Identity Mappings in Deep Residual Networks

1 Citation

He, Kaiming, et al. "Identity mappings in deep residual networks." European conference on computer vision. Springer, Cham, 2016.

`https://arxiv.org/pdf/1603.05027v2.pdf`

2 Abstract

We create a new and improved residual unit that makes training faster and generalization better.

3 Introduction

Recall that a residual unit takes the form:

$$\mathbf{y}_l = h(\mathbf{x}_l) + F(\mathbf{x}_l, W_l)$$
$$\mathbf{x}_{l+1} = f(\mathbf{y}_l)$$

Typically, we set h = identity mapping and f = ReLU. We find that if you set both h and f to the identity mapping, then it's easy to propagate loss from any deeper unit to any shallower unit. This makes training much faster.

Ultimately, we propose a different residual unit. For the original unit, $\mathbf{x}_l$ takes two paths to the element-wise addition: (1) direct, (2) weight, batch normalization (BN), ReLU, weight, BN. Then, after the addition, we apply another ReLU. We propose a NEW unit where $\mathbf{x}_l$ takes the following paths to the element-wise addition: (1) direct, (2) BN, ReLU, weight, BN, ReLU, weight. Then, we do NOT do any ReLU after the addition and feed into the next part of the network. Thus, we have set h and f to be identity functions and we have inserted ReLU before weights rather than after.

4 Analysis of Deep Residual Networks

Setting h and f to the identity gives:

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} F(\mathbf{x}_i, W_i)$$

Let's compute the gradient of the loss function (ξ)

$$\frac{\partial \xi}{\partial \mathbf{x}_l} = \frac{\partial \xi}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l}$$

$$\frac{\partial \xi}{\partial \mathbf{x}_l} = \frac{\partial \xi}{\partial \mathbf{x}_L}(1 + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} F(\mathbf{x}_i, W_i))$$

Notice that the 1 in the second term of the product protects the gradient from vanishing because it's unlikely that the summation will always be -1. Thus, setting h and f yields a gradient that can be directly propagated between units and is likely to be healthy.

5 On the Importance of Identity Skip Connections

Suppose we instead set $h(\mathbf{x}) = \lambda \mathbf{x}$ for constant λ. The 1 in the parenthesis of the gradient becomes a product that can very easily decay to zero. Thus, we don't get any nice properties of the gradient.

Thus, any change to h or f can make us lose the good gradient properties. We verify this with experiments where we try different choices of h and keep f = ReLU. We train a ResNet-110 (54 residual units) on CIFAR-10.

First, we try $h(\mathbf{x}) = \lambda \mathbf{x}$. It does not converge well and gets high test error (12.35%).

Second, we try a gating function (like in Highway Networks). Here, we compute $g(\mathbf{x}) = \sigma(W_g \mathbf{x} + b_g)$. We then scale the F path by $g(\mathbf{x})$ and the shortcut path by $1 - g(\mathbf{x})$. We use grid search to pick the best b_g (it is -6). This is better than the $h(\mathbf{x}) = \lambda \mathbf{x}$ approach from before (8.35%), but still can't beat the baseline (6.61%).

Third, we try the gating approach again, but we don't scale the F path. This gives 12.86% test error - not good. However, if we use a heavy negative bias like $b_g = -6$, then $1 - g(\mathbf{x}) \approx 1$ and we find that performance is good at 6.91%. This is another indicator that the identity is the best choice.

Fourth, we tried a 1×1 convolution on the shortcut. It's bad (12.22%).

Fifth, we tried dropout. The network fails to converge.

Thus, the experiments above show that, even if add more parameters as in gating and 1×1 convolution, h = identity is still best.

6 Experiments on Activation

Here, we experiment with ResNet-110 and ResNet-164 (a residual unit where, instead of two 3×3 convolutions, we have just one 3×3 convolution that is preceeded by a 1×1 convolution to reduce dimension and succeeded by a 1×1 convolution to increase dimension). ResNet-164 achieves 5.93% test error on CIFAR-10.

First, we setting f into BN followed by ReLU (instead of just ReLU). Test error rises.

Second, we try removing the ReLU and moving it before the addition on the F path. This causes the F path to only contribute nonnegative values to the addition, which hurts test error.

Third, we try using a pre-activation. Assume that h = identity and notice that we have:

$$\mathbf{y}_{l+1} = f(\mathbf{y}_l) + F(f(\mathbf{y}_l), W_l)$$

We try changing this to:

$$\mathbf{y}_{l+1} = \mathbf{y}_l + F(\hat{f}(\mathbf{y}_l), W_l)$$

Thus, we only put activation on the F path. With this new idea, we consider both ReLU-only preactivation (i.e. the F path is ReLU, weight, BN, ReLU, weight, BN) and full preactivation (i.e. BN, ReLU, weight, BN, ReLU, weight). The former does poorly on test error but the latter does great, even beating the baseline. Hence, we include the full preactivation in our final proposed residual unit design.

The preactivation helps because it turns f into an identity and adds another batch normalization to the F path to improve regularization. We test this with a 1001 layer ResNet. With the old approach, the loss decreases slowly, but with the new approach, loss decreases quickly.

We find that the benefit of the new residual unit shows more when the ResNet is deeper. For example, when training on ImageNet, a ResNet-152 improves its top-1 error by 0.2% (from 21.3% to 21.1%) when using the new residual unit. The ResNet-200, on the other hand, improves 1.1% (from 21.8% to 20.7%).

Note that Inception-ResNet-v2 (from another paper) gets a top-5 error of 19.9%.

7 Conclusions

Set h = identity, f = identity, and use full preactivation on the F path. This yields ResNets that train faster and generalize better.