Rethinking the Architecture of Very Deep Residual Networks

First Author\textsuperscript{1}, Second Author\textsuperscript{2}, Third Author\textsuperscript{2,3}

\textsuperscript{1}First Affiliation
\textsuperscript{2}Second Affiliation
\textsuperscript{3}Third Affiliation

\{first, second\}@example.com, third@other.example.com,

Abstract

Residual network (ResNet) is the most commonly used solution for training very deep neural networks. However, the first version of ResNet still has a performance drop when training over 1000 layers. In the second version, pre-activation ResNet that preserves identity mappings can mitigate this problem. In this paper, we revisit the problem of training very deep ResNet and argue that the variance explosion problem can be another limitation to train very deep ResNet. We proposed PMResNet (plus and minus ResNet) for training very deep ResNet. Compared to the original ResNet, PMResNet has no extra parameters or needs extra computation, but has much better convergence when the network is very deep. There were three key points to improve the performance: 1) Plus-Minus block can preserve post-activation structure while ensuring identity mapping; 2) \(1/\sqrt{T}\) initialization will effectively reduce the variance of the model output; 3) using global BN layer. In experiments, our approach successfully improves the performance of various representative architectures, and we report the improved results on CIFAR-10/100 and ImageNet. Code will be made publicly available.

1 Introduction

The development of neural networks is one of the major advances in the artificial intelligence community. Since deeper neural networks usually have better performance but are harder to train, training very deep neural networks remains an interesting challenge, in both theory and practice. Vanishing/exploding gradient problems may be the first obstacle for training deep neural networks. Special initialization techniques [Glorot and Bengio, 2010; He et al., 2015] and normalization mechanisms [Ioffe and Szegedy, 2015] can solve this problem. As the networks getting more deeper, network degradation becomes another problem: with the network depth increasing, even accuracy on the training dataset gets saturated and then degrades rapidly. Residual networks (ResNet) [He et al., 2016a] that have skip connections provide bypaths for propagating signals, making the training of deep neural networks much easier. ResNet solves the degradation problem and is the first ultra deep neural network.

However, the first version of ResNet still has a performance drop when training over 1000 layers. One reason for the performance drop is that the first version cannot preserve identity mappings. Although put the ReLU layer (between two residual blocks) before residual connections is a naive solution to solve the identity mapping problem, it will leads to a non-negative output from the residual branches, leading to a worse performance. The second version of pre-activation ResNet [He et al., 2016b] changes the order of layers in the residual branch and boosts the performance of very deep ResNet. But our experiments show that for extremely deep networks, there will be performance degradation regardless of the original structure or the pre-act structure (see Fig. 1).

We revisit the changes between the original ResNet and pre-activation ResNet, and found a small difference which is less mentioned before but important for pre-activation ResNet training. The pre-activation ResNet has a extra batch normalization (BN) layer after the last residual block compare with the original ResNet. We found that without this layer, pre-activation ResNet with very deep layers (such as 1000+ layers) cannot even converge. Further research shows that the variance of the input of this layer is surprisingly large. For pre-activation ResNet 1202, the variance can be \(\sim 100\).
Although the design of skip connections solves the problem of network degradation, we argue that the variance explosion could be another limitation to the training of very deep ResNet.

The variance explosion problem is easy to understand. Suppose $x_t$ is the input to the $t$th residual branch, $F(\cdot; \Theta_t)$ is the $t$th residual block where $\Theta_t$ is the parameters. The output of this block (also the input to the $(t+1)$th residual block) can be formulated as $x_{t+1} = x_t + F(x_t; \Theta)$. The variance of $x_{t+1}$ is inevitably greater than the variance of $x_t$. As the network getting very deep, the variance of the last residual block’s output would be extremely large, making the training more difficult. Using BN in the network can prevent this problem from becoming a gradient explosion problem. However, extremely large variance still has some bad influence on training, which we will later discuss in detail. We need to mention that we are not the first to observe the variance explosion problem. [Zhang et al., 2019] is the first to state this problem explicitly. We propose a new $1/\sqrt{T}$ initialization method to solve the variance explosion problem.

