Path-BN: Towards Effective Batch Normalization in the Path Space for ReLU Networks

Xufang Luo[1] **Qi Meng**[1] **Wei Chen**[1] **Yunhong Wang**[2] **Tie-Yan Liu**[1]

[1]Microsoft Research, Beijing, China
[2]State Key Laboratory of Virtual Reality Technology and System, Beihang University, Beijing, China,

Abstract

Neural networks with ReLU activation functions (abbrev. ReLU Networks), have demonstrated their success in many applications. Recently, researchers noticed that ReLU networks are positively scale-invariant (PSI) while the weights are not. This mismatch may lead to undesirable behaviors in the optimization process. Hence, some new algorithms that conduct optimization directly in the *path space* (the path space is proven to be PSI) were developed, such as Stochastic Gradient Descent (SGD) in the path space. However, it is still unknown that whether other deep learning techniques such as batch normalization (BN), could also have their counterparts in the path space. In this paper, we conduct a formal study on the design of BN in the path space. First, we propose *path-reparameterization* of ReLU networks, in which the weights in the networks are reparameterized by path-values. Then, the feedforward and backward propagation of the path-reparameterized networks can calculate the values of the hidden nodes and the gradients in the path space, respectively. Next, we design the a novel way to do batch normalization for the path-reparameterized ReLU networks, called *Path-BN*. Specifically, we notice that, path-reparameterized ReLU NNs have a portion of constant weights which play more critical roles to form the basis of the path space. We propose to exclude these constant weights when doing batch normalization and prove that, by doing so, the scale and the direction of the trained parameters can be more effectively decoupled during training. Finally, we conduct experiments on benchmark datasets. The results show that our proposed Path-BN can improve the performance of the optimization algorithms in the path space.

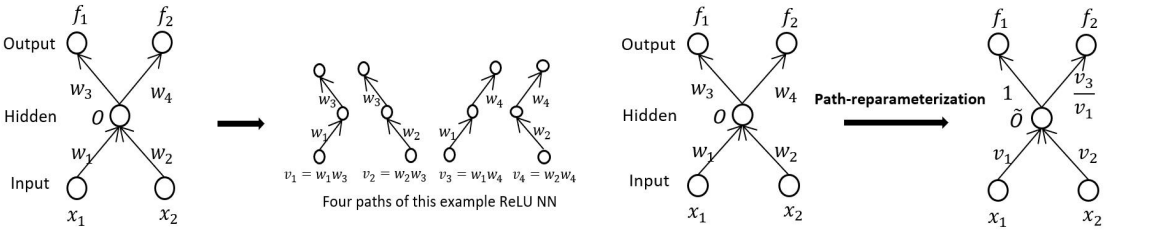
1 INTRODUCTION

In recent years, neural networks with rectified linear unit (ReLU) activation functions (abbrev. ReLU networks) , have been successfully applied to many domains, such as image classification He et al. [2016], Huang et al. [2017], game playing Mnih et al. [2015], and text processing Kim [2014]. Recently, it has been noticed that ReLU networks, are positively scale-invariant (PSI) Neyshabur et al. [2015a, 2016], Dinh et al. [2017], Rangamani et al. [2019], which means when the incoming weights of a hidden node are multiplied by a positive constant and the outgoing weights of the hidden node are divided by this positive constant, the outputs of ReLU networks will keep unchanged for any input. However, the vector space, composed of weights, in which the conventional optimization algorithms, e.g., Stochastic Gradient Descent (SGD), optimize the neural networks, is not PSI, and such mismatch may lead to undesirable behaviors in the optimization process Neyshabur et al. [2015a].

Some recent studies have shown that, one ReLU network can be optimized in a completely new PSI parameter space, i.e., the *path space*, instead of its original weight space Meng et al. [2019]. To be specific, while regarding ReLU networks as directed acyclic graphs (DAGs), we can calculate the outputs of ReLU networks by using the path-values, i.e., the multiplication of weights along each input-output path in the DAG. Here, path-values are invariant to the positive rescaling of weights, which exactly matches the PSI property of ReLU networks, and the optimization algorithms (e.g., SGD) in the path space (i.e., the vector space composed of path-values), are shown to be superior to those in the weight space Neyshabur et al. [2015a, 2016], Meng et al. [2019].

There are already some optimization algorithms developed in the path space, however, to the best of our knowledge, it is still unknown whether other deep learning techniques beyond SGD, can also have their counterparts in the path space. For example, Batch Normalization (BN) is one of the most widely used normalization approaches Ioffe and

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).



Weights	Paths-values
(w_1, w_2, w_3, w_4)	(v_1, v_2, v_3, v_4)
Value of hidden node: $O = ReLU(w_1x_1 + w_2x_2)$ Output: $f_1 = w_3 \cdot O$ $f_2 = w_4 \cdot O$	Value of hidden node: ? Output: $f_1 = v_1a_1x_1 + v_2a_1x_2$ $f_2 = v_3a_1x_1 + v_4a_1x_2$ where $a_1 = I(x_1w_1 + x_2w_2 > 0)$

(a)

Weights	Path-values
(w_1, w_2, w_3, w_4)	(v_1, v_2, v_3)
Value of hidden node: $O = ReLU(w_1x_1 + w_2x_2)$ Output: $f_1 = w_3 \cdot O$ $f_2 = w_4 \cdot O$	Value of hidden node: $\tilde{O} = ReLU(v_1x_1 + v_2x_2)$ Output: $f_1 = \frac{v_3}{v_1} \cdot \tilde{O}$ $f_2 = 1 \cdot \tilde{O}$

(b)

Figure 1: Fig.(a) shows the paths of the example ReLU NN and how to calculate output using path-values. Fig.(b) shows the parameters of ReLU NN after path-reparameterization.

Szegedy [2015], Santurkar et al. [2018], and is crucial for facilitating the training process of neural networks. With the help of BN, the training process of neural networks can be accelerated, by decoupling the scale and direction of weight vectors Arora et al. [2018].

In this paper, we conduct a formal study on the design of BN in the path space. *First*, it is known that, BN is successfully utilized during the forward and backward process in the weight space, but it still remains unclear that, how to ensure the forward propagation in the path space, i.e., the way to calculate the value of the hidden nodes layer by layer via path-values is unknown.[1] For example, in Fig. 1(a), we show how to calculate the outputs of ReLU networks using weights (w_1, w_2, w_3, w_4) and path-values (v_1, v_2, v_3, v_4) (given the sign of the hidden node value), respectively. However, how to calculate the value of the hidden nodes by the path-values is still unknown.

