遥感图像分类

1.5 ResNet

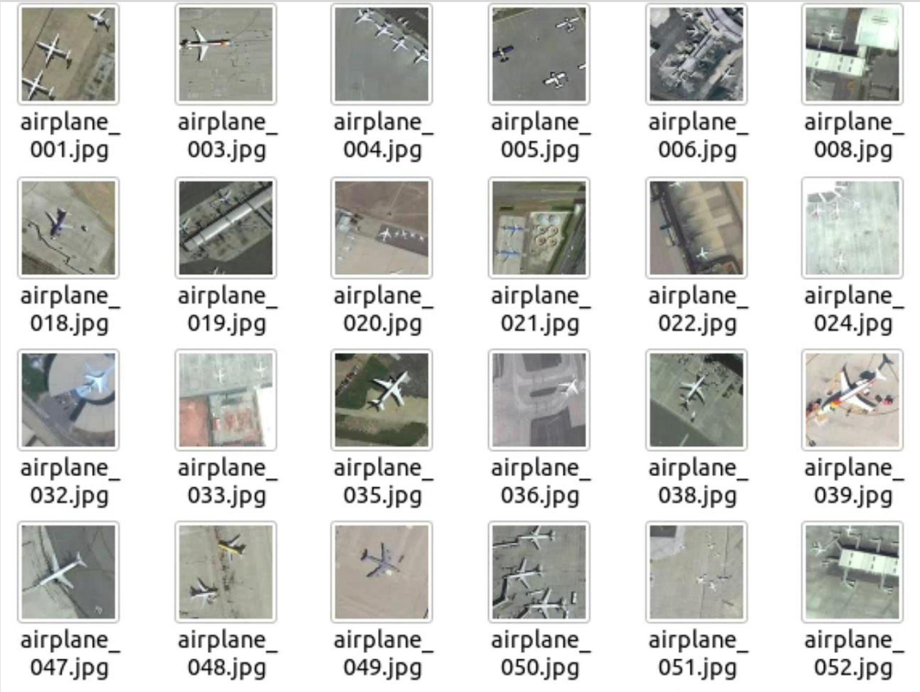
@tm9161

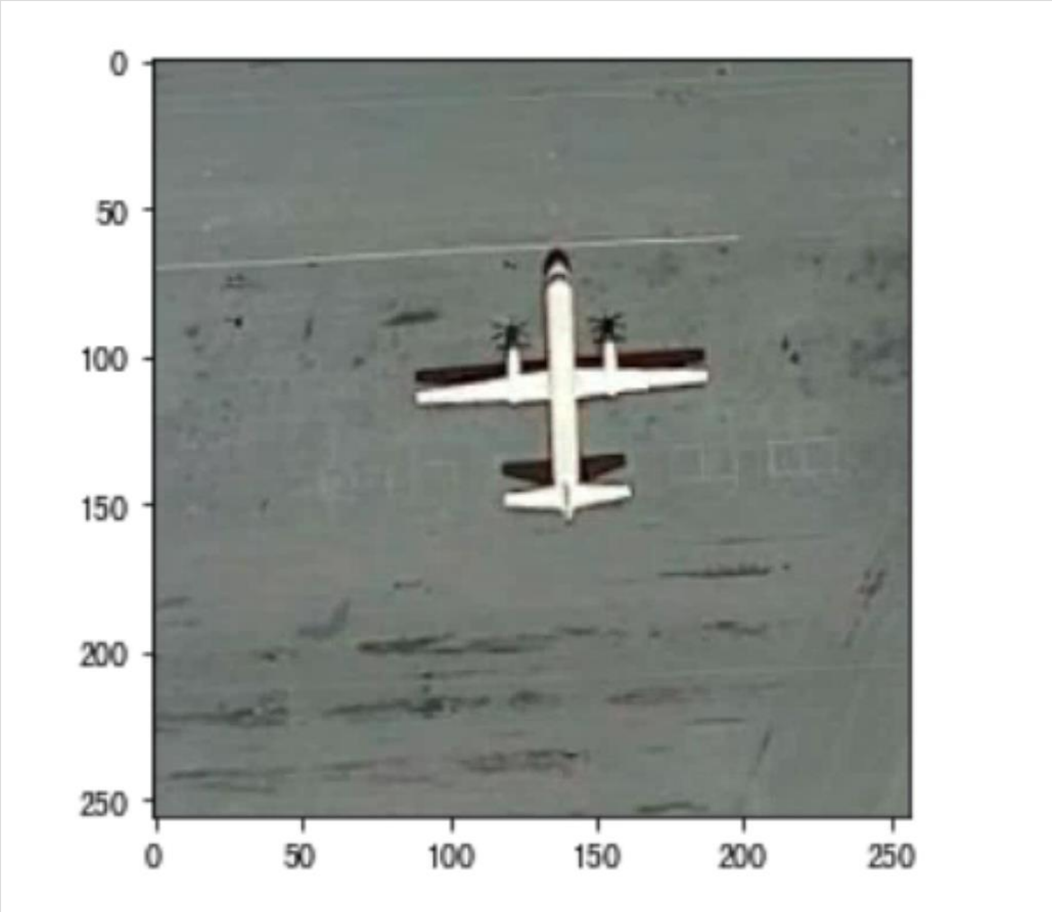
L5

ResNet

1. ResNet 结构
2. 批量归一化和残差结构
3. ResNet 代码实现

遥感图像数据集





包含31500张遥感图像（45类*700张），256x256像素的彩色图。

本次使用其中的5类，划分每类630张为训练集，70张为测试集。

载入数据

1.按路径读取

2.预处理
a.归一化
b.水平翻转
c.批大小
d.随机
e.尺寸
f.独热编码

```
train_dir = 'sat2/train'
test_dir = 'sat2/val'

im_size = 224
batch_size = 32
```

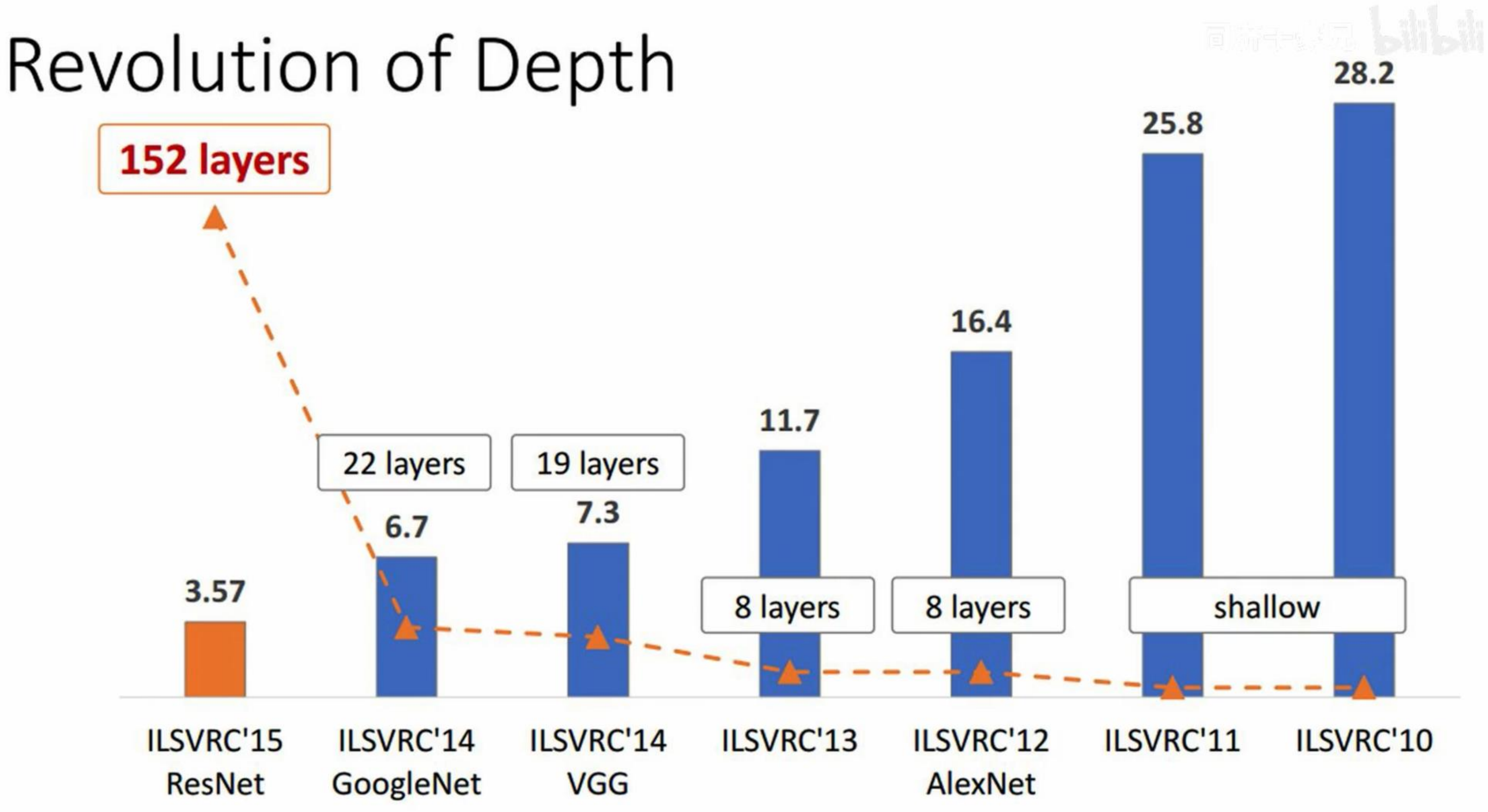
```
train_images = ImageDataGenerator(rescale = 1/255,horizontal_flip=True)
test_images = ImageDataGenerator(rescale = 1/255)
#归一化
```

```
train_gen = train_images.flow_from_directory(directory=train_dir,
                                             batch_size=batch_size,
                                             shuffle=True,
                                             target_size=(im_size, im_size),
                                             class_mode='categorical')
#按路径载入图片，批处理大小，随机，尺寸，读热编码
```

```
Found 3150 images belonging to 5 classes.
```

```
val_gen = test_images.flow_from_directory(directory=test_dir,
                                          batch_size=batch_size,
                                          shuffle=False,
                                          target_size=(im_size, im_size),
                                          class_mode='categorical')
#按路径载入图片，批处理大小，随机，尺寸，读热编码
```

