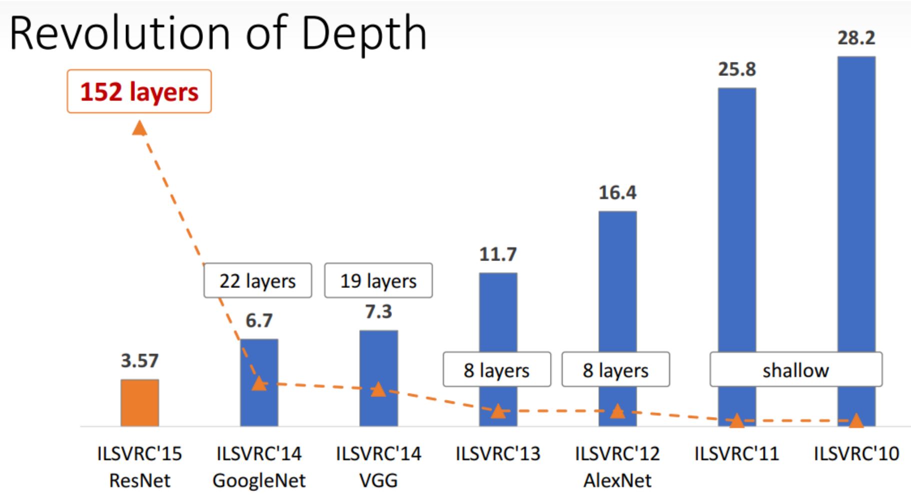


You must know the CNN model: ResNet

Little General
Excellent answerer on the topic of artificial intelligence

Welcome to exchange and reprint, the article will be published simultaneously on the official account: Full-stack Engineer of Machine Learning Algorithms (Jeemy110)

introduction

The introduction of the Deep Residual Network (ResNet) is a milestone in the history of CNN images. Let's first look at ResNet's performance in ILSVRC and COCO 2015:

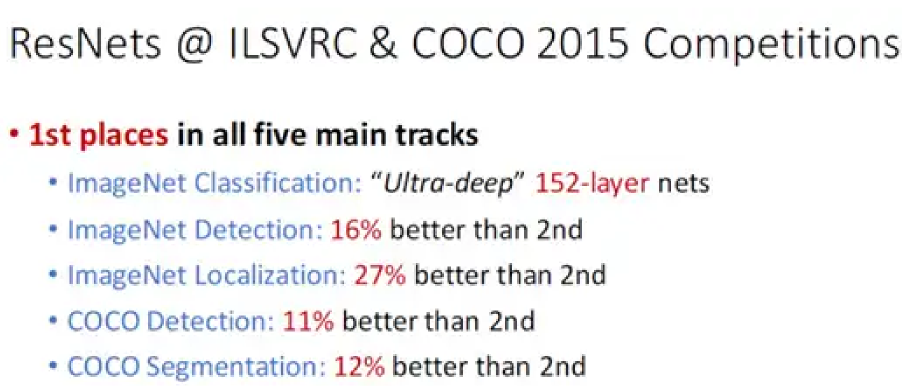


Figure 1 ResNet's performance on ILSVRC and COCO 2015

ResNet achieved 5 firsts and once again broke the record of CNN models on ImageNet:

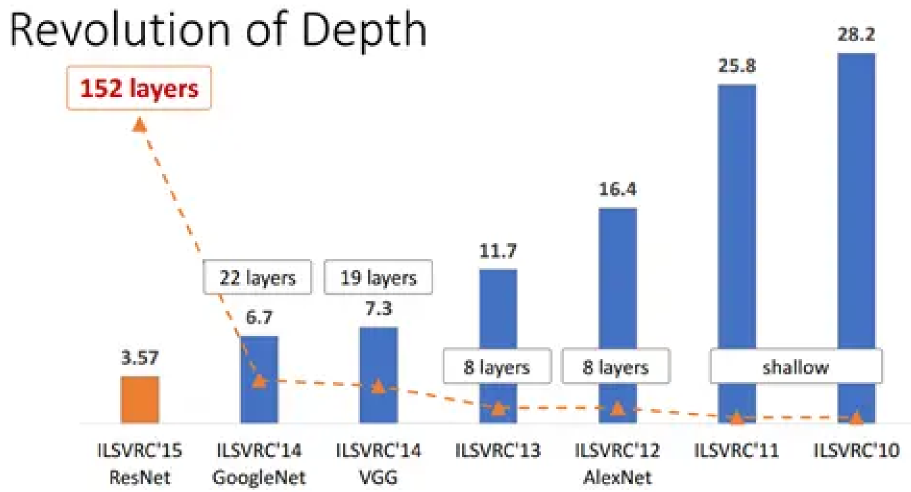


Figure 2 ImageNet classification Top-5 error

ResNet's author He Kaiming also won the CVPR2016 Best Paper Award. Of course, Dr. He's achievements are far more than that. If you are interested, you can search for his later brilliant achievements. So why does ResNet have such excellent performance? In fact, ResNet solves the problem of difficult training of deep CNN models. From Figure 2, you can see that VGG in 2014 has only 19 layers, while ResNet in 2015 has as many as 152 layers. This is completely different in network depth. So if you look at this picture at first glance, you will definitely think that ResNet wins by depth. This is of course the case, but ResNet also has an architectural trick that makes the depth of the network play a role. This trick is residual learning. The following is a detailed description of the theory and implementation of ResNet.

The degradation problem of deep networks

From experience, the depth of the network is crucial to the performance of the model. When the number of network layers is increased, the network can extract more complex feature patterns, so when the model is deeper, better results can be achieved in theory. Figure 2 also shows a practical evidence that the deeper the network, the better the effect. But will the performance of a deeper network be better? Experiments have found that deep networks have a degradation problem: when the network depth increases, the network accuracy becomes saturated or even decreases. This phenomenon can be seen intuitively in Figure 3: the 56-layer network is worse than the 20-layer network. This is not an overfitting problem, because the training error of the 56-layer network is also high. We know that deep networks have the problem of gradient disappearance or explosion, which makes deep learning models difficult to train. But now there are some technical means such as BatchNorm to alleviate this problem. Therefore, the degradation problem of deep networks is very surprising.

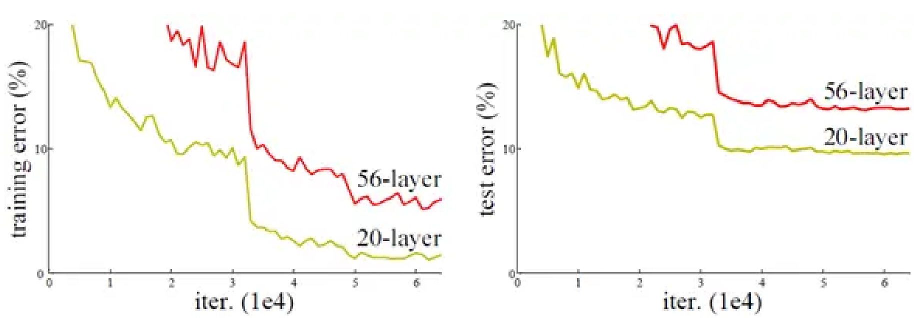


Figure 3 Errors of 20-layer and 56-layer networks on CIFAR-10

Residual Learning

The degradation problem of deep networks at least shows that deep networks are not easy to train. But let's consider the fact that now you have a shallow network, and you want to build a deep network by stacking new layers upwards. An extreme case is that these added layers do not learn anything, but just copy the features of the shallow network, that is, the new layers are identity mapping. In this case, the deep network should at least perform as well as the shallow network, and there should be no degradation. Well, you have to admit that there must be something wrong with the current training method, which makes it difficult to find a good parameter for the deep network.

This interesting hypothesis inspired Dr. He to come up with residual learning to solve the degradation problem.x , the learned features are recorded as$H(x)$, now we hope that it can learn the residual$F(x) = H(x) - x$, so the original learning feature is$F(x) + x$. This is because residual learning is easier than learning the original features directly. When the residual is 0, the stacking layer only does an identity mapping, at least the network performance will not decrease. In fact, the residual will not be 0, which will also enable the stacking layer to learn new features based on the input features, thereby achieving better performance. The structure of residual learning is shown in Figure 4. This is a bit like a "short circuit" in a circuit, so it is a shortcut connection.