To solve this problem, we propose to re-parameterize the weights in ReLU networks, by the path-values. For example, in Fig. 1(b), weight (w_1, w_2, w_3, w_4) are reparameterized as $(v_1, v_2, 1, v_3/v_1)$. We call it *path-reparameterization* of ReLU networks. We prove that, the output of path-reparameterized network equals the output of the network in path/weight space, but still with the MLP structure. Then, we can conduct the standard feedforward over the path-reparameterized ReLU network, and calculate the value of the hidden nodes by the path-values layer by layers. Furthermore, the backward propagation over path-reparameterized network can calculates the gradients of path-values in a more efficient way than the Inverse-Chain-Rule and Weight-Allocation method proposed in Meng et al. [2019].

Then, we design a novel batch normalization in the path space. We notice that, after the path-reparameterization, a proportion of the weights in the networks, which play critical role to form the basis in the path space, are constant 1 and will not be updated by the optimization algorithms in the path space. If we apply conventional BN to the path-reparameterized ReLU network, the scale and direction of the weights cannot be effectively decoupled during training in path space. Hence, we propose to exclude these constant weights before we do batch normalization for the path-reparameterized ReLU network and add them back afterwards. We prove hat, by doing so, the scale and the direction of the trained parameters in the path space can be effectively decoupled during training. We call the new batch normalization method *Path-BN*. We also prove that, our proposed Path-BN can ensure more stable gradient propagation in the path space, because the gradient magnitude is less sensitive to the standard deviation.

Finally, we train ReLU networks in the path space by SGD with Path-BN on benchmark datasets CIFAR and ImageNet. The results demonstrate that Path-BN can improve the accuracy of SGD in the path space, and outperform SGD with BN in the weight space.

2 BACKGROUND

2.1 RELATED WORK

In recent years, some researchers have conducted many theoretical studies on the *path* of ReLU networks. In terms

[1] In Meng et al. [2019], although ReLU networks are optimized in the path space, the feedforward propagation is still implemented in the weight space.

of generalization, path-based approaches are widely used to analyze the capacity of ReLU networks. In the works Neyshabur et al. [2015b] and Zheng et al. [2018], the relationship between Rademacher complexity and path-norm or basis path-norm have been studied. In the works Theisen et al. [2019]Barron and Klusowski [2018], path sampling methods are designed to produce a sparse approximant of the original network, which leads to tighter generalization error bound. The work E et al. [2018] also leveraged the path representation of ReLU networks, to analyze the generalization error of them. In terms of optimization, there are two typical optimization algorithms designed based on the path of ReLU networks, i.e., PathSGD Neyshabur et al. [2015a] and *G*-SGD Meng et al. [2019]. PathSGD is proposed to optimize the path-norm regularized loss. It heuristically utilizes a coordinate-wised solution for minimizing the loss function. *G*-SGD algorithm directly optimizes the model in the path space, i.e., update the model by utilizing the gradients with respected to path-values which are obtained through Inverse-Chain-Rule (ICR) and Weight-Allocation (WA) methods.

However, the aforementioned studies have not involved normalization approaches Ba et al. [2016], Wu and He [2018], such as batch normalization Ioffe and Szegedy [2015], which are proved to be much crucial for training neural networks. In this paper, we will give a formal study on designing BN in the path space.

2.2 RELU NETWORKS IN THE PATH SPACE

We consider an L-layer feedforward ReLU network f_W^L with weight matrices $W := \{w^l; l = 1, \cdots, L\}$. Given an input $x \in \mathbb{R}^d$, the value of the hidden nodes can be propagated as $o^l(x) = g(w^l o^{l-1}(x))$, for $l = 1 \cdots L-1$, where $g(\cdot) = max(\cdot, 0)$ is the ReLU activation function. The output of the ReLU network can be obtained as

$$f_W^L(x) = w^L o^{L-1}(x). \tag{1}$$

Regarding the network structure as a directed acyclic graph consisting of nodes and edges, a **path** can be defined as: a list of nodes or edges, starting from an input node, successively passing by several hidden nodes along the edges, and finally ending up with an output node. Given an L-layer feedforward ReLU network with width d, we denote a path as $p = [p_0, p_1, p_2, \cdots, p_L] \in \mathbb{Z}_+^{L+1}$ that satisfy $p_l \leq d$ for $l = 0, \cdots, L$. Here, p_l denotes the index of hidden node at layer l that the path passes by. The **path-value** of path p is defined as a multiplication of the weights along p, i.e., $v_p = \prod_{l=1}^{L} w^l_{p_l p_{l-1}}$ where w^l_{jk} represents the weight at j-th row and k-th column of matrix w^l. Therefore, the j-th output $f_j(x)$ can be calculated by using path-values as

$$f_j(x) = \sum_{s=1}^{d} \sum_{p \in \mathbb{P}_{i,j}} v_p \cdot a_p \cdot x_s. \tag{2}$$

Here, $\mathbb{P}_{s,j}$ is a set of paths which satisfy $p_0 = s$ and $p_L = j$, the ReLU activation status of path p, can be calculated as $a_p = \prod_{l=1}^{L-1} \mathbb{I}(o^l_{p_l}(x) > 0)$ where $o^l_j(x)$ denotes the j-th element of vector $o^l(x)$.

Although the output of ReLU networks can be calculated using path-values, how to do forward and backward propagation using path-values is still unknown. In next section, we will show path-reparameterization technique to reparameterize ReLU MLP using path-values.

3 PATH-REPARAMETERIZATION OF RELU NETWORKS

In this section, we first introduce a path-reparameterization method, to ensure the forward propagation in the path space, and calculate the values of hidden nodes by using path-values. Then, we will show that the conventional BN cannot effectively propagate the error signals in the path space, which calls us to design an effective BN in the path space.