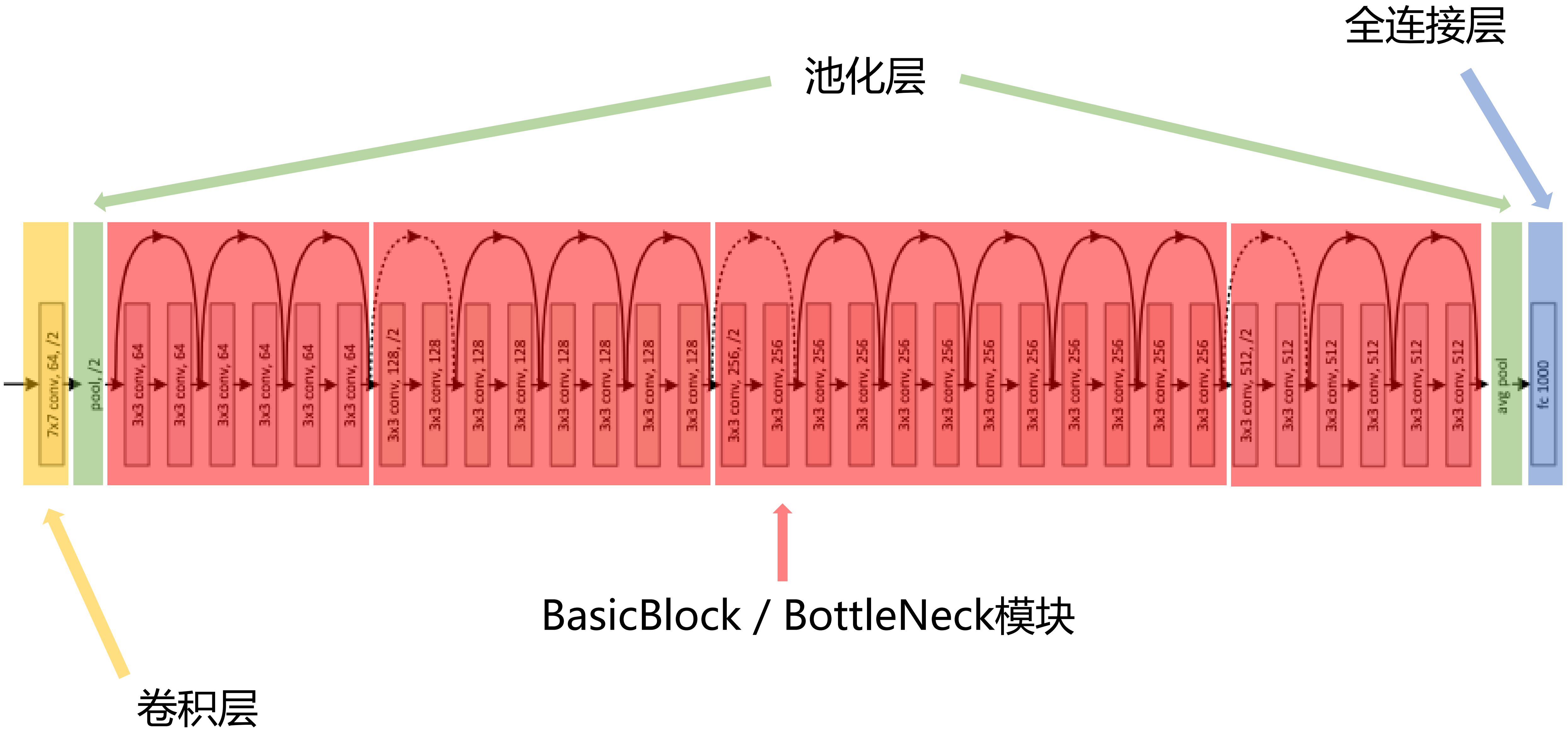
```
Found 350 images belonging to 5 classes.
```



