Introduction

In recent years Deep Convolutional Neural Networks (CNN) demonstrated a high performance on image classification tasks. Experiments showed that the number of layers (depth) in a CNN is correlated to the performance in image recognition tasks. This led to the idea that deeper networks should perform better. Creating deep networks is not as simple as adding layers. One problem is the vanishing/exploding gradients, which hamper the convergence. This obstacle can be overcome by normalized initialization and intermediate normalization layers, so that networks start converging for stochastic gradient descend (SGD) using the backpropagation algorithm. Another problem is the degradation, if the depth of a network increases, the accuracy gets saturated and then degrades rapidly. (Figure 1) A way to counter the degradation problem is using residual learning. (1)

Deep Residual Learning

Residual Learning

It is possible to fit an desired underlying mapping $H(x)$ by a few stacked nonlinear layers, so they can also fit an another underlying mapping $F(x) = H(x) - x$. As a result, it is possible to reformulate it to $H(x) = F(x) + x$, which consists of the Residual Function $F(x)$ and input $x$. The connection of the input to the output is called a skip connection or identity mapping. The general idea is that if multiple nonlinear layers can approximate the complicated function $H(x)$, then it is possible for them to approximate the residual function $F(x)$. Therefore the stacked layers are not used to fit $H(x)$, instead these layers approximate the residual function $F(x)$. Both forms should be able to fit the underlying mapping.

One reason for the degradation problem could be the difficulties in approximating identity mappings by nonlinear layers. The reformulation used identity mapping as a reference and let the residual function represent the perturbations. The identity mapping can be generated by the solver through driving the weights of the residual function to zero if need be. (1)

Implementation
Residual learning is implemented to every few stacked layers. Figure 2 shows an example of 2 layers. As an example, formulation (1) can be defined as:

\[ F(x) = W_2 \sigma(W_1 x) + x \]

Where \( W_1 \) and \( W_2 \) are the weights for the convolutional layers and \( \sigma \) is the activation function, in this case a RELU function. The operation \( F + x \) is realized by a shortcut connection and element-wise addition. The addition is followed by an activation function \( \sigma \).

The resulting formulation for a residual block is:

\[ y(x) = \sigma(W_2 \sigma(W_1 x) + x) . \]

After each convolution (weight) layer a batch normalization method (BN) is adopted. The training of the network is achieved by stochastic gradient descent (SGD) with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus. The weight decay rate is 0.0001 and has a value of 0.9. (1)

**Network Architectures**

**Plain Networks**

The plain networks are adopted from the VGG nets (Figure 3(left)). The convolutional layers have mostly 3x3 filters and the design follows two rules:

1. For the same output feature map size, the layers have the same number of filters, and
2. if the feature map size is halved, the number of filters is doubled in order to preserve the time complexity per layer.

The downsampling operation is performed by the convolutional layers that have a stride of 2, hence no pooling layers. The network ends with a global average pooling layer and a 1000-way fully connected layer with softmax function.

Figure 3 (middle) shows a plain model with 34 layers. (1)

**Residual Network**

To convert the plain model to the residual version, shortcut connections are added, as demonstrated in the figure 3 (right). The solid line shortcuts are identity mapping. When the dimensions increase there are 2 options (dotted line shortcut):

1. The shortcut still performs identity mapping with zero padding to increasing the dimensions or
2. the shortcut is used to match dimensions utilizing 1x1 convolution.

In both options, when the shortcut go across feature maps of different sizes, they used a stride of 2. Generally the second option is used. (1)

**Results**

![Figure 3. Example network architectures for ImageNet. Left: the VGG-19 model as a reference. Middle: a plain network with 34 layers. Right: a residual network with 34 layers. The dotted shortcuts increases dimensions](image-url)
The residual and plain networks are compared on the ImageNet 2012 classification dataset that consist of 1000 classes. All Networks are trained on 1.28 million training images and evaluated on the 50k validation images. The final result was obtained on the 100k test images.

The evaluation of the plain models showed that the 34 layer network has a higher training error than the 18 layer model. (Figure 4 left) The reason behind this result is the degradation problem.

The residual models show a contradictory result. The deeper ResNet with 34 layers has a smaller training error than the 18 layer (Figure 4 right). This result proves that the degradation problem can be addressed with residual learning and that an increased network depth results in a gain accuracy. (1)

Optimization

The original design of the residual block (Figure 1) can be represented in a more detailed way (Figure 5 a). A proposed optimization is shown in Figure 5 b. The proposed design consist of a direct path for the propagating information through the residual block as a result, through the entire network. This allows the signal to propagate from one block to any other block, during both forward and backward passes. The complexity of the training also becomes simpler with the new block design.

The original Residual block is described as \( y(x) = \sigma(x + F(x)) \). The new design change the representation to \( y(x) = x + F(x) \), because it is important to have a "clean" path from one block to the next one. The removal of the ReLU function on the main path and the different design of the Residual function \( F(x) \) is related (Figure 6).

![Figure 4. Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.](image)

![Figure 5. (a) original Residual Unit design; (b) proposed Residual Unit. The grey arrows indicate the easiest paths for the information to propagate.](image)
Figure 6 shows that the activation function from the main path is moved to the residual function of the next block. This means that the activation functions (ReLU) are now a "pre-activation" function of the weight layers. Experiments showed that the ReLU-only pre-activation performs similarly to the original design. By adding BN to the pre-activation, the result can be improved by a healthy margin. This "pre-activation" model shows consistently better results than the original counterpart (Table 1) and the computational complexity is linear to the depth of the network.

<table>
<thead>
<tr>
<th>dataset</th>
<th>network</th>
<th>baseline unit</th>
<th>pre-activation unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>ResNet-110 (1layer skip)</td>
<td>9.90</td>
<td>8.91</td>
</tr>
<tr>
<td></td>
<td>ResNet-110</td>
<td>6.61</td>
<td>6.37</td>
</tr>
<tr>
<td></td>
<td>ResNet-164</td>
<td>5.93</td>
<td>5.46</td>
</tr>
<tr>
<td></td>
<td>ResNet-1001</td>
<td>7.61</td>
<td>4.92</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>ResNet-164</td>
<td>25.16</td>
<td>24.33</td>
</tr>
<tr>
<td></td>
<td>ResNet-1001</td>
<td>27.82</td>
<td>22.71</td>
</tr>
</tbody>
</table>

Table 1: Classification error (%) on the CIFAR-10/100 test set using the original Residual units and the pre-activation Residual units.

**Literature**


**Weblinks**

Links to related/additional Content in the Web