3.1 PATH-REPARAMETERIZATION

In this subsection, we introduce path-reparameterization for the parameters of ReLU networks, which can replace each weight with a path-value or a ratio of path-values, and the outputs of the reparameterized network will keep unchanged for any input. For ease of presentation, we use $v^1_{kk} = \prod_{l=1}^{L} w^l_{kk}$ to denote the path-value of the path that satisfies $p_0 = p_1 = \cdots = p_L = k$, and $v^l_{jk} = (\prod_{s<l} w^s_{kk}) \cdot w^l_{jk} \cdot (\prod_{s>l} w^l_{jj})$ to denote the path-value of the path that satisfies $p_0 = \cdots = p_{l-1} = k, p_l = \cdots = p_L = j$. We use $sgn(\cdot)$ to denote the sign function, i.e., $sgn(x) = 1$ if $x > 0$, $sgn(x) = -1$ if $x < 0$.

Theorem 1 *Consider an L-layer ReLU network with weight matrices $\{w^l; l = 1, \cdots, L\}$, if the value of hidden nodes is calculated by path-values as follows:*

(1) for $l = 1$,

$$\hat{o}^l_j(x) = g\left(\sum_{k=1}^{d}\left(\prod_{s=2}^{L} sgn(w^s_{jj})\right) \cdot v^l_{jk} \cdot x_k\right) \tag{3}$$

(2) For $l \geq 2$, $\hat{o}^l_j(x) =$

$$g\left(sgn(w^l_{jj}) \cdot \hat{o}^{l-1}_j(x) + \sum_{k=1, k\neq j}^{d}\left(\prod_{s>l}^{L} sgn(w^s_{kk})\right) \cdot \frac{v^l_{jk}}{v^1_{kk}} \cdot \hat{o}^{l-1}_k(x)\right), \tag{4}$$

it can generate final output $\hat{f}^L(x)$ equal to the one calculated in Eq.(1).

Proof: Based on the PSI property, we will prove Theorem 1 by designing the path-reparameterization process which consists several weight rescaling steps. After these steps, the ReLU network can be equivalently re-parameterized, which means that the outputs will keep unchanged for any input

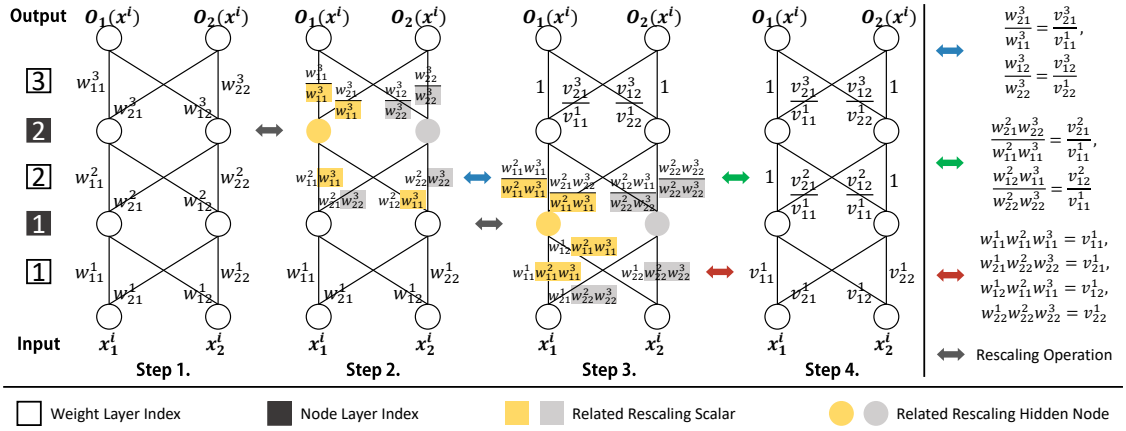


Figure 2: An example to demonstrate the path-reparameterization process for an MLP.

after the path-reparameterization. Here, we will next demonstrate the path-reparameterization process by showing that it can be conducted for a 2-layer MLP, as well as generalized to a multi-layer setting.

As for a 2-layer neural network, Eq.(3) and Eq.(4) have the form

$$\begin{aligned}\hat{o}_k^1(x) &= g\left(\sum_{k'=1}^{d} sgn(w_{kk}^2)\cdot v_{kk'}^l \cdot x_{k'}\right)\\ &= g\left(\sum_{k'=1}^{d} |w_{kk}^2|\cdot w_{kk'}^1 \cdot x_{k'}\right)\end{aligned}$$

and

$$\begin{aligned}\hat{f}_j^2(x) &= \sum_{k=1}^{d} sgn(w_{kk}^2)\cdot \frac{v_{jk}^2}{v_{kk}^1}\cdot \hat{o}_k^1(x)\\ &= \sum_{k=1}^{d} sgn(w_{kk}^2)\cdot \frac{v_{jk}^2}{v_{kk}^1}\cdot g\left(\sum_{k'=1}^{d} |w_{kk}^2|\cdot w_{kk'}^1\cdot x_{k'}\right)\\ &= \sum_{k=1}^{d} sgn(w_{kk}^2)\cdot \frac{v_{jk}^2}{v_{kk}^1}\cdot |w_{kk}^2|\cdot g\left(\sum_{k'=1}^{d} w_{kk'}^1\cdot x_{k'}\right)\\ &= \sum_{k=1}^{d} w_{jk}^2\cdot g\left(\sum_{k'=1}^{d} w_{kk'}^1\cdot x_{k'}\right), \qquad (5)\end{aligned}$$

where Eq.(5) is established according to the definition of v_{kk}^l, v_{jk}^l and the positively scale-invariant property of ReLU function, i.e., $c\cdot g(x) = g(c\cdot x), c>0$.

As for a multi-layer setting, the path-reparameterization process is illustrated in Figure. 2. In particular, this process can be proceeded by rescaling weights orderly, from which is connected with the nodes in the last hidden layer to the first hidden layer (i.e., from the pair (w^L, w^{L-1}) to (w^2, w^1)), and the scalar of the rescaling operation at each node, is the corresponding outgoing diagonal weight. After such rescaling steps, the weights can be reparameterized as follows: (1) $w_{jk}^1 \to v_{jk}^1$; (2) $w_{kk}^l \to 1$; (3) $w_{jk}^l \to \frac{v_{jk}^l}{v_{kk}^1}, l\neq 1, j\neq k$.

