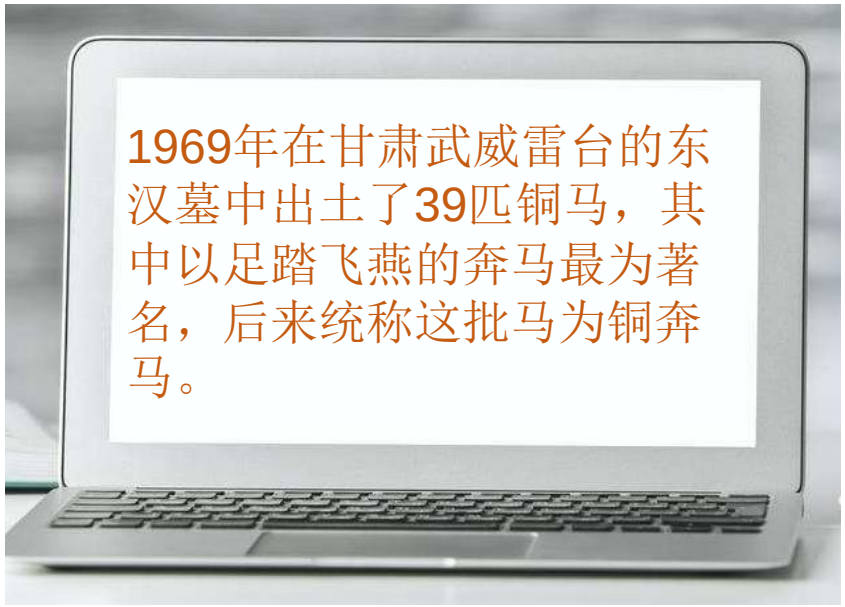
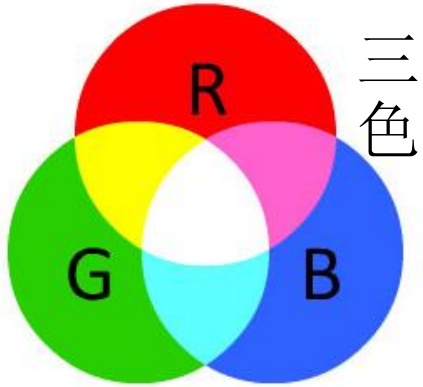
深度卷积神经网络

雍宾宾

yongbb@lzu.edu.cn


1969年在甘肃武威雷台的东汉墓中出土了39匹铜马，其中以足踏飞燕的奔马最为著名，后来统称这批马为铜奔马。

计算机图像基础



三原色

=

红色通道

153, 151, 151, 152, 153, 152, 153, 153, 153, 153, 153, 153, 149,
150, 151, 162, 171, 157, 136, 150, 145, 114, 104, 101, 98, 104,
111, 121, 123, 126, 125, 123, 111, 114, 115, 105, 106, 102, 96,
102, 98, 100, 106, 102, 94, 91, 91, 84, 77, 80, 91, 98,
100, 95, 95, 100, 108, 110, 106, 107, 90, 80, 74, 71, 74,
63, 70, 92, 94, 90, 101, 115, 109, 79, 67, 83, 91, 74,
60, 69, 80, 93, 88, 70, 69, 75, 84, 81, 77, 72, 61,
62, 74, 68, 63, 81, 77, 77, 120, 152, 150, 144, 143, 145,
145, 146, 148, 148, 147, 146, 145, 144, 144, 145, 146, 144, 144,
146, 146, 144, 144, 144, 144, 145, 147, 146, 143, 144, 144, 144,
143, 144, 144, 143, 142, 141, 142, 143, 144, 143, 142, 143, 144,
145, 145, 144, 142, 142, 143, 142, 143, 142, 140, 141, 144, 143,
142, 141, 142, 140, 140, 142, 141, 139, 139, 140, 139, 139, 141,
141, 139, 137, 138, 139, 140, 140, 140, 140, 139, 139, 140, 139,
139, 139, 140, 139, 140, 140, 139, 138, 138, 138, 139, 138, 137,
138, 137, 137, 138, 137, 138, 138, 137, 136, 136, 135, 134, 136,
137, 138, 139, 138, 138, 136, 137, 137, 137, 137, 136, 136, 136,
136, 136, 138, 137, 135, 135, 136, 136, 135, 137, 135, 135, 134,