ResNet由微软研究院的何恺明、张祥雨、任少卿、孙剑等提出的。

该网络发现了通过**残差结构**避免网络退化现象，神经网络的“深度”首次突破了100层。在2015 年的ILSVRC中取得了**冠军**。

ResNet 结构



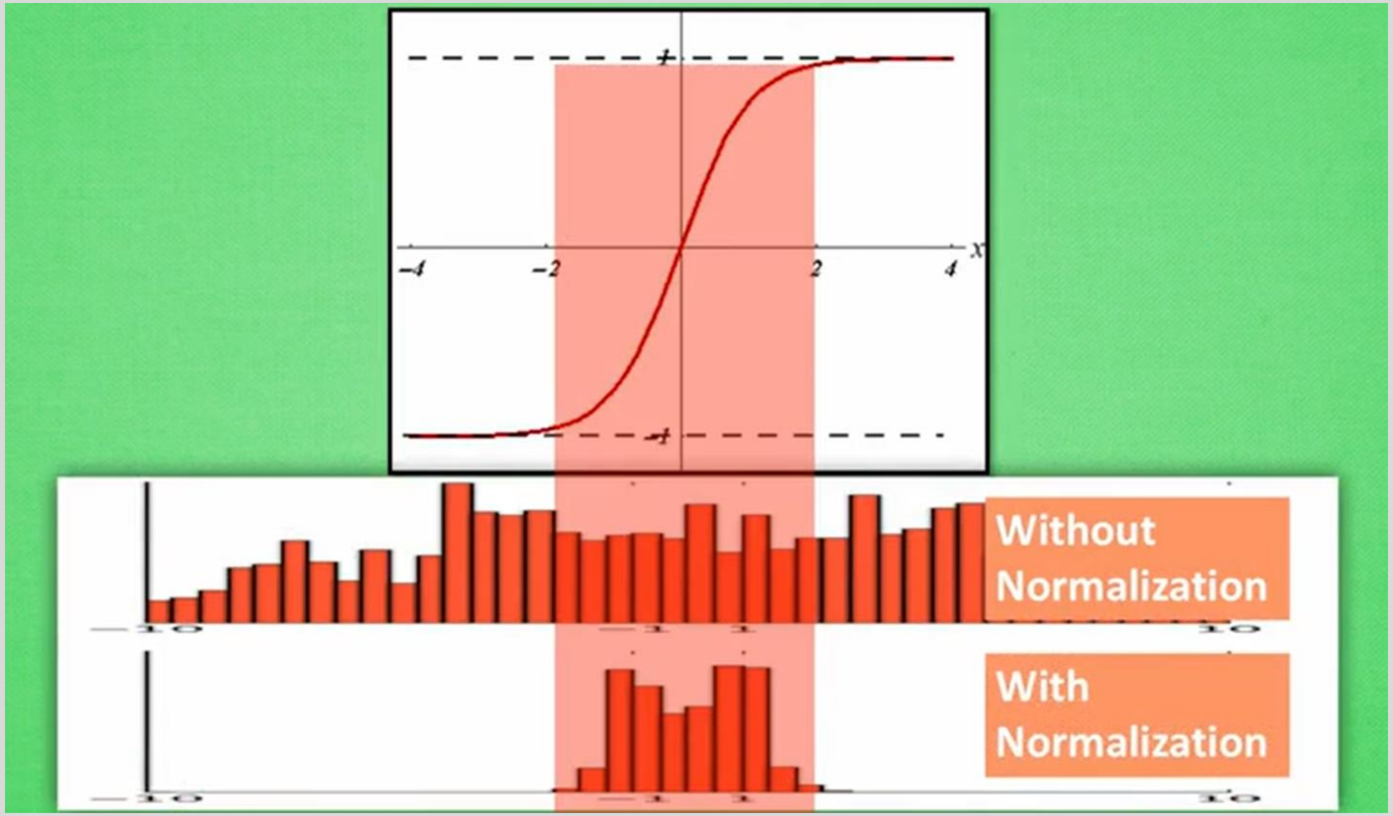
全连接层
池化层
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
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
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
BasicBlock / BottleNeck模块
卷积层