In summary, a ReLU network (e.g., Step 1 in Fig. 2) can be equivalently re-parameterized into another network which is parameterized by path-values or a ratio of path-values (e.g., Step 4 in Fig. 2), by rescaling its weights layer by layer. After the path-reparameterization, the outputs of hidden nodes can be calculated by using path-values, in the same way as they are calculated by weights, and also, based on PSI property, the output of the network will keep unchanged after the reparameterization. Hence, Theorem 1 is established. ■

Remark: The path-reparameterization can be extended to other kinds of NN structures such as convolutional neural networks (CNN) and ResNet with ReLU activations. For CNN, we regard each feature map as a hidden node in MLP. For example, we suppose the 4-triple of weight matrices is $[64,64,3,3]$ where the first two elements denote the number of input channels and output channels, the last two elements denote the size of filter which is 3×3. The element $[0,0,1,1],[1,1,1,1],[2,2,1,1],\cdots,[63,63,1,1]$ are the corresponding diagonal weights. Similar as MLP, we scale the incoming weights and outgoing weights for a feature map using the value of the diagonal weights. For ResNet, the network is composed by several residual block and we rescale the weights inside each residual block independently. For example, suppose the structure inside one residual block is a three-layer CNN, the path-reparameterization process for the three layer is the same with that for a CNN.

Theorem 1 explicitly describe how the value of the hidden nodes are calculated by the path-values layer by layer under the condition that the signs of diagonal elements of weight matrices $w^l, (l>1)$ are fixed. After path-reparameterization, the weights of ReLU networks can be reparameterized as follows: (1) $w_{jk}^l \to \left(\prod_{s=2}^{L} sgn(w_{kk}^s)\right)\cdot v_{jk}^l$; (2) $w_{kk}^l \to sgn(w_{kk}^l)$; (3) $w_{jk}^l \to \left(\prod_{s>l}^{L} sgn(w_{kk}^s)\right)\cdot \frac{v_{jk}^l}{v_{kk}^1}, l\neq 1, j\neq k$. Thus, efficient forward and backward propagation in the path-reparameterized network can be ensured.

3.2 THE OPTIMIZATION PROCESS AFTER PATH-REPARAMETERIZATION

In this section, we show how the network be optimized after it has been path-reparameterized. Specifically, after path-reparameterization, we can optimize the loss function according to gradients with respect to the path-values using the optimization algorithm such as SGD as follows. First, we denote the parameter matrix of the path-reparameterized network at the *l-th* layer as U_l. Next, we can use back propagation to obtain the gradient of U_l. According to the chain rule and the relation between the path-values and the weights (i.e., $u^l_{jk} = \frac{v^l_{jk}}{v^1_{kk}}, j \neq k, u^l_{kk} = 1, l \neq 1; u^1_{kk} = v^1_{kk}$), the gradient for $v^l_{jk}, j \neq l$ can be calculated as:

$$\nabla_{v^l_{jk}} \ell = \frac{\nabla_{u^l_{jk}} \ell}{v^1_{kk}}. \tag{6}$$

where ℓ is the loss function. For v^1_{kk}, its gradient can be calculated as:

$$\nabla_{v^1_{kk}} \ell = \nabla_{u^1_{kk}} \ell - \frac{v^l_{jk} \sum_{l=2}^{L} \sum_{j=1}^{d} \nabla_{u^l_{jk}} \ell}{(v^1_{kk})^2} \tag{7}$$

Then, we can derive the update rule of SGD in path space.

Remark: Here, please note that the update rules derived from path-reparameterization is consistent with that provided in Meng et al. [2019], and path-reparameterization provides a much simpler and straightforward way to explain the complex update rules. Moreover, $v^1_{kk}, (k = 1, \cdots, d)$ appears in the denominator for most of its elements, and hence, these elements are initialized to be 1 to ensure numerical stability.

3.3 DISCUSSIONS

In this section, we will discuss the combination of conventional BN and the path-reparameterized network. For ease of presentation, we will assume the signs of diagonal elements of weight matrices $w^l, (l > 1)$ are positive in the following context without loss of generality [2]. Given a mini-batch of inputs $\{x^i; i = 1, \cdots, m\}$, BN normalizes the values of hidden node as $BN(o^l_j(x^i)) = \gamma^l_j \cdot \frac{o^l_j(x^i) - \mu^l_j}{\sigma^l_j} + \beta^l_j$, where μ^l_j and σ^l_j represent the mean and standard deviation of the hidden value o^l_j over the minibatch, and γ^l_j, β^l_j are the scale and shift term, respectively. Normalizing hidden values in this way can decouple the scale and direction of weight vector, which is one the reason for BN can stabilize the training process of deep neural networks in original weight space Kohler et al. [2019]. However, the next proposition shows that if we apply conventional BN to the reparameterized network, the scale of the trained parameters can not be effectively decoupled.

Proposition 2 *Suppose* $\hat{o}^{l-1}_j(x^i) \neq 0$ *and* $\hat{o}^l_j(x^i) = \hat{o}^{l-1}_j(x^i) + \sum_{k=1,k\neq j}^{d} \frac{v^l_{jk}}{v^1_{kk}} \cdot \hat{o}^{l-1}_k(x^i)$. *If the scale of the trained parameters* $\{\frac{v^l_{jk}}{v^1_{kk}}\}_{k=1,\cdots,d;k\neq j}$ *is changed by a positive scalar* $c, (c \neq 1)$ *as* $\{c \cdot \frac{v^l_{jk}}{v^1_{kk}}\}_{k=1,\cdots,d;k\neq j}$ *, the value of* $BN(\hat{o}^l_j(x^i))$ *will be changed.*

We put the proof of Proposition 2 in the Supplementary. Proposition 2 shows that the output after conventional BN still relies on the scale (which can be measured by the L_2-norm) of the trained parameters. Thus conventional BN can not decouple the scale and direction of the trained parameters in path space. The reason is that after path-reparameterization, some parameters are degenerates to constants that won't be trained for optimization in the path space.

Then, we analyze the gradient propagation process for the reparameterized network with conventional BN. We conduct experiments to observe the magnitude of gradient of cross-entropy loss w.r.t the hidden values in each layer. The detailed experimental setting is shown in Section 5.2. The results is shown in the left figure in Fig. 4. The blue curve shows L_2-norm of gradient of loss w.r.t the hidden values in each layer. We can conclude that the L_2-norm of the gradient w.r.t hidden values is exponentially increasing as it is propagated from output layer to input layer. It indicates that the gradient propagation for path-reparameterized network with conventional BN is unstable.