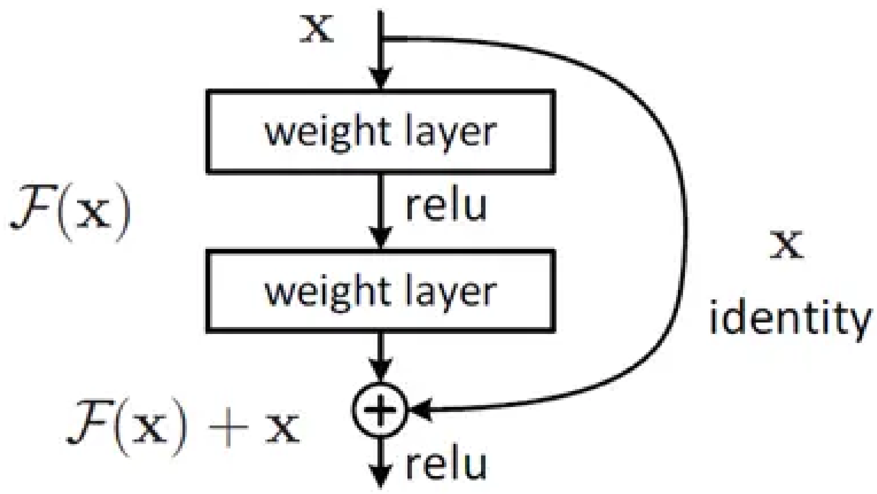


Figure 4 Residual learning unit

Why is residual learning relatively easier? Intuitively, residual learning requires less content to learn, because the residual is generally smaller and less difficult to learn. However, we can analyze this problem from a mathematical perspective. First, the residual unit can be expressed as:

$$y_l = h(x_l) + F(x_l, W_l)$$
$$x_{l+1} = f(y_l)$$

inx_l andx_{l+1} respectively represent the$l$ residual units. Note that each residual unit generally contains multiple layers.F is the residual function, which represents the learned residual, and $h(x_l) = x_l$ represents the identity mapping,f is the ReLU activation function. Based on the above formula, we get$l$ to the deep layer$L$ are:

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, W_i)$$

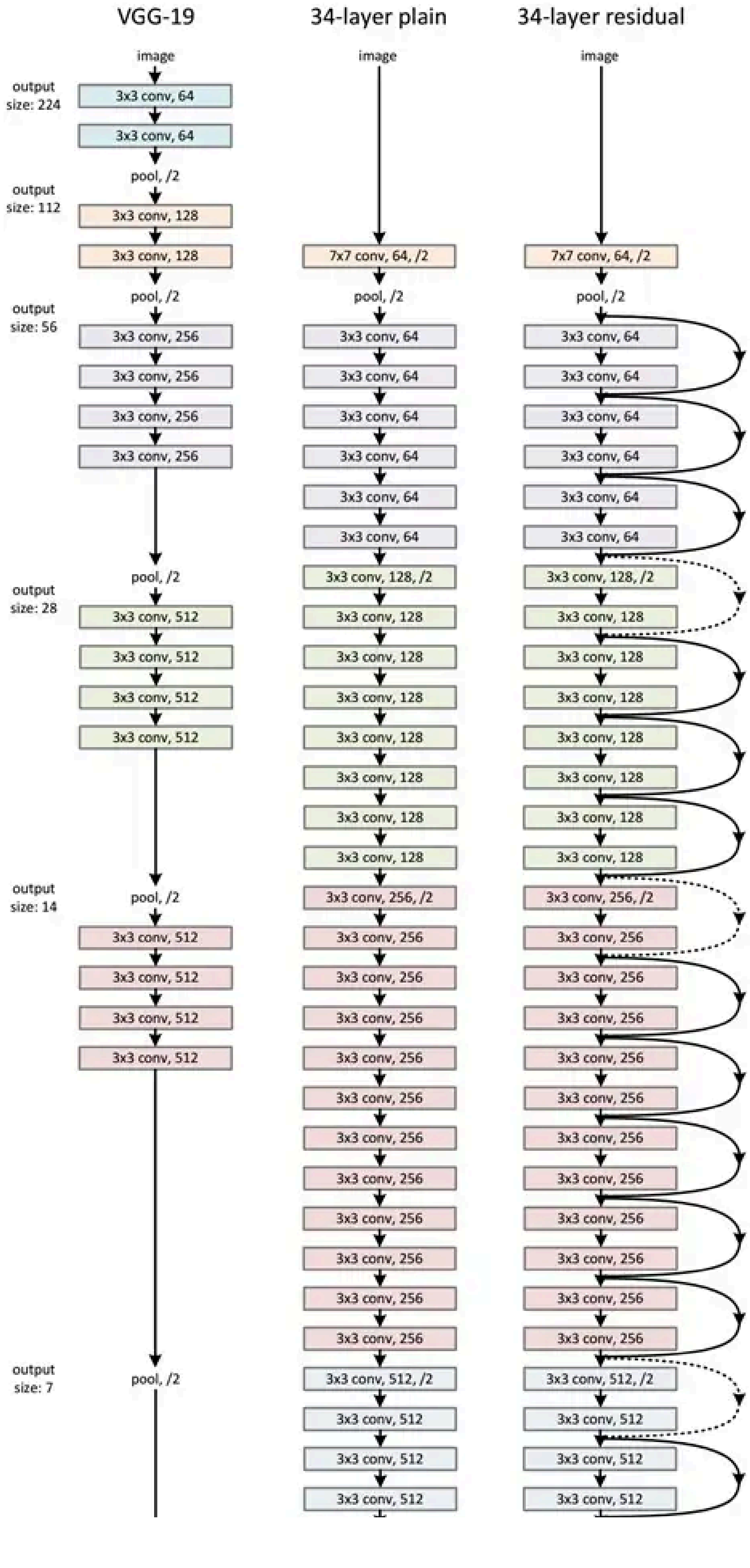
Using the chain rule, we can find the gradient of the reverse process:

$$\frac{\partial loss}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \left(1 + \frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} F(x_i, W_i)\right)$$

The first factor of the formula $\frac{\partial loss}{\partial x_L}$ The loss function represented by \frac{\partial loss}{\partial {{x}_{L}}} arrives at L indicates that the short-circuit mechanism can propagate the gradient losslessly, while the other residual gradient needs to pass through the layer with weights, and the gradient is not directly transmitted. The residual gradient will not be all -1 by chance, and even if it is relatively small, the existence of 1 will not cause the gradient to disappear. Therefore, residual learning will be easier. Please note that the above derivation is not a strict proof.

ResNet network structure

The ResNet network refers to the VGG19 network, and is modified on its basis, and residual units are added through a short-circuit mechanism, as shown in Figure 5. The changes are mainly reflected in that ResNet directly uses stride=2 convolution for downsampling, and replaces the fully connected layer with a global average pool layer. An important design principle of ResNet is that when the feature map size is reduced by half, the number of feature maps is doubled, which maintains the complexity of the network layer. As can be seen from Figure 5, compared with ordinary networks, ResNet adds a short-circuit mechanism between every two layers, which forms residual learning, and the dotted line indicates that the number of feature maps has changed. The 34-layer ResNet shown in Figure 5 can also build a deeper network as shown in Table 1. From the table, we can see that for 18-layer and 34-layer ResNet, residual learning is performed between two layers. When the network is deeper, residual learning is performed between three layers. The convolution kernels of the three layers are 1x1, 3x3 and 1x1 respectively. It is worth noting that the number of feature maps in the hidden layer is relatively small and is 1/4 of the number of output feature maps.


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512

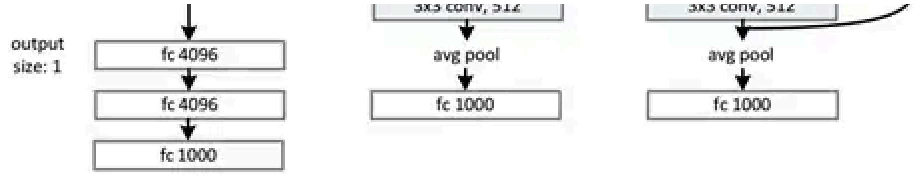


Figure 5 ResNet network structure diagram

Table 1 ResNet with different depths

Next, let's analyze the residual unit. ResNet uses two types of residual units, as shown in Figure 6. The left figure corresponds to a shallow network, while the right figure corresponds to a deep network. For short-circuit connections, when the input and output dimensions are the same, the input can be directly added to the output. However, when the dimensions are inconsistent (corresponding to doubling the dimension), they cannot be added directly. There are two strategies: (1) Use zero-padding to increase the dimension. In this case, you usually need to do a downsamp first. You can use pooling with strde=2, which will not increase the parameters; (2) Use a new mapping (projection shortcut), generally using 1x1 convolution, which will increase the parameters and the amount of calculation. In addition to directly using the identity mapping, short-circuit connections can of course use projection shortcuts.

Figure 6 Different residual units

The author compares the network effects of 18-layer and 34-layer, as shown in Figure 7. It can be seen that the ordinary network has degradation, but ResNet solves the degradation problem very well.