Batch Normalization 批量归一化

每一层输入的时候，先做一个归一化处理，然后再进入网络的下一层。

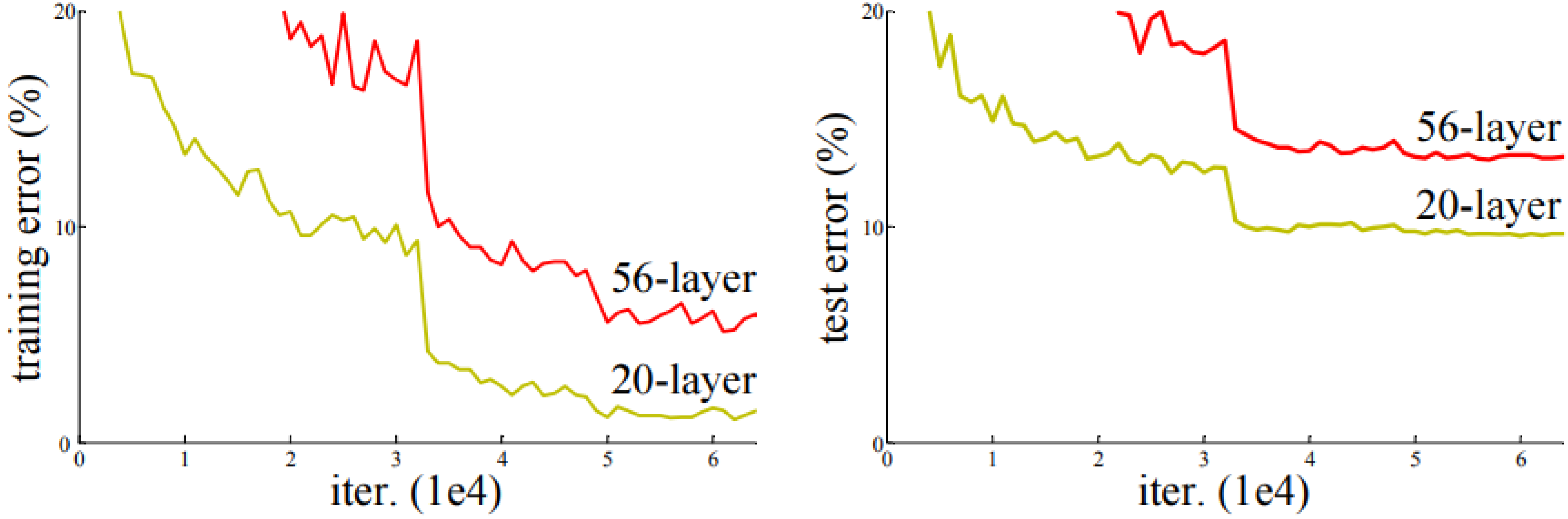
这个输入值的分布强行拉回到均值为0方差为1。

避免梯度消失和爆炸，训练更稳定。



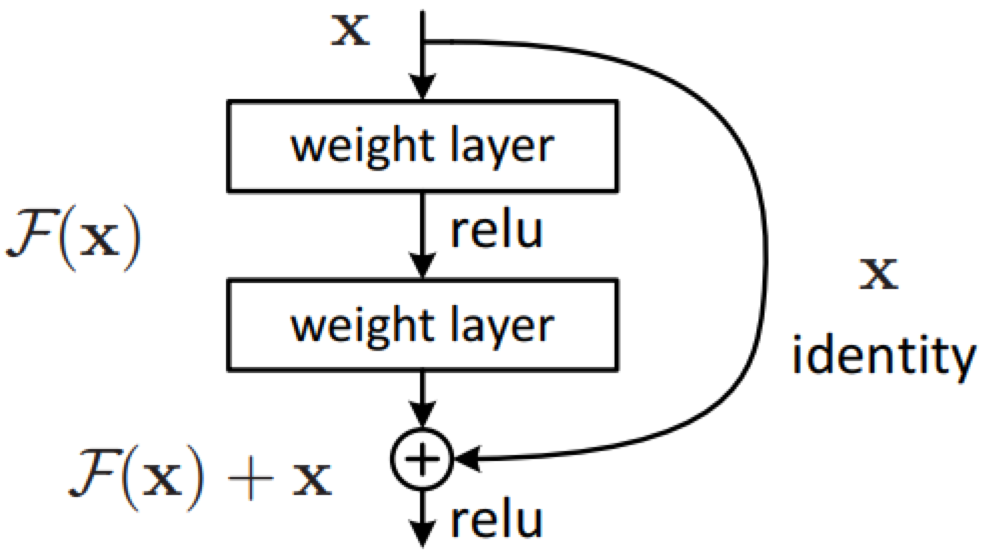

```
x = tf.keras.layers.BatchNormalization()(x)
```