The above discussions motivate us to design BN in the path space by modifying the conventional BN, in order to effectively decouple the scale and direction of the trained parameters in path space, and stably propagate the error signals. We will next detailedly introduce the proposed BN in the path space, i.e., Path-BN.

4 PATH-BN: EFFECTIVE BN IN THE PATH SPACE

In this section, we design the BN in the path space, namely Path-BN, attuned to the path-reparameterized ReLU network. Motivated by the calculation for the values of hidden nodes in Theorem 1, as well as the discussion in Proposition 2, we propose to normalize the terms related to the trained parameters (i.e., Eq. 3 and the *2nd* term in Eq. 4), and exclude the constant terms (i.e., the *1st* term in Eq. 4). Specifically, we use $\tilde{z}^l_j(x^i)$ to denote the hidden value after Path-BN and ReLU, and detailedly describe the forward process of Path-BN in the following two steps.

(1) For the *1st* layer, the operation of Path-BN is the same

[2] When the network is wide, fixing signs of diagonal weights do little damage to the expressiveness of the model because the portion of diagonal weights is small.

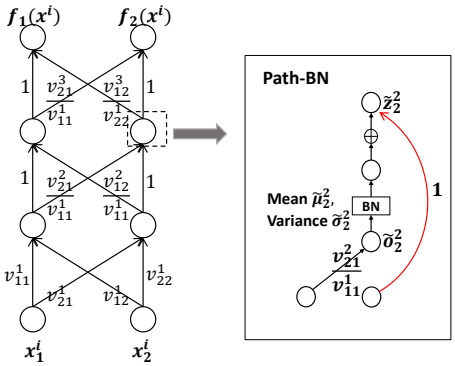


Figure 3: The left figure shows a network after path-reparameterization and the right figure illustrates Path-BN which is applied to a hidden node of the path-reparameterized network.

as the conventional BN, i.e., $\tilde{z}_j^1(x^i) = z_j^1(x^i) = g\left(BN(\hat{o}_j^1(x^i))\right)$.

(2) For the *l-th* layer ($l \geq 2$), as shown in Fig. 3, there are three sub-steps for Path-BN as follows.

First, we calculate the partial weighted summation. Here, the input related to the diagonal constant element in the parameter matrix is excluded, i.e.,

$$\tilde{o}_j^l(x^i) = \sum_{k=1,k\neq j}^{d} \frac{v_{jk}^l}{v_{kk}^1} \cdot \tilde{z}_k^{l-1}(x) \tag{8}$$

Second, normalize the partial weighted summation above, i.e.,

$$BN(\tilde{o}_j^l(x^i)) = \gamma_j^l \cdot \frac{\tilde{o}_j^l(x^i) - \tilde{\mu}_j^l}{\tilde{\sigma}_j^l} + \beta_j^l \tag{9}$$

Third, add the excluded term $\tilde{z}_j^{l-1}(x^i)$ into the normalized partial weighted summation in Eq. 9, i.e.,

$$\tilde{z}_j^l(x^i) = g\left(BN(\tilde{o}_j^l(x^i)) + \tilde{z}_j^{l-1}(x^i)\right) \tag{10}$$

Consequently, Path-BN can be easily combined with the optimization algorithm in the path space for ReLU networks. It is easy to verify that normalizing the path-reparameterized network using Path-BN can decouple the scale and direction of the trained parameters, and thus Path-BN can address the issue of conventional BN showed in Proposition 2.

Next, we analyze the gradient propagation in the reparameterized network with Path-BN and conventional BN. We will show that the gradients can be effectively propagated by using Path-BN in Theorem 3. Here, we use ℓ and $\nabla_{z^l(x^i)}\ell = \left(\frac{\partial \ell}{\partial z_1^l(x^i)}, \cdots, \frac{\partial \ell}{\partial z_d^l(x^i)}\right)$, to denote loss function and the gradient vector w.r.t hidden values in layer l, respectively. Then, we can give an estimation of the norm of $\nabla_{z^l(x^i)}\ell$ and $\nabla_{\tilde{z}^l(x^i)}\mathscr{L}$ in Theorem 3.

Theorem 3 *We use V_l to denote the parameter matrix in layer l for ReLU network after the path-reparameterization. Suppose the scale term is of $\mathscr{O}(1)$ and the minibatch size is large enough, the gradient norm of loss w.r.t $\tilde{z}_j^l(x^i)$ for path-reparameterized network with Path-BN can be upper bounded as*

$$\|\nabla_{\tilde{z}^l(x^i)}\ell\| \leq \mathscr{O}\left(\prod_{s=l+1}^{L} \|\left(I + (\tilde{\sigma}^s)^{-1} \cdot V_s'\right)\|\right) \tag{11}$$

and the gradient norm of loss w.r.t $z_j^l(x^i)$ for path-reparameterized network with BN can be upper bounded as

$$\|\nabla_{z^l(x^i)}\ell\| \leq \mathscr{O}\left(\prod_{s=l+1}^{L} \|(\sigma^s)^{-1} \cdot (I + V_s')\|\right), \tag{12}$$

where $V_l' = V_l - I$, $\sigma^l = diag(\sigma_1^l, \cdots, \sigma_d^l)$, $\tilde{\sigma}^l = diag(\tilde{\sigma}_1^l, \cdots, \tilde{\sigma}_d^l)$, and $diag(a_1, \cdots, a_d)$ represents a diagonal matrix whose diagonal elements are $a_1, \cdots, a_d$.[3]