We proposed PM (plus and minus)-ResNet to preserve the identity mapping and keep the conv-BN-ReLU (post-activation) architecture, which we will show that is better than BN-ReLU-conv (pre-activation). Compared to the original ResNet, PmResNet has no extra parameters or needs extra computation, but has much better convergence when the network is very deep. The formulation of PmResNet is quite simple: $x_{t+1} = x_t + (-1)^t F(x_t; \Theta)$ and $F$ is the stack of conv-BN-ReLU sequences. The difference between PmResNet and original ResNet: $x_{t+1} = \text{ReLU}(x_t + F(x_t; \Theta))$ is that PmResNet preserves identical mappings with alternatively use plus and minus in residual connections. The pre-activation ResNet can be formalized as $x_{t+1} = x_t + F'(x_t; \Theta)$, where $F'$ is the stack of BN-ReLU-conv sequences. We will compare post-activation and pre-activation, our studies show that PmResNet has much better convergence and performance especially when the network is very deep.

Both PmResNet and $1/\sqrt{T}$ initialization are very neat and only requires several lines of codes. Codes to replicate the experiments of this paper will be made publicly available.

In the remaining of this paper, related work is presented in Section 2. We then analyze the exploding variance problem of residual networks, and develop PmResNet in Section 3. In Section 4 we quantify the properties of PmResNet and compare it against state-of-the-art methods on real world benchmarks.

2 Related Work

Residual Neural Network. A residual neural network (ResNet) is a kind of artificial neural network, which utilize skip connections or shortcuts to jump over some layers.

Typical ResNet models are implemented with double- or triple-layer skips that contain nonlinearities (ReLU) and batch normalization in between. [He et al., 2016a] proposes the the first ultra deep convolutional neural network and shows that it can solve the degradation problem.

An additional weight matrix may be used to learn the skip weights; such as HighwayNets [Srivastava et al., 2015]. Models with several parallel skips are referred to as DenseNets [Huang et al., 2017], which connects each layer to every other layer in a feed-forward fashion.

Weight Initialization. Weight initialization process usually has a high influence in the performance of neural networks. Initialization with random weights drawn from Gaussian distributions become very popular after the success of CNNs in IVSRC 2012 [Krizhevsky et al., 2012]. With fixed standard deviations (e.g., 0.01 in AlexNet), it is not possible to train very deep models from scratch [Simonyan and Zisserman, 2015].
[Glorot and Bengio, 2010] proposed to adopt a properly scaled uniform distribution for initialization. This called “Xavier” initialization in [Jia et al., 2014]. It estimated the standard deviation on the number of input and output channels of the layers. Despite invalidity of the linear activation assumption, “Xavier” initialization works well in many applications. [He et al., 2015] extended this formula to the ReLU ([Glorot et al., 2011]) non-linearity and showed its superior performance. [Saxe et al., 2014] showed that orthonormal matrix initialization works much better for linear networks than Gaussian noise. It also work for networks with non-linearities.

Normalization. Normalization of training data has been known to be helpful in optimization. For example, whitening of data is a way to remove the underlying correlation in the data, and is widely used for training shallow models such as Support Vector Machines and Logistic regression. Normalization layers in deep networks had been widely used for a long time. [Krizhevsky et al., 2012] proposed Local Response Normalization (LRN) in AlexNet. It computes the statistics in a small neighborhood for each pixel.

Batch Normalization (BN) [Ioffe and Szegedy, 2015] is a highly successful and widely used normalization method. Its use of mini-batch mean and variance statistics to normalize the activations. This has been shown to accelerate learning and enable training of very deep neural network architectures. However, BN did not perform well if the mini-batch size is too small. Several normalization methods have been proposed to address these issues.

Layer Normalization (LN) [Ba et al., 2016] computes the normalization statistics from the entire layer, and Instance Normalization (IN) [Ulyanov et al., 2016] performs BN-like normalization statistics over groups of channels. The ideal group size is experimentally determined.