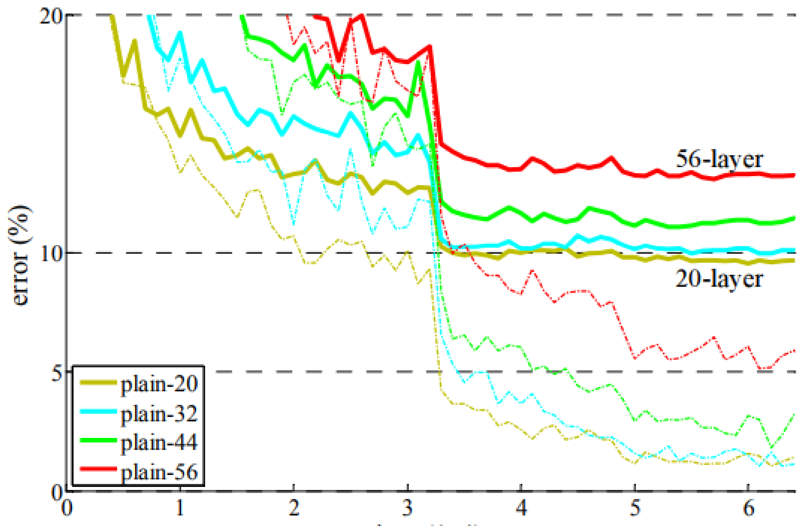
退化现象

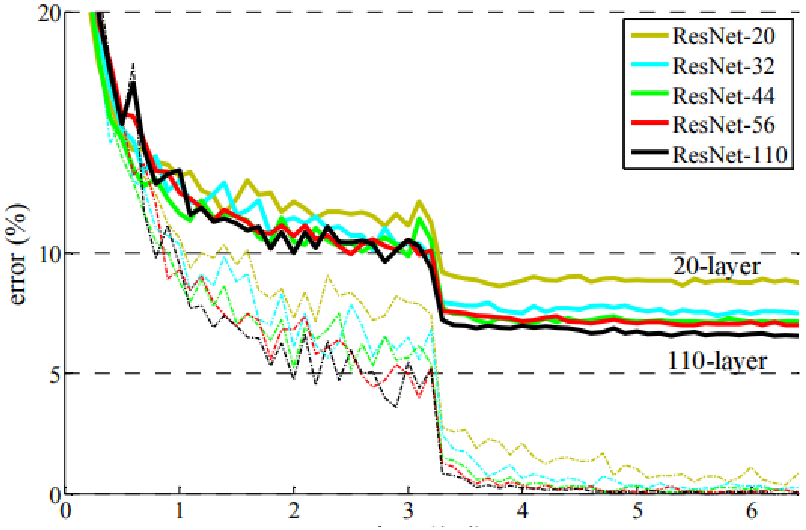


网络层数的增多，训练集loss逐渐下降，然后趋于饱和。
当你再增加网络深度的话，训练集loss反而增大。

捷径分支







输出为 H (x) = F (x) + x，
权重层实际上是学习一种残差映射：F (x) = H (x) - x

模型搭建

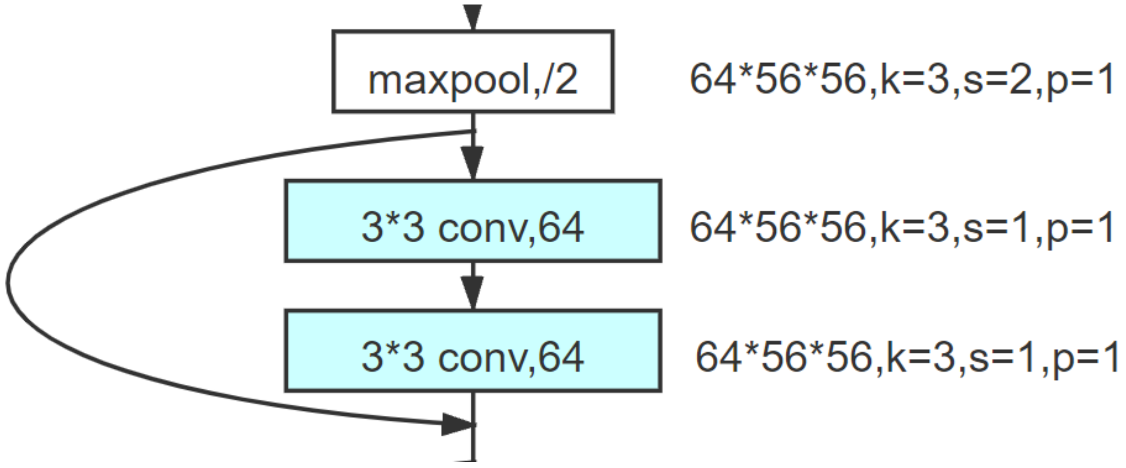
```python
model.add(tf.keras.layers.Conv2D(filters = 6,kernel_size = (5,5),input_shape=(28,28,1),padding = 'same',activation
model.add(tf.keras.layers.AveragePooling2D(pool_size = (2, 2)))
model.add(tf.keras.layers.Conv2D(filters = 16,kernel_size = (5,5),activation = "sigmoid"))
model.add(tf.keras.layers.AveragePooling2D(pool_size = (2, 2)))
model.add(tf.keras.layers.Conv2D(filters = 120,kernel_size = (5,5),activation = "sigmoid"))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(84, activation='sigmoid'))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

为满足多分支的模型，使用x = tf.keras.layers.XXX (   ) (X)搭建模型

```python
def LeNet():
    input_image = tf.keras.layers.Input(shape=(28, 28, 1))
    x = tf.keras.layers.Conv2D(6, kernel_size=5, padding="same", activation="sigmoid")(input_image)
    x = tf.keras.layers.AveragePooling2D(pool_size=2)(x)
    x = tf.keras.layers.Conv2D(16, kernel_size=5, activation="sigmoid")(x)
    x = tf.keras.layers.AveragePooling2D(pool_size=2)(x)
    x = tf.keras.layers.Conv2D(120, kernel_size=5, activation="sigmoid")(x)
    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dense(84, activation="sigmoid")(x)
    x = tf.keras.layers.Dense(10, activation="sigmoid")(x)