Proof: As for Eq. (11), the upper bound of gradient norm of in the reparameterized network with Path-BN, according to the chain rule, we have

$$\nabla_{\tilde{z}^l}\ell = \nabla_{\tilde{z}^L}\ell \cdot \left(\frac{\partial \tilde{z}^L}{\partial \tilde{o}^L} \cdot \frac{\partial \tilde{o}^L}{\partial \tilde{z}^{L-1}} + \frac{\partial \tilde{z}^L}{\partial \tilde{z}^{L-1}}\right) \cdots \left(\frac{\partial \tilde{z}^{l+1}}{\partial \tilde{o}^{l+1}} \cdot \frac{\partial \tilde{o}^{l+1}}{\partial \tilde{z}^l} + \frac{\partial \tilde{z}^{l+1}}{\partial \tilde{z}^l}\right),$$

where $\frac{\partial \tilde{z}^s}{\partial \tilde{o}^s}$ equals

$$\frac{\partial \tilde{z}^s}{\partial \tilde{o}^s} = \begin{bmatrix} \frac{\partial \tilde{z}_1^s(x^1)}{\partial \tilde{o}_1^s(x^1)} & \frac{\partial \tilde{z}_1^s(x^1)}{\partial \tilde{o}_2^s(x^1)} & \cdots & \frac{\partial \tilde{z}_1^s(x^1)}{\partial \tilde{o}_d^s(x^m)} \\ \frac{\partial \tilde{z}_2^s(x^1)}{\partial \tilde{o}_1^s(x^1)} & \frac{\partial \tilde{z}_2^s(x^1)}{\partial \tilde{o}_2^s(x^1)} & \cdots & \frac{\partial \tilde{z}_2^s(x^1)}{\partial \tilde{o}_d^s(x^m)} \\ & \cdots & \cdots & \\ \frac{\partial \tilde{z}_d^s(x^m)}{\partial \tilde{o}_1^s(x^1)} & \frac{\partial \tilde{z}_d^s(x^m)}{\partial \tilde{o}_2^s(x^1)} & \cdots & \frac{\partial \tilde{z}_d^s(x^m)}{\partial \tilde{o}_d^s(x^m)} \end{bmatrix}. \tag{13}$$

Assuming that $\tilde{\gamma}^l = diag(\tilde{\gamma}_1^l, \cdots, \tilde{\gamma}_d^l)$ and $\tilde{D}^l(x^i) = diag(\tilde{D}_1^l(x^i), \cdots, \tilde{D}_d^l(x^i))$ where $\tilde{D}_j^l(x^i) = 1$ if $\tilde{z}_j^l(x^i) > 0$, otherwise $\tilde{D}_j^l(x^i) = 0$, we have

(1) if $j_1 = j_2 = j$ and $i_1 = i_2 = i$,

$$\frac{\partial \tilde{z}_j^s(x^i)}{\partial \tilde{o}_j^s(x^i)} = \tilde{D}_j^s(x^i) \cdot \frac{\tilde{\gamma}_j^s}{\tilde{\sigma}_j^s}(1 - \frac{1}{m} - \frac{1}{m} \cdot (BN(\tilde{o}_j^s(x^i))^2);$$

(2) if $j_1 = j_2 = j$ and $i_1 \neq i_2$,

$$\frac{\partial \tilde{z}_j^s(x^{i_1})}{\partial \tilde{o}_j^s(x^{i_2})} = \tilde{D}_j^s(x^{i_1}) \cdot \frac{\tilde{\gamma}_j^s}{\tilde{\sigma}_j^s}(-\frac{1}{m} - \frac{1}{m} \cdot BN(\tilde{o}_j^s(x^{i_1})) \cdot BN(\tilde{o}_j^s(x^{i_2})));$$

(3) if $j_1 \neq j_2$, $\frac{\partial \tilde{z}_{j_1}^s(x^{i_1})}{\partial \tilde{o}_{j_2}^s(x^{i_2})} = 0$.

Obviously, most of the elements in the matrix equals to zero. Under the assumption that $\tilde{z}_j^l(x^i)$ is bounded for all i, j, l, the

[3] Here, $\|\cdot\|$ denotes the spectral norm of a matrix or the L_2 norm of a vector.

above matrix tend to be the diagonal matrix $\tilde{D}^l \cdot \tilde{\gamma}^l \cdot (\tilde{\sigma}^l)^{-1}$ if m approaches ∞. Under the assumption that the scale term $\tilde{\gamma}_i^l$ is of $\mathcal{O}(1)$ and the minibatch size m is large enough, we have $\tilde{D}^l \cdot \tilde{\gamma}^l \cdot (\tilde{\sigma}^l)^{-1} \leq \mathcal{O}((\tilde{\sigma}^l)^{-1} I)$.

As for the matrix $\frac{\partial \tilde{o}^s}{\partial \tilde{z}^{s-1}}$, we have

$$\frac{\partial \tilde{o}^s}{\partial \tilde{z}^{s-1}} = I. \tag{14}$$

As for the matrix $\frac{\partial \tilde{o}^s}{\partial \tilde{z}^{s-1}}$, we have

$$\frac{\partial \tilde{o}^s}{\partial \tilde{z}^{s-1}} = \begin{bmatrix} V_s' & 0 & \cdots & 0 \\ 0 & V_s' & \cdots & 0 \\ & & \cdots & \\ 0 & 0 & \cdots & V_s' \end{bmatrix} \tag{15}$$

Thus, using the fact that $\|AB\| \leq \|A\|\|B\|$, we can get the result in the theorem.

Besides, the upper bound of gradient norm of in the reparameterized network with BN (i.e., Eq. (12)), can be proved in the same way. ■

According to Theorem 3, we have the following corollary.

Corollary 4 *Suppose $\lambda = \|V_l'\|$. A sufficient condition for that $\|\nabla_{z^l(x^i)}\ell\|$ and $\|\nabla_{\tilde{z}^l(x^i)}\ell\|$ can be upper bounded by constant (i.e., will not diverge as $L \to \infty$) is $\sigma_j^l \in [\frac{L(1+\lambda)}{L+1}, \frac{L(1+\lambda)}{L-1}]$ and $\tilde{\sigma}_j^l \in [\lambda L, \infty)\ \forall j$, respectively.* [4]

Discussion: According to Theorem 3, the standard derivation σ_j^l can not be much larger or smaller than 1 if we want the gradient of path-reparameterized network with conventional BN stably propagated, while we only need $\tilde{\sigma}_j^l$ not be too small if we want the gradient of path-reparameterized network with Path-BN stably propagated. Although σ_j^l is not a variable that we can control in the training process, corollary 4 just show the tolerance on the magnitude of the standard derivation to ensure stable back-propagation. Corollary 4 shows explicit range for standard derivations and we can conclude that the range for $\tilde{\sigma}_j^l$ is much larger than that for σ_j^l. Furthermore, if $\lambda \leq \frac{1}{L}$, we have $[\frac{L(1+\lambda)}{L+1}, \frac{L(1+\lambda)}{L-1}] \subset [\lambda L, \infty)$ (i.e., the range for σ_j^l is fully contained in the range $\tilde{\sigma}_j^l$), which shows that the gradient norm for NN with Path-BN is less sensitive to the magnitudes of the standard deviations in batch normalization. It indicates the backward stability of ReLU network with Path-BN. We will observe the standard deviations in each layer in Section 5.2. The results in Figure. 4 demonstrate that the gradient of ReLU network with Path-BN is more stably propagated compared to NN with conventional BN.

