red, left axis) and the training loss after 30 epochs (blue, right axis) with depth. Right: the orthogonality gap (red, left axis) and the training loss in each epoch (blue, right axis). Mean and 95% confidence interval of 4 independent runs.

6.2 Iterative orthogonalization accelerates optimization

We have seen that the slowdown in SGD for deeper networks correlates with the orthogonality gap before training. Here we show that by preemptively orthogonalizing representations, we avoid the slowdown with depth. While in MPL with linear activations, hidden representations remain orthogonal simply by taking orthogonal matrices as weights (Pennington et al., 2018; Saxe et al., 2013), the same does not hold for networks with non-linear activations, such as ReLU. To enforce the orthogonality in the absence of BN, we introduce a dependency between weights of successive layers that ensures deep representations remain orthogonal. More specifically, we incorporate the SVD decomposition of the hidden representation of each layer into the initialization of the subsequent layer. To emphasis this dependency between layers and to distinguish it from purely orthogonal weight initialization, we refer to this as *iterative orthogonalization*.

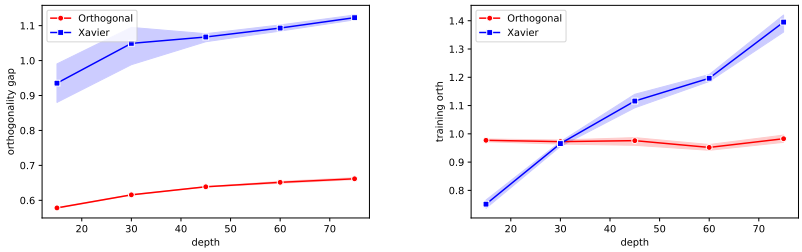
We take a large batch of samples $n \geq d$, as the input batch for initialization. Let us assume that weights are initialized up to layer $W_{\ell-1}$. To initialize W_ℓ, we compute SVD decomposition of the representations $H_\ell = U_\ell \Sigma_\ell V_\ell^\top$ where matrices $U_\ell \in \mathbb{R}^{d\times d}$ and $V_\ell \in \mathbb{R}^{n\times d}$ are orthogonal. Given this decomposition, we initialize W_ℓ by

$$W_\ell = \frac{1}{\|\Sigma_\ell^{1/2}\|_F} V_\ell' \Sigma_\ell^{-1/2} U_\ell^\top \tag{6}$$

where $V_\ell' \in \mathbb{R}^{d\times d}$ is an orthogonal matrix obtained by slicing $V_\ell \in \mathbb{R}^{n\times d}$. Notably, the inverse in the above formula exists when n is sufficiently larger than d [1]. It is easy to check that $V(W_\ell H_\ell) <$

[1] We may inductively assume that H_ℓ is almost orthogonal by the choice of $W_1, \ldots, W_{\ell-1}$. Thus, Σ_ℓ is invertible.

$V(H_\ell)$ holds for the above initialization (see Appendix F), similar to BN. We observed this method effectively orthogonalizes hidden representations when the batch size is sufficiently large. Fig. 4a compares the orthogonality gap of the representations for this initialization method with a standard initialization. By enforcing the orthogonality, this initialization significantly alleviates the slow down of training with depth (see Fig. 4b), with no need for BN.



(a) orthogonality gap (b) training loss

Figure 4: **Iterative orthogonalization.** Horizontal axis: depth. Left: the orthogonality gaps for Xavier's initialization (blue), our initialization (red). Right: the training loss after 30 epochs for Xavier's initialization (blue), our initialization (red). Mean and 95% confidence interval of 4 independent runs.

7 Discussion

To recap, we proved the recurrence of random linear transformations and BN orthogonalizes samples. Our experiments underline practical applications of this theoretical finding: starting from orthogonal representations effectively avoid the training slowdown with depth for MLPs. Based on our experimental observations in Appendix H, we believe that this result extends to standard convolution networks used in practice. In other words, a proper initialization ensuring the orthogonality of hidden representations may replace BN in neural architectures. This future research direction has the potentials to boost the training of deep neural networks and change benchmarks in deep learning.