    model = tf.keras.models.Model(inputs=input_image, outputs=x)
    return model
```

残差模块

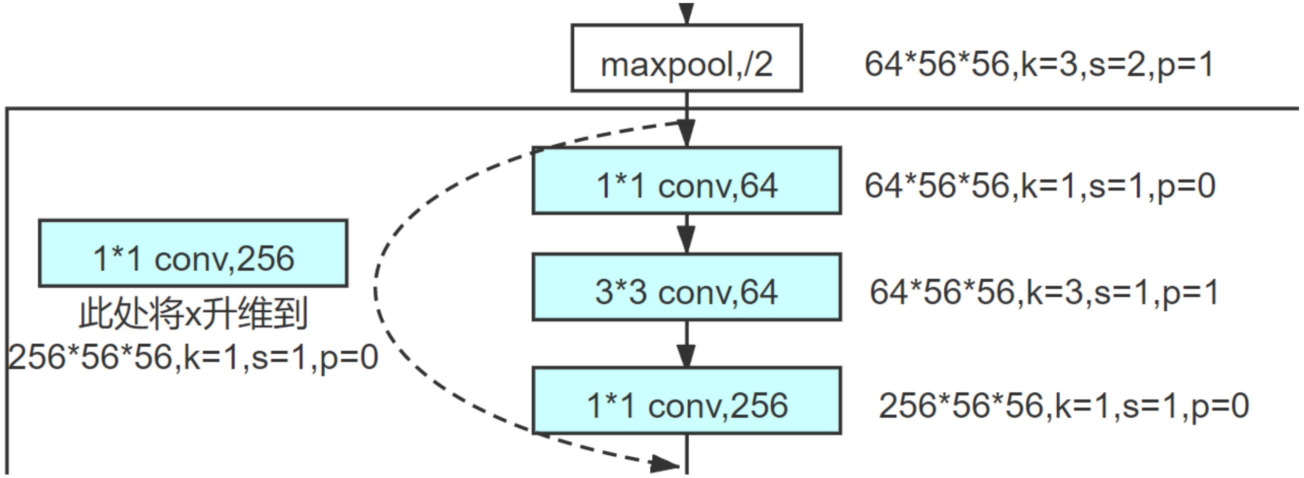


BasicBlock 模块

```
x = tf.keras.layers.Conv2D(filters=filter_num, kernel_size=3, strides=strides, padding='same')(_inputs)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Conv2D(filter_num, kernel_size=3, strides=1, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
```

```
y = tf.keras.layers.Conv2D(filters=filter_num,kernel_size=1,strides=strides)(_inputs)
y = tf.keras.layers.BatchNormalization()(y)
```

残差模块



BottleNeck 模块

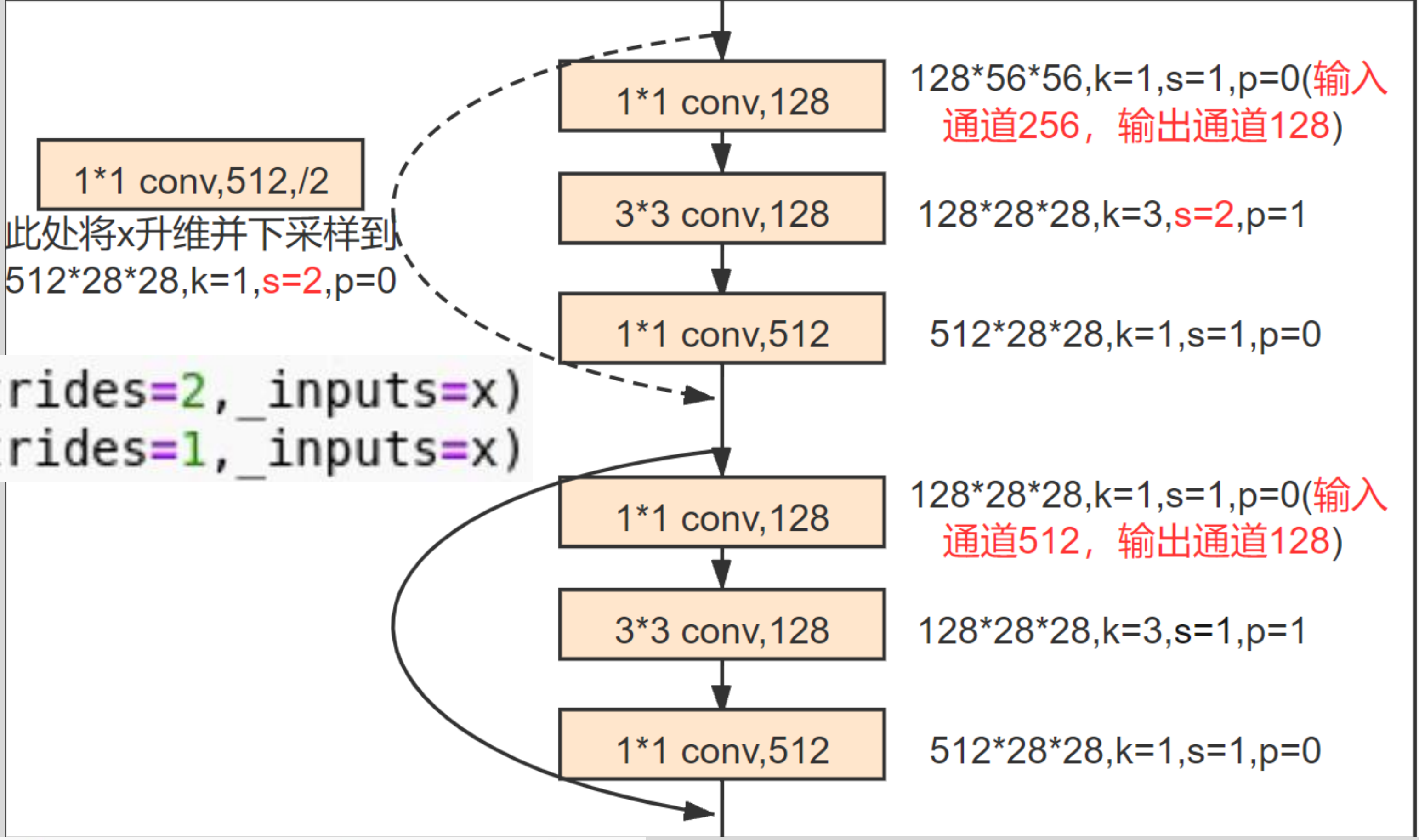
1×1的卷积将输入降到64维，然后通过1×1恢复。

减少参数量和计算量