+

绿色通道

156, 154, 152, 151, 152, 153, 153, 153, 152, 153, 152, 151, 152,
152, 150, 152, 153, 152, 151, 152, 153, 152, 151, 152, 152, 154,
154, 152, 150, 152, 154, 155, 156, 156, 153, 152, 152, 151, 153,
152, 150, 150, 151, 153, 153, 152, 153, 154, 154, 153, 152, 151,
151, 152, 151, 151, 152, 152, 153, 152, 150, 151, 152, 150, 148,
151, 159, 172, 172, 160, 136, 140, 152, 130, 114, 103, 91, 106,
120, 126, 125, 103, 126, 118, 95, 113, 113, 92, 89, 94, 91,
92, 85, 84, 92, 98, 104, 98, 73, 63, 75, 71, 78, 85,
82, 88, 86, 77, 79, 93, 96, 94, 84, 81, 85, 86, 79,
70, 82, 86, 78, 90, 106, 103, 90, 86, 86, 76, 75, 84,
86, 83, 75, 81, 92, 85, 66, 70, 82, 68, 69, 69, 65,
71, 74, 82, 85, 75, 67, 100, 148, 154, 146, 145, 144, 144,
144, 144, 144, 144, 144, 144, 143, 143, 144, 145, 145, 144, 143,
144, 144, 145, 145, 144, 143, 144, 145, 145, 143, 142, 144, 143,
143, 146, 143, 143, 143, 142, 143, 144, 143, 142, 141, 140, 141,
142, 144, 143, 140, 140, 141, 141, 141, 142, 141, 141, 143, 142,
141, 140, 140, 141, 141, 141, 139, 139, 140, 141, 140, 141, 140,
140, 138, 137, 138, 139, 140, 139, 138, 140, 140, 139, 139, 140,
139, 139, 139, 138, 139, 138, 136, 138, 139, 138, 138, 138, 136,
135, 135, 136, 137, 136, 136, 136, 136, 135, 135, 134, 136, 137,

+

蓝色通道

155, 156, 156, 155, 155, 154, 154, 155, 155, 156, 154, 153, 150,
155, 170, 177, 151, 134, 154, 164, 151, 119, 118, 143, 141, 123,
131, 132, 131, 137, 112, 110, 123, 85, 90, 104, 95, 98, 98,
91, 89, 91, 99, 100, 83, 83, 98, 96, 98, 111, 112, 104,
101, 86, 75, 80, 85, 92, 92, 91, 92, 83, 94, 117, 102,
61, 70, 82, 63, 88, 94, 75, 99, 116, 106, 89, 80, 82,
84, 80, 93, 97, 74, 75, 80, 84, 103, 86, 70, 66, 59,
57, 73, 88, 68, 60, 70, 76, 68, 93, 152, 164, 148, 146,
149, 150, 149, 149, 148, 147, 146, 146, 149, 149, 147, 146, 146,
148, 149, 148, 149, 148, 146, 147, 146, 145, 146, 148, 147, 147,
147, 147, 147, 146, 146, 146, 145, 144, 145, 144, 144, 144, 146,
145, 144, 144, 144, 143, 144, 145, 145, 144, 144, 145, 147, 148,
147, 146, 144, 143, 142, 141, 142, 144, 145, 143, 142, 141, 142,
143, 142, 143, 144, 144, 145, 144, 143, 142, 141, 142, 142, 141,
140, 140, 142, 141, 140, 141, 141, 140, 141, 141, 140, 139, 140,
141, 140, 141, 141, 142, 141, 139, 140, 139, 138, 139, 139, 140,
142, 141, 140, 138, 137, 138, 140, 141, 140, 139, 138, 137, 139,