3 Methodology

3.1 Variance Explosion Problem in ResNet

Though similar statements have been made in [Zhang et al., 2019], here we briefly show the problem again for self-consistency. Let $x_t$ be the input to the $t$th residual block, $F(\cdot; \Theta_t)$ be the residual block, and $\Theta_t$ be the parameters of the $t$th residual block. [Zhang et al., 2019] point out that ResNet (without BN) output variance grows exponentially with network depth. We will analyze the variance of various structures containing BN layers. As shown in Figure 2, the order of components are slightly different:

- Original ResNet and ReLU only pre-activation. For the architectures of Figure 2 (a, c), the last component is a BN layer, the variance of $F(x_t; \Theta_t)$ is $\gamma_t^2$ where $\gamma_t$ is the weight of the last BN layer.
- Pre-activation. As show in Figure 2 (b), the last component is a convolutional layer, the variance of $F(x_t; \Theta_t)$ is approximately 1 with initialization methods such as [He et al., 2015].

- ReLU before skip connection. If the last component is a BN-ReLU sequence (Figure 2 (d)), the variance of $F(x_t; \Theta_t) = c\gamma_t^2$ where $\gamma_t$ is the weight of the BN layer before the last ReLU layer $1$.

Since the weights of BN layers are generally around 1, the variance of the residual branch is $O(1)$ for all architectures. Consider the residual connection $x_{t+1} = x_t + F(x_t; \Theta_t)$, since the residual block $F$ is a nonlinear system with multiple nonlinear activation layers, we can assume $x_t$ and $F(x_t; \Theta_t)$ are not corrected: $\text{Var}[x_{t+1}] = \text{Var}[x_t] + \text{Var}[F(x_t; \Theta_t)]1^2$. Suppose there are $T$ residual blocks, we can rewrite the expression of $x_T$ using all residual blocks ($x_0$ denotes the input):

$$\text{Var}[x_T] = \text{Var}[x_0] + \sum_{t=0}^{T-1} \text{Var}[F(x_t; \Theta_t)] = O(T). \quad (1)$$

Thus the variance of residual block’s output grows linearly with the network’s depth, leading to a variance explosion problem when the network is very deep.

Global BN. If there is no BN layer after the large variance residual output, the network would have a gradient explosion problem and be hard to optimize. Networks with BN layers would still have some optimization difficulties. Let $\{y_i\}_{i=1}^B$ be a batch of data, the batch mean and variance be $\mu_B$ and $\sigma_B^2$. The output of the BN layer (ignore the affine operation) is $z_t = (y_t - \mu_B)/\sigma_B$. The back propagation of the BN layer is:

$$\frac{\partial \text{Loss}}{\partial y_t} = \frac{1}{\sigma_B} \left[ \frac{\partial \text{Loss}}{\partial z_t} - \frac{z_t}{B} \sum_{j=1}^B \frac{\partial \text{Loss}}{\partial z_j} \right] - \frac{1}{B} \sum_{j=1}^B \frac{\partial \text{Loss}}{\partial z_j} \quad (2)$$

Since $z_t$ is a normalized term, $\sigma_B$ is the only term in the gradient that can be very large. A linearly increasing variance would decrease the gradient signal: $\frac{\partial \text{Loss}}{\partial y_t} \propto O(1/\sqrt{T})$ where $T$ is the number of residual blocks.

1/√$T$ Initialization. As we derived earlier, $\text{Var}[x_T] = O(T)$. We propose to use $1/\sqrt{T}$ to initialize the weights of all last BN layers in the residual branches (by default, the weights of BN layers are initialized as 1). The increased variance by each residual block is $O(1/T)$, and the variance of the last block’s output is $O(1)$. This initialization method can also improve the performance of previous ResNet architectures (Figure 2 a & c). Note that the “zero-γ” initialization that uses zeros to initialize the weights of BN layer is also frequently used. We will later show that $1/\sqrt{T}$ initialization is better than the “zero-γ” initialization in experiments. The effectiveness of “zero-γ” initialization may come from the fact that $1/\sqrt{T}$ is close to 0 when $T$ is large.