[4] Please note that the conditions are tight in the sense that if we change L in the ranges to be $L^{1-\varepsilon}$, the upper bound of both the gradient norms for BN and Path-BN will diverge.

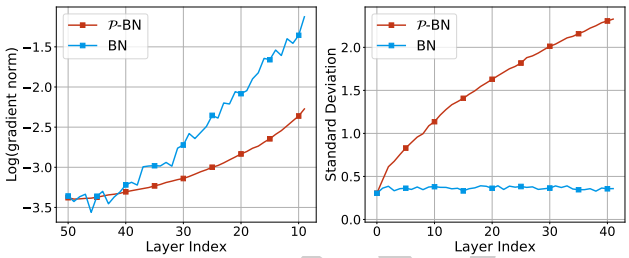


Figure 4: Experimental observations on standard derivation and gradient norm for path-reparameterized network with BN and Path-BN.

Remark: Although Theorem 3 is established for the gradient with respect to the hidden output, according to Equation (6) and (7), the stability of $\nabla_{z^l}\ell$ can ensure the stability of $\nabla_{u_{jk}^l}\ell$, and hence, the stability of $\nabla_{v_{jk}^l}\ell$.

5 EXPERIMENTS

In this section, we conduct experiments by first comparing the performance of Path-BN with the baselines on various datasets, and then showing some experimental observations to support theoretical analyses.

5.1 PERFORMANCE EXPERIMENTS

In this section, we evaluate the performance of Path-BN on training deep neural networks by conducting experiments on CIFAR-10, CIFAR-100, and ImageNet datasets. First, we will introduce the network structures and compared methods.

5.1.1 Network Structures and Compared Methods

In this subsection, we describe the network structures and the compared methods. *First*, as for the network structures, we apply Path-BN to train ResNet and PlainNet He et al. [2016]. *Second*, we show the setting details of four compared methods as follows.

(1) SGD+BN: We use SGD to train the network with BN.

(2) *G*-SGD+BN: We use *G*-SGD to train the network with the conventional BN.

(3) *G*-SGD+BN(wnorm): We use a method which was intuitively proposed to handle the conventional BN without any theoretical analysis Meng et al. [2019]. Here, the gradient of the path-value, is normalized by the L2-norm of the weights pointing to the same hidden unit.

(4) *G*-SGD+Path-BN: Our proposed Path-BN targets on BN in the path space, and the network with Path-BN should be optimized by the algorithms in the path space. Thus, we use *G*-SGD to train the network with Path-BN.

Dataset	Method	PlainNet		ResNet		
		18	34	18	34	50
CIFAR-10	SGD+BN	6.93% (±0.12)	7.76% (±0.22)	6.76% (±0.10)	6.40% (±0.09)	6.55% (±0.24)
	G-SGD+BN	6.66% (±0.12)	6.74% (±0.13)	6.84% (±0.30)	6.54% (±0.06)	6.62% (±0.19)
	G-SGD+BN(wnorm)	6.26% (±0.17)	6.67% (±0.26)	6.31% (±0.14)	6.33% (±0.15)	6.31% (±0.14)
	G-SGD+Path-BN$^{\text{ours}}$	**5.99% (±0.13)**	**6.04% (±0.16)**	**6.04% (±0.14)**	**5.66% (±0.10)**	**5.99% (±0.12)**
CIFAR-100	SGD+BN	28.10% (±0.19)	33.37% (±0.41)	26.97% (±0.12)	26.42% (±0.23)	25.62% (±0.18)
	G-SGD+BN	27.08% (±0.35)	28.19% (±0.35)	27.13% (±0.39)	26.61% (±0.10)	25.99% (±0.40)
	G-SGD+BN(wnorm)	26.76% (±0.05)	27.61% (±0.24)	26.60% (±0.17)	26.72% (±0.27)	25.65% (±0.22)
	G-SGD+Path-BN$^{\text{ours}}$	**26.70% (±0.10)**	**27.04% (±0.12)**	**26.39% (±0.10)**	**26.24% (±0.29)**	**25.29% (±0.19)**

Table 1: Test error rate on CIFAR-10 and CIFAR-100.

5.1.2 CIFAR

We conduct experiments on CIFAR-10 dataset and CIFAR-100 dataset Krizhevsky et al. [2009]. Here, we train ResNet of 18, 34, and 50 layers, and PlainNet of 18 and 34 layers, respectively. As for the hyper-parameters of methods mentioned in their corresponding papers, i.e., SGD+BN He et al. [2016] and G-SGD+BN(wnorm) Meng et al. [2019], we use the same settings as their original ones. Besides, we tune the hyper-parameters for other methods, i.e., G-SGD+BN, and the proposed G-SGD+Path-BN.

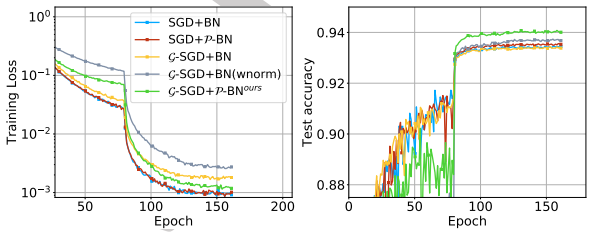
In Table 1, we show the performance results on the test error rate. In Figure 5, we plot the training curves and test accuracy for training ResNet-50 on CIFAR datasets. Such results demonstrate that: *(1)* G-SGD+Path-BN outperforms others on all tested datasets and network structures, which shows the superiority of Path-BN; *(2)* Combining the path space and the conventional BN directly hurts performance, which empirically motivates us to propose Path-BN; *(3)* G-SGD+Path-BN outperforms G-SGD+BN(wnorm), which shows the benefit and significance of our proposed formal analyses.