```
x = tf.keras.layers.Conv2D(filters=filter_num,kernel_size=1,strides=1,padding='same')(_inputs)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Conv2D(filters=filter_num,kernel_size=3,strides=strides,padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Conv2D(filters=filter_num * 4,kernel_size=1,strides=1,padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
```

```
y = tf.keras.layers.Conv2D(filters=filter_num* 4,kernel_size=1,strides=strides)(_inputs)
y = tf.keras.layers.BatchNormalization()(y)
```

```
x = BottleNeck(filter_num=128,strides=2,_inputs=x)
x = BottleNeck(filter_num=128,strides=1,_inputs=x)
```




```
x = tf.keras.layers.Conv2D(filters=filter_num,kernel_size=1,strides=1,padding='same')(_inputs)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Conv2D(filters=filter_num,kernel_size=3,strides=strides,padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Conv2D(filters=filter_num * 4,kernel_size=1,strides=1,padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)

if strides != 1 or down==True:
    y = tf.keras.layers.Conv2D(filters=filter_num* 4,kernel_size=1,strides=strides)(_inputs)
    y = tf.keras.layers.BatchNormalization()(y)
else:
    y = _inputs
```

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

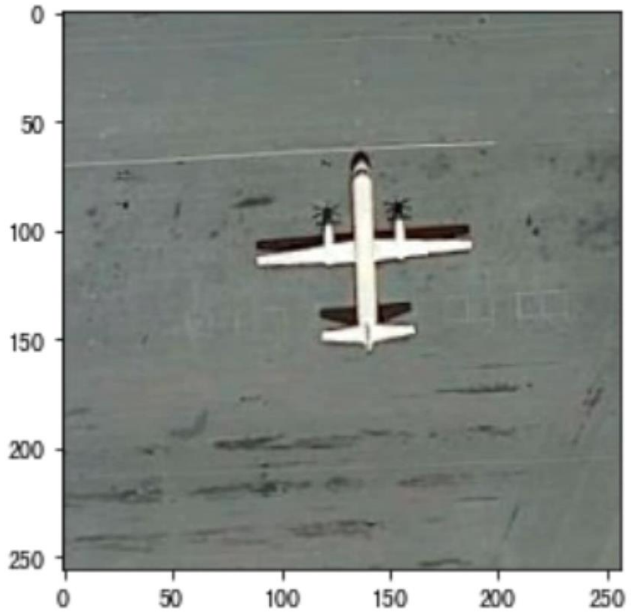
Downsampling is performed by conv3 1, conv4 1, and conv5 1 with a stride of 2

图片读取&预处理

```
In [3]: img = cv2.imread('1.jpg',1)
        #读取图片

In [4]: plt.imshow(img)

Out[4]: <matplotlib.image.AxesImage at 0x7fdb782b0ac0>
```



1.图片读取：cv2.imread

2.图片大小调整：cv2.resize

3.图片维度调整：reshape

4.归一化：/255

```
In [5]: img.shape

Out[5]: (256, 256, 3)

In [6]: img = cv2.resize(img,(224,224))
        img = img.reshape(1,224,224,3)
        img = img/255
        #图片预处理

In [7]: img.shape

Out[7]: (1, 224, 224, 3)
```

模型预测

In [7]:
```
img.shape
```
Out[7]:
```
(1, 224, 224, 3)
```

In [8]:
```
predict = model.predict(img)
```

In [9]:
```
predict
```
Out[9]:
```
array([[9.8531371e-01, 6.8616024e-03, 1.4397403e-06, 6.1821360e-03,
        1.6411012e-03]], dtype=float32)
```

In [10]:
```
label = ['airplane','bridge','palace','ship','stadium']
```

参考资料：

1.Deep Residual Learning for Image Recognition
https://arxiv.org/abs/1512.03385

2.残差网络 ResNet【动手学深度学习v2】
https://www.bilibili.com/video/BV1bV41177ap

3. ResNet网络结构，BN以及迁移学习详解
https://www.bilibili.com/video/BV1T7411T7wa

4. resnet18 50网络结构（SVG图）
https://www.jianshu.com/p/085f4c8256f1

5.什么是 Batch Normalization 批标准化
https://www.bilibili.com/video/av16000304?zw

6. ResNet深度残差网络
https://www.bilibili.com/video/BV1vb4y1k7BV