Figure 7 Network effects of 18-layer and 34-layer

Finally, the comparison results of ResNet network and other networks on ImageNet are shown in Table 2. It can be seen that the error of ResNet-152 is reduced to 4.49%, and when the integrated model is used, the error can be reduced to 3.57%.

Table 2 Comparison results of ResNet and other networks

Let me say a little about residual units. We have mentioned several ways to deal with short-circuit connections above. In fact, the author has conducted detailed analysis and experiments on different residual units in the literature [2] . Here we directly present the optimal residual structure, as shown in Figure 8. An obvious change before and after the improvement is the use of pre-activation, with BN and ReLU both advanced. In addition, the author recommends using identity transformation for short-circuit connections, which ensures that there will be no obstruction in short-circuit connections. If you are interested, you can read this article.

Figure 8 Improved residual unit and its effect

TensorFlow implementation of ResNet

Here is the TensorFlow implementation of ResNet50. The implementation of the model refers to the implementation of the Caffe version . The core code is as follows:

```
                 scope="resnet50"):
        self.inputs =inputs
        self.is_training = is_training
        self.num_classes = num_classes

        with tf.variable_scope(scope):
            # construct the model
            net = conv2d(inputs, 64, 7, 2, scope="conv1") # -> [batch, 112, 112
            net = tf.nn.relu(batch_norm(net, is_training=self.is_training, scop
            net = max_pool(net, 3, 2, scope="maxpool1")  # -> [batch, 56, 56, 6
            net = self._block(net, 256, 3, init_stride=1, is_training=self.is_t
                              scope="block2")          # -> [batch, 56, 56, 25
            net = self._block(net, 512, 4, is_training=self.is_training, scope=
                                                        # -> [batch, 28, 28, 51
            net = self._block(net, 1024, 6, is_training=self.is_training, scope
                                                        # -> [batch, 14, 14, 10
            net = self._block(net, 2048, 3, is_training=self.is_training, scope
                                                        # -> [batch, 7, 7, 2048
            net = avg_pool(net, 7, scope="avgpool5")    # -> [batch, 1, 1, 2048
            net = tf.squeeze(net, [1, 2], name="SpatialSqueeze") # -> [batch, 2
            self.logits = fc(net, self.num_classes, "fc6")       # -> [batch, n
            self.predictions = tf.nn.softmax(self.logits)


    def _block(self, x, n_out, n, init_stride=2, is_training=True, scope="block
        with tf.variable_scope(scope):
            h_out = n_out // 4
            out = self._bottleneck(x, h_out, n_out, stride=init_stride,
                                   is_training=is_training, scope="bottlencek1"
            for i in range(1, n):
                out = self._bottleneck(out, h_out, n_out, is_training=is_traini
                                       scope=("bottlencek%s" % (i + 1)))
            return out

    def _bottleneck(self, x, h_out, n_out, stride=None, is_training=True, scope
        """ A residual bottleneck unit"""
        n_in = x.get_shape()[-1]
        if stride is None:
            stride = 1 if n_in == n_out else 2

        with tf.variable_scope(scope):
            h = conv2d(x, h_out, 1, stride=stride, scope="conv_1")
            h = batch_norm(h, is_training=is_training, scope="bn_1")
            h = tf.nn.relu(h)
            h = conv2d(h, h_out, 3, stride=1, scope="conv_2")
            h = batch_norm(h, is_training=is_training, scope="bn_2")
            h = tf.nn.relu(h)
            h = conv2d(h, n_out, 1, stride=1, scope="conv_3")
            h = batch_norm(h, is_training=is_training, scope="bn_3")

            if n_in != n_out:
                shortcut = conv2d(x, n_out, 1, stride=stride, scope="conv_4")
                shortcut = batch_norm(shortcut, is_training=is_training, scope=
            else:
                shortcut = x
            return tf.nn.relu(shortcut + h)
```

The full implementation can be found on GitHub .

ResNet solves the degradation problem of deep networks through residual learning, allowing us to train deeper networks. This can be regarded as a historical breakthrough in deep networks. Perhaps there will be a better way to train deeper networks soon, let us look forward to it!

References

1. Deep Residual Learning for Image Recognition .
2. Identity Mappings in Deep Residual Networks .
3. Go worship the great God .

Welcome to exchange and reprint, the article will be published simultaneously on the official account: Full-stack Engineer of Machine Learning Algorithms (Jeemy110)

Edited on 2022-03-09 21:51

"Sincere appreciation leaves a lasting fragrance on your hands"

Appreciation

No one has appreciated it yet, come and be the first one to appreciate it!