---

1The factor $c$ is approximately between 0 and 0.5: suppose a random variable $X$ follows a normal distribution $X \sim N(0, \sigma^2)$, the variance of ReLU($X$) is $1 - 2c/\pi$.

2In the original ResNet case, there is a ReLU layer before $x_{t+1}$. However, [He et al., 2016b] observed that the ReLU layer does not truncate $x_{t+1}$ frequently after some training. Thus our derivation still works for this case.
3.2 Pre/Post-activation in Residual Blocks

Identity mappings are very important for training very deep neural networks [He et al., 2016b]. The original ResNet has an activation layer ReLU between two residual blocks, thus has a depth limit. To solve the problem, the simplest method is to put the ReLU layer before residual connections. However, this leads to a non-negative output from the residual branches, leading to a worse performance. The second version of ResNet changes the order of the layers so that the last layer is convolutional layers or BN layers (Figure 2 b & c).

These two versions preserve identity mappings. However, we argue that Figure 2 b & c are not the optimal solutions:

- There should be a BN layer after the last convolutional layer and before the residual connection. The distribution of feature maps after convolutional layers may change frequently after training. Directly adding the unnormalized feature maps to the main branch may slow down training.

- There should be an activation layer after the last convolutional layer (but not before the first convolutional layer). Having a ReLU activation before the first convolutional layer would lose half information for residual learning, but having a ReLU activation after the last convolutional would select useful features to contribution to the main branch.

According to the two reasons stated, post-activation is more effective than pre-activation as long as identity mappings can be preserved.

**PMResNet.** We make a small change to Figure 2 (d) by using plus and minus alternatively. Odd residual blocks are the same as Figure 2 d: \( x_{t+1} = x_t + F(x_t; \Theta_t) \), but even residual blocks use “minus”: \( x_{t+1} = x_t - F(x_t; \Theta_t) \). Figure 2 e shows our proposed Plus-Minus ResNet.

The PM architecture is only meaningful when the last layer in the residual branch is a ReLU layer which only outputs positive values. If the last layer is a convolutional layer or BN layer, it’s equivalent to Figure 2 b & c. By using the PM architecture, both identical mappings and post-activation can be preserved. The networks are also flexible to learn \( f(x) = x \) especially when the networks have many residual blocks.

Here we summarize the difference between our PMResNet and the original ResNet:

- The original residual block (Figure 2 a) is replaced by the PM-architecture (Figure 2 e) which uses plus and minus alternatively.

- Instead of the default initialization or the “zero-\( \gamma \)” initialization, we use \( 1/\sqrt{T} \) Initialization for the weights of all last BN layers in the residual branches.
• Add one BN layer (called global BN) after the last the residual block and before the global pooling layer.

4 Experiments

4.1 Ablation Studies on CIFAR Datasets

The CIFAR10 dataset consists of 60,000 colour images of size 32 × 32 pixels and 10 classes. The official version of the dataset is split into a training set with 50,000 images and a test set with 10,000 images. The CIFAR 100 dataset is similar to CIFAR10 dataset but has 100 classes. We conduct image classification experiments on the two datasets.

We follow the setting from [He et al., 2016a]. The network inputs are 32 × 32 and normalized using per-channel mean and standard deviation. The data augmentation methods are as follows: first zero-pad the images with 4 pixels on each side to obtain a 40 × 40 pixel image, then randomly crop a 32 × 32 pixel image, and finally mirror the images horizontally with 50% probability. For all of these experiments, we use the same optimizer: training for 64k iterations with batches of 128 images using stochastic gradient descent, momentum of 0.9, and weight decay of 5e-4. We start with a learning rate of 0.1, divide it by 10 at 90, 135 and 170 epochs, and terminate training at 180 epochs. Each experiment is repeated 5 times and we report the average of the 5 runs.

![Figure 4: Input variance for each block with default initialization and our $1/\sqrt{T}$ initialization.](image)

Visualization of the Variance Problem To visualize the variance explosion problem in ResNet, we conduct an experiment on the ResNet1202 network proposed in [He et al., 2016a]. The network consists of three stages and each stage is a stack of 200 residual blocks. We compute the input variance of each residual blocks: we compute the variance of each channel and report the average variance among channels. We test on three architectures: ResNet, Pre-Activation ResNet and PMResNet (Figure 2 a b and e). We first initialize the weights of convolution layers with He initialization and initialize the weights of BN layers with constant 1. We also use our $1/\sqrt{k}$ initialization for the last BN layer of each residual block, and repeat the experiments.

