This draft is the supplementary materials for the paper "Path-BN: Towards Effective Batch Normalization in the Path Space for ReLU Networks".

1 OTHER CHOICES ON PATH-REPARAMETERIZATION

In this section, we discuss other choices on path-reparameterization and the influence. The path-reparameterization method using path-values showed in the proof of Theorem 1 is not unique. For example, we can obtain another kind of path-reparameterization, via multiplying and dividing the incoming and outgoing weights of $O_j^1$ by $v_{11}$ in step 4 of Figure.2 in the main paper. This will not influence analyses much in this paper for the following two reasons. First, whatever path-reparameterization method used, each path-reparameterized network can serve as a sufficient representation for ReLU networks in the path space, because the outputs of the network will keep unchanged for any input after path-reparameterization. Second, our studies are based on Theorem 1 in the main paper, which can be generalized to other path-reparameterization methods.

2 PROOF FOR PROPOSITION 2

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

**Proof:** If the scale of the trained parameters $\{\frac{v_{jk}^l}{v_{kk}^l}\}_{k=1, \ldots, d, k \neq j}$ is changed by a positive scalar $c$, we have

$$BN\left(\hat{o}_j^{l-1}(x') + \sum_{k=1,k \neq j}^d c \cdot \frac{v_{jk}^l}{v_{kk}^l} \cdot \hat{o}_k^{l-1}(x')\right)$$

$$\neq BN\left(c \cdot \hat{o}_j^{l-1}(x') + \sum_{k=1,k \neq j}^d c \cdot \frac{v_{jk}^l}{v_{kk}^l} \cdot \hat{o}_k^{l-1}(x')\right)$$

$$= BN\left(c \cdot \hat{o}_j^{l-1}(x') + \sum_{k=1,k \neq j}^d \frac{v_{jk}^l}{v_{kk}^l} \cdot \hat{o}_k^{l-1}(x')\right).$$

3 DETAILS ON EXPERIMENTS

3.1 EXPERIMENTAL SETTING DETAILS

In particular, following the settings in [Meng et al., 2019], we apply $G$-SGD or Path-BN with residual blocks, because there is no positive scaling invariance across residual blocks. In addition, we introduce a coefficient $\lambda (\lambda < 1)$ for $\hat{z}_j^{l-1}(x')$ for experiments of ResNet, to prevent the output exploding during the forward process, and $\lambda$ for all layers in ResNet is set to be 0.1.

Here we describe pre-processing steps, which is same as [He et al., 2016]. For CIFAR-10 and CIFAR-100, we randomly crop the input image by size of 32 with padding size of 4, and normalize every pixel value to $[0, 1]$. Then the random horizontal flipping to the image is applied. For ImageNet, we randomly crop the input image by size of 224, and normalize every pixel value to $[0, 1]$. Then the random horizontal flipping to the image is also applied.

We use 1 NVIDIA Tesla P100 or P40 GPU to run the experiments on CIFAR, and use 4 NVIDIA Tesla P100 or P40 GPUs in one machine to run the experiments on ImageNet. All experiments are averaged over 5 runs with different random seeds, and PyTorch [Paszke et al., 2017] is used for implementation.
Here we list hyper-parameters used for CIFAR-10 and CIFAR-100. For all hidden layers, we add other baseline, i.e., SGD+Path-BN, in which we use SGD to train the network with Path-BN, and in Table 1 we show the test error for SGD+Path-BN. We can observe that G-SGD+Path-BN outperforms SGD+Path-BN, which demonstrate the benefits of optimizing in the path space.

### 3.3 Analysis of Computational Costs

Different from the conventional BN, Path-BN is to identify the diagonal elements of weight matrices. For convolutional networks, the outputs of hidden nodes in MLP always correspond to the feature maps. Hence, on the one hand, if the stride length is 1, we can then set the diagonal elements in weight matrices to 0 and add skip connections for all layers (except the 1st layer), which will not bring extra computational cost. On the other hand, if the stride length is larger than 1, we can also set the diagonal elements in weight matrices to 0, and additionally, use a fixed-weight convolutional layer, with only diagonal elements fixed to 1 and others fixed to 0. Thus, with the skip connection or the fixed-weight convolutional layer, the back propagation process can be easily implemented. Therefore, according to the above implementation, Path-BN will not bring much extra computational cost compared with the conventional BN. Statistically, in our experiments, the ratio of run time between G-SGD+Path-BN and SGD+BN, is less than 4 : 3, and the extra computational cost is mainly brought by the update of G-SGD instead of Path-BN.

### References


<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>PlainNet</th>
<th>ResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>SGD+Path-BN</td>
<td>7.00% (±0.08)</td>
<td>6.78% (±0.15)</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>SGD+Path-BN</td>
<td>28.57% (±0.46)</td>
<td>26.71% (±0.22)</td>
</tr>
</tbody>
</table>

Table 1: Test error rate on CIFAR-10 and CIFAR-100 for SGD+Path-BN.