Convolutional Neural Networks (CNNs) Deep Learning Computer Vision

Speak rationally and interact in a friendly manner

77 Comments default up to date

Mary Shen
He Kaiming is really awesome
2018-06-04 reply 81

Chen Lei
At that time, his paper on denoising guided images was also a great one. Now, all the one-click cutouts use this theory. The key is to read his paper and feel some of his thinking patterns and style. He just likes to compare things.😓
2021-09-14 reply 32

Lera
He is the top scorer in science in Guangdong Province. His IQ is far superior to ours. 😳
2023-03-21 reply 2

View all 6 replies ›

pudding
I hope to make the cause and effect of several mathematical formulas clear. It is better not to write mathematical formulas, which will mislead people.
2022-01-09 reply 38

Bird Nest
The formula for the chain rule seems to be wrong. The brackets should be the partial derivative of l, not L.
2018-05-30 reply 28

I've never felt this way before.

Yes, that's the problem.

2018-08-13 reply 6

Socreates

I read it for a long time but didn't find the mistake. When I scrolled down, I found that the author had already changed it.😂

2023-11-08 reply 2

Expand 3 more replies

Haoyang

It seems that the single-layer convolution of the CNN neural network has a weak ability to abstract complex objects, and a combination of more than 100 layers of networks is required to achieve human-like recognition capabilities.

However, for the recognition of handwritten digits, this kind of simple object extraction of pure shape is still sufficient, and only 3-5 layers of convolution are enough. (Refer to Lenet5)

2019-05-16 reply 20

Roland

" represents the identity mapping,

$f$is the ReLU activation function. Based on the above formula, we obtain the learning features from the shallow layer $l$to the deep layer Las follows:

$$x_L = x_l + \sum_{i-l}^{L-1} F(x_i, W_i)$$

"

Here in the article, I remember that it should be assumed in the paper

fThis formula is derived from an identity transformation... I don't know if I remember it correctly = -

2018-05-16 reply 11

Little General author

No, this f refers to the nonlinear activation of the residual unit output.

2018-05-16 reply 2

Imming ‣ **Little General**

But f doesn't seem to be reflected in the formula.

2020-10-11 reply 3

Expand 1 more reply

Chen Xiaosheng

No, I don't need to know.

2019-09-25 reply 8

Gotchaha

?

2020-11-26 reply 3

What's worth the fire?

kindness?

2021-03-17 reply 2

Expand 1 more reply

Shaun

In the text, *using the chain rule, we can find the gradient of the reverse process* The small l at the end is not the big L, right?

2019-08-11 reply 3

2022-01-09 reply like

Junziheng

INFO:2021-24-06 16:47:21:0, 'loss:', 1.2901623249053955
INFO:2021-24-06 16:47:29:0, 'test acc:', 0.5665
INFO:2021-24-06 16:51:43:1, 'loss:', 0.980273425579071
INFO:2021-24-06 16:51:51:1, 'test acc:', 0.6277
INFO:2021-24-06 16:55:57:2 , 'loss:', 0.7849510908126831
INFO:2021-24-06 16:56:05:2, 'test acc:', 0.6708
INFO:2021-24-06 17:00:17:3, 'loss:', 0.6872305274009705
INFO:2021-24-06 17:00:25:3 , 'test acc:', 0.6699
INFO:2021-24-06 17:04:38:4, 'loss:', 0.5196524858474731
INFO:2021-24-06 17:04:46:4, 'test acc:', 0.6602
INFO:2021-24-06 17:08:50:5, 'loss:', 0.3566785454750061
INFO:2021-24-06 17:08:58:5, 'test acc:', 0.67
INFO:2021-24-06 17:13:07:6, 'loss:', 0.21998071670532227
INFO:2021-24- 06 17:13:15:6, 'test acc:', 0.67

2021-06-24 reply 3

Little General author

? ?

2021-06-27 reply 4

Lao Wang

Is there a problem in deriving the general formula from layer l to layer L when discussing the superiority of the residual network training process?

07-01 reply 1

gifted luckyman

When do we need n_in *== n_out and stride = 1? It is always n_in != n_out.*

2019-03-01 reply 1

Click to view all comments ›

Speak rationally and interact in a friendly manner

The article is included in the following columns

Machine Learning Algorithm Engineer
WeChat public account, the content is more exciting!

Machine Learning Algorithm Engineer
Welcome to follow the WeChat public account of the same name

Recommended Reading