Figure 4 shows the visualization results. Results are averaged over 5 runs. The results are consistent with our analysis.

The input variance is almost a linear function of the block index\(^3\). The coefficient for Pre-Activation ResNet is approximately 1, the coefficient for ResNet is smaller than 1 (since there is a ReLU layer after each residual block), and the coefficient for PMResNet is smaller than 0.5 (since there is a ReLU layer after each residual branch).

Using the default initialization, the variance of the last block’s input is over 100 for ResNet and Pre-Activation ResNet, leading to optimization difficulties. By using our initialization method, the input variances increase much slower and the variance of the last block’s input is smaller than 2. Thus the problem can be solved.

Convergence at Increasing Depth We show that our proposed PMResNet has very good convergence properties, especially the network is very deep. Figure 3 shows the training curves of training 164-layer networks on the CIFAR10 and CIFAR100 datasets using four architectures: original ResNet (Figure 2 a), pre-activation ResNet (Figure 2 b), ReLU only pre-activation ResNet (Figure 2 c) and our proposed PMResNet. We can find that our proposed PMResNet (red line) converges much faster than other architectures and there are significant margins in the last stage of the curves.

We further evaluate how each architectures affects training very deep nets by measuring the test accuracy after the first epoch as we increase depth [Zhang et al., 2019]. We use the default training settings mentioned above. Figure 1 shows the test accuracy at the first epoch as depth increases. The figure shows that our proposed PMResNet has no optimization difficulty when the network is extremely deep, however other architectures would be more difficult to optimize when the network is very deep.

Performance of Different Architectures We report the final performance of different architectures. We test the original ResNet, Pre-Activation ResNet, ReLU only Pre-Activation ResNet (PreResNet V2) and our PMResNet on 110-layer, 164-layer and 1202-layer network. Unless otherwise stated, we use the BasicBlock [He et al., 2016a] as the backbone. Since the networks are deeper than usual networks, we add a spatial Dropout layer before the second convolution layer in the residual block, with a drop rate 0.05 for 110-layer and 0.1 for 164 and 1202-layer networks. As we analysis earlier, the global BN layer (the BN layer after all residual blocks and before global pooling layer) is important for training very deep residual network since the variance problem. We test the difference of adding or removing such global BN layer. Note that the original Pre-Activation ResNet has the global BN layer while the original version of other three architectures do not have the global BN layer.

Table 1 and 2 show the results of the final performance. For each run, we record the average validation accuracy of the last 10 epochs. Each experiment is repeated 5 times and we report the average of the 5 runs. The two tables show consistent improvement of the global BN layer. Further, we can find that PMResNet performs the best and the original ResNet is the worst when the network is very deep. The

\(^3\)For ease of visualization, we use the log of the y-axis so that the two initialization methods can be plotted in one figure. If we directly plot the figure, we can easily find it’s a linear function.
two versions of Pre-Activation ResNet have similar performance. Without the global BN layer, PreResNet V2 is even better. These results contradict the conclusions in [He et al., 2016b], but follow our conclusions: the two version of Pre-Activation ResNet have no significant difference. The reason why the results of PreResNet V2 in [He et al., 2016b] are not better is the absence of the global BN layer, which can mitigate the variance explosion problem. The original ResNet does not perform well since it cannot preserve identity mappings. Our PMResNet performs the best since it can preserve identity mappings and post-activation in the same time. Post-activation can make the training more stable than pre-activation does.