Although our theoretical bounds hold for MLPs with linear activations, our experiments confirm similar orthogonal stability for various neural architectures. Appendix Figure 5 demonstrates this stability for MLPs with ReLU activations and a convolutional networks. This plot compares the evolution of the orthogonality gap, through layers, for BN networks with vanilla networks. For vanilla networks, we observe that the gap increases with depth. On the contrary, the gap decreases by adding BN layers and stabilizes at a term that is constant with regard to depth. Based on our observations, we conjecture that hidden representations of modern BN networks obey similar stability. A more formal statement of the conjecture is presented in Appendix G.

Broader impact

Since our contribution is understanding of training of neural network, our work is a fundamental research; hence, a broader impact discussion is not applicable.

References

Arora, S., Cohen, N., Golowich, N., and Hu, W. (2018a). A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*.

Arora, S., Li, Z., and Lyu, K. (2018b). Theoretical analysis of auto rate-tuning by batch normalization. *arXiv preprint arXiv:1812.03981*.

Bahri, Y., Kadmon, J., Pennington, J., Schoenholz, S. S., Sohl-Dickstein, J., and Ganguli, S. (2020). Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*.

Bartlett, P. L., Helmbold, D. P., and Long, P. M. (2019). Gradient descent with identity initialization efficiently learns positive-definite linear transformations by deep residual networks. *Neural computation*, 31(3):477–502.

Bietti, A. and Mairal, J. (2019). On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*.

Bjorck, J., Gomes, C., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. *arXiv preprint arXiv:1806.02375*.

Bougerol, P. et al. (2012). *Products of random matrices with applications to Schrödinger operators*, volume 8. Springer Science & Business Media.

Daneshmand, H., Kohler, J., Bach, F., Hofmann, T., and Lucchi, A. (2020). Batch normalization provably avoids rank collapse for randomly initialised deep networks. *NeurIPS*.

De Palma, G., Kiani, B. T., and Lloyd, S. (2018). Random deep neural networks are biased towards simple functions. *arXiv preprint arXiv:1812.10156*.

Eberle, A. (2009). Markov processes. *Lecture Notes at University of Bonn*.

Frankle, J., Schwab, D. J., and Morcos, A. S. (2020). Training batchnorm and only batchnorm: On the expressive power of random features in cnns. *arXiv preprint arXiv:2003.00152*.

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. (2018). Deep convolutional networks as shallow gaussian processes. *arXiv preprint arXiv:1808.05587*.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Hazan, T. and Jaakkola, T. (2015). Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*.

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.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, P.-S., Avron, H., Sainath, T. N., Sindhwani, V., and Ramabhadran, B. (2014). Kernel methods match deep neural networks on timit. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 205–209. IEEE.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*.

Johnson, S. G. (2014). The nlopt nonlinear-optimization package.

Karakida, R., Akaho, S., and Amari, S.-i. (2019). The normalization method for alleviating pathological sharpness in wide neural networks. In *Advances in Neural Information Processing Systems*, pages 6403–6413.

Kemeny, J. G. and Snell, J. L. (1976). *Markov chains*, volume 6. Springer-Verlag, New York.

Khasminskii, R. (2011). *Stochastic stability of differential equations*, volume 66. Springer Science & Business Media.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*.

Kohler, J., Daneshmand, H., Lucchi, A., Zhou, M., Neymeyr, K., and Hofmann, T. (2018). Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. *arXiv preprint arXiv:1805.10694*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kushner, H. and Yin, G. G. (2003). *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media.

Kushner, H. J. (1967). Stochastic stability and control. Technical report, Brown Univ Providence RI.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017). Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*.

Neal, R. M. (1996). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Panaretos, V. M. and Zemel, Y. (2019). Statistical aspects of wasserstein distances. *Annual review of statistics and its application*, 6:405–431.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pennington, J., Schoenholz, S., and Ganguli, S. (2018). The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1924–1932. PMLR.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*.

Sawa, T. (1972). Finite-sample properties of the k-class estimators. *Econometrica: Journal of the Econometric Society*, pages 653–680.

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*.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. (2016). Deep information propagation. *arXiv preprint arXiv:1611.01232*.

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):354–359.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. (2019). A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See $\mathcal{A}_1$
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs are presented in Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] For 2bthe bars are not visible.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 6
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] All papers and codes are cited, with the list in the bibliography section
   (b) Did you mention the license of the assets? [Yes] We used CIFAR10 (Krizhevsky et al., 2009, MIT license) and PyTorch (Paszke et al., 2019, BSD license). Both were cited with their license in the main text.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Codes to create plots in the paper are attached to the supplementary materical
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No personal data was used
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No personally identifiable data was used
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]