5.1.3 ImageNet

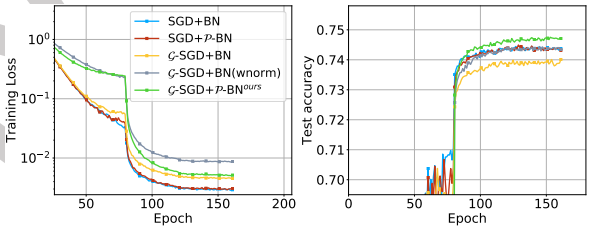
We conduct experiments on ImageNet dataset Russakovsky et al. [2015]. Here, we train ResNet of 50, 101 and 152 layers, to demonstrate that the proposed Path-BN can help to train very deep networks on a large dataset, and we compare the performance results between SGD+BN and G-SGD+Path-BN. The method for tuning the hyper-parameters is similar to that in Section 5.1.2.

In Fig. 6, we plot the training curve and test accuracy during training ResNet of 50, 101, and 152 layers. These results well demonstrate the superiority of G-SGD+Path-BN, and show that for the large dataset and very deep networks, Path-BN can still outperform the conventional BN. [5]

[5] We put the explanation on computational cost of Path-BN compared to BN in Supplementary.



(a) CIFAR-10.



(b) CIFAR-100.

Figure 5: Training curves and test accuracy for training ResNet-50 on CIFAR datasets.

5.2 OBSERVATIONS ON STANDARD DEVIATIONS AND GRADIENTS

After analyzing the advantage of Path-BN over the conventional BN theoretically in Section 3.3 and 4, we now provide some experimental observations to support our analyses. Here, we will show the magnitude of standard deviations in BN, and the gradients of ReLU networks with conventional BN and Path-BN.

In these experiments, we use a stacked CNN of 50 layers with 128 channels in each layer, and use CIFAR-10 as training data, with the batch size of 128. Then, we add the conventional BN or Path-BN at the end of each hidden layer in the stacked CNN, which is denoted as *BN* and *Path-BN*, respectively. For the model with the conventional BN, it is randomly initialized. For the model with Path-BN, the diagonal elements in parameter matrices (expect for the *1st* layer) are initialized to 1 (cf. Section 3), and other elements are initialized with small random values.

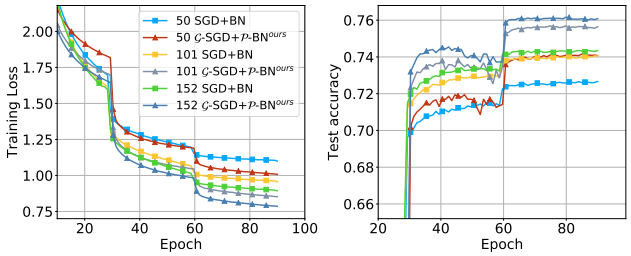


Figure 6: Curves for training ImageNet.

Because the theoretical analyses are not related to the training process, we will just observe some quantities at initialization as follows.

We log standard derivation σ_j^l for each hidden node during the forward process of the *1st* mini-batch, and then calculate the average standard deviation among the nodes in each layer. Then we log the L_2-norm of gradients of loss w.r.t. the hidden values in each layer. The curves regarding standard deviations and the log of the gradient norm are shown in Fig.4. From Fig. 4, we can observe that: *(1)* For the conventional BN, the standard deviations are smaller than 1 and gradient norm grows exponentially as the depth decreases. *(2)* For Path-BN, the standard deviation grows when the layer index becomes larger, and most of them are larger than 1. The gradient norm grows slower than that of NN with conventional BN. These observations match the claim in Theorem 3 that Path-BN is less sensitive to the magnitudes of standard derivations which is one of the reasons for relatively stable gradient propagation.

6 CONCLUSION

In this paper, we conduct a formal study on the design of BN in the path space. We propose path-reparameterization of ReLU networks to ensure the forward and backward propagation in the path space. Via path-reparameterization, the values of hidden nodes can be calculated by using path-values. Then, we design a new batch normalization method for path-reparameterized ReLU networks, called *Path-BN*. We prove that Path-BN can efficiently decouple of scale and direction of trained weight in the path space and can ensure stable gradient propagation. Experiments on two benchmark datasets, CIFAR and ImageNet show that our proposed Path-BN can improve optimization algorithms in the path space. The path-reparameterization gives the path representation of ReLU networks an explicit characterization, and it makes the study of other techniques in the path space easier. Based on the path-reparameterization, we will explore the counterpart of other techniques in path space, such as layer normalization and dropout in future.

References

Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. *arXiv preprint arXiv:1812.03981*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Andrew R Barron and Jason M Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv preprint arXiv:1809.03090*, 2018.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

Weinan E, Chao Ma, and Lei Wu. A priori estimates for two-layer neural networks. *arXiv preprint arXiv:1810.06397*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Thomas Hofmann, Ming Zhou, and Klaus Neymeyr. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *AISTATS*, pages 806–815, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Qi Meng, Shuxin Zheng, Huishuai Zhang, Wei Chen, Zhi-Ming Ma, and Tie-Yan Liu. G-SGD: Optimizing ReLU neural networks in its positively scale-invariant space. In *ICLR*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Behnam Neyshabur, Ruslan Salakhutdinov, and Nathan Srebro. Path-SGD: Path-normalized optimization in deep neural networks. *arxiv preprint arXiv:1506.02617*, 2015a.

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

Behnam Neyshabur, Yuhuai Wu, Ruslan R Salakhutdinov, and Nati Srebro. Path-normalized optimization of recurrent neural networks with ReLU activations. In *NeurIPS*, pages 3477–3485, 2016.

Akshay Rangamani, Nam H Nguyen, Abhishek Kumar, Dzung Phan, Sang H Chin, and Trac D Tran. A scale invariant flatness measure for deep network minima. *arXiv preprint arXiv:1902.02434*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 115(3): 211–252, 2015.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *NeurIPS*, pages 2483–2493, 2018.

Ryan Theisen, Jason M Klusowski, Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Global capacity measures for deep relu networks via path sampling. *arXiv preprint arXiv:1910.10245*, 2019.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, pages 3–19, 2018.

Shuxin Zheng, Qi Meng, Huishuai Zhang, Wei Chen, Nenghai Yu, and Tie-Yan Liu. Capacity control of ReLU neural networks by basis-path norm. *arXiv preprint arXiv:1809.07122*, 2018.