<table>
<thead>
<tr>
<th>CIFAR10</th>
<th>110 layers w/o BN</th>
<th>110 layers w/o BN</th>
<th>164 layers w/o BN</th>
<th>164 layers +Bottleneck w/o BN</th>
<th>1202 layers w/o BN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>Fig 2 a</td>
<td>94.202</td>
<td>94.211</td>
<td>94.172</td>
<td>94.346</td>
</tr>
<tr>
<td>PreResNet</td>
<td>Fig 2 b</td>
<td>26.650</td>
<td>94.240</td>
<td>NA</td>
<td>94.352</td>
</tr>
<tr>
<td>PreResNet V2</td>
<td>Fig 2 c</td>
<td>94.072</td>
<td>94.284</td>
<td>94.245</td>
<td>94.251</td>
</tr>
<tr>
<td>PMResNet</td>
<td>Fig 2 e</td>
<td>94.209</td>
<td>94.486</td>
<td>94.282</td>
<td>94.405</td>
</tr>
</tbody>
</table>

Table 1: Top 1 accuracy results on CIFAR10 Datasets. The option “BN” refers to the global BN layer.

Table 2: Top 1 accuracy results on CIFAR100 Datasets. The option “BN” refers to the global BN layer.

<table>
<thead>
<tr>
<th>CIFAR100</th>
<th>110 layers w/o BN</th>
<th>110 layers w/o BN</th>
<th>164 layers w/o BN</th>
<th>164 layers +Bottleneck w/o BN</th>
<th>1202 layers w/o BN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>Fig 2 a</td>
<td>74.551</td>
<td>74.699</td>
<td>75.086</td>
<td>75.503</td>
</tr>
<tr>
<td>PreResNet</td>
<td>Fig 2 b</td>
<td>NA</td>
<td>74.690</td>
<td>NA</td>
<td>75.601</td>
</tr>
<tr>
<td>PreResNet V2</td>
<td>Fig 2 c</td>
<td>74.653</td>
<td>74.983</td>
<td>75.181</td>
<td>75.669</td>
</tr>
<tr>
<td>PMResNet</td>
<td>Fig 2 e</td>
<td>75.200</td>
<td>75.299</td>
<td>75.877</td>
<td>76.126</td>
</tr>
</tbody>
</table>

4.2 ImageNet Datasets

The ILSVRC classification dataset contains 1000 classes. There are about 1.2 million images for training, and 50000 for validation. We use the same data argumentation as [He et al., 2016a] in in for training. For evaluation, we only use the single-crop with input image size of 224. We use the same optimizer for all of these experiments: training for 120 epochs with batches of 1024 images using SGD, momentum of 0.9, and weight decay of 1e-4. We start with a learning rate of 0.4, divide it by 10 at 30, 60 and 90 epochs.

Table 3 shows comparison of the original ResNet, PreResNet and the proposed PMResNet on 101/152 layers networks. PMResNet improves the absolute top-1 accuracy of ResNet-101 on ImageNet about 0.91%. In contrast, the performance improvement of PreResNet is only about 0.22%. For ResNet-152, we achieve 77.62% top-1 accuracy outperforming the original one about 0.62%, and PreResNet is 0.25% better than original model.

Table 3: Top 1 accuracy on ImageNet dataset.

<table>
<thead>
<tr>
<th></th>
<th>ResNet</th>
<th>PreResNet</th>
<th>PMResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>101 layers</td>
<td>76.30</td>
<td>76.52</td>
<td>77.21</td>
</tr>
<tr>
<td>152 layers</td>
<td>77.00</td>
<td>77.25</td>
<td>77.62</td>
</tr>
</tbody>
</table>

5 Conclusion

In this paper, we revisit the problem of training very deep ResNet. By analysing of the variance of the middle layer output, and studying the differences between the pre-activation and the original structure, we found there key points to improve the performance of the residual neural networks: 1) Plus-Minus block can preserve post-activation structure while ensuring identity mapping; 2) $1/\sqrt{T}$ initialization will effectively reduce the variance of the model output; 3) using global BN layer. Our method improves the performance of the original ResNet and the PreResNet. Improvement about 0.3% was achieved on CIFAR-10, and 0.6-0.8% on CIFAR-100. Our approach also leads 0.7% and 0.4% top-1 accuracy improvement on the large-scale dataset ImageNet compared to PreResNet-101/152.
References