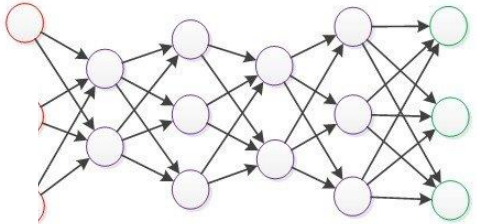
语义鸿沟

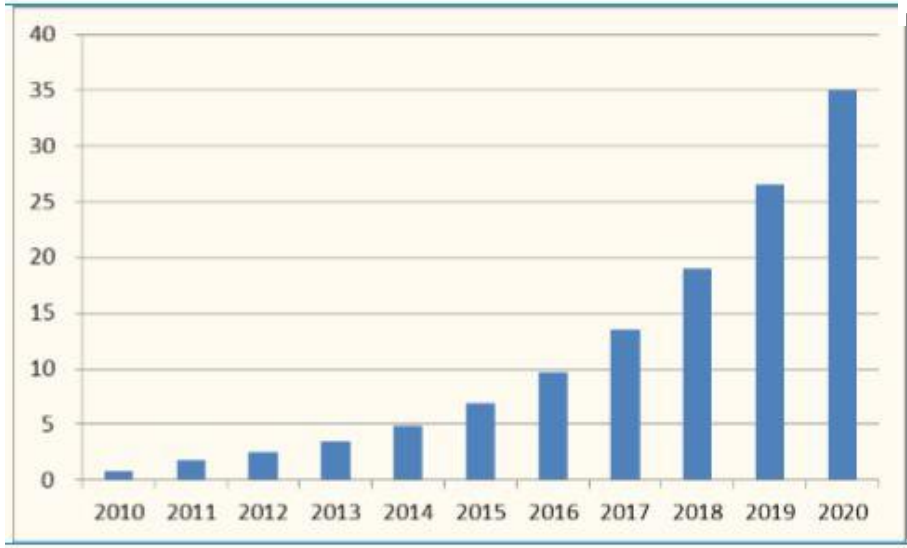
图像识别的困难

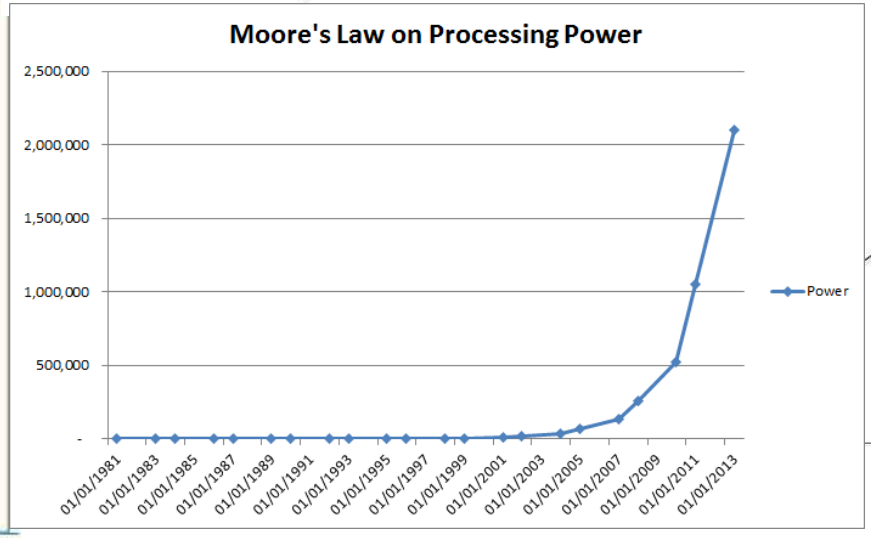
- 颜色不同
- 角度不同
- 大小不同
- 背景不同
- 光线不同
- 遮挡不同

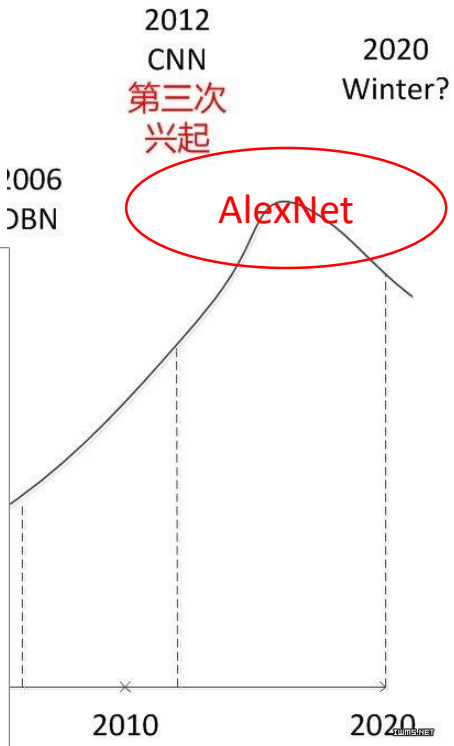
卷积神经网络(CNN)基石

多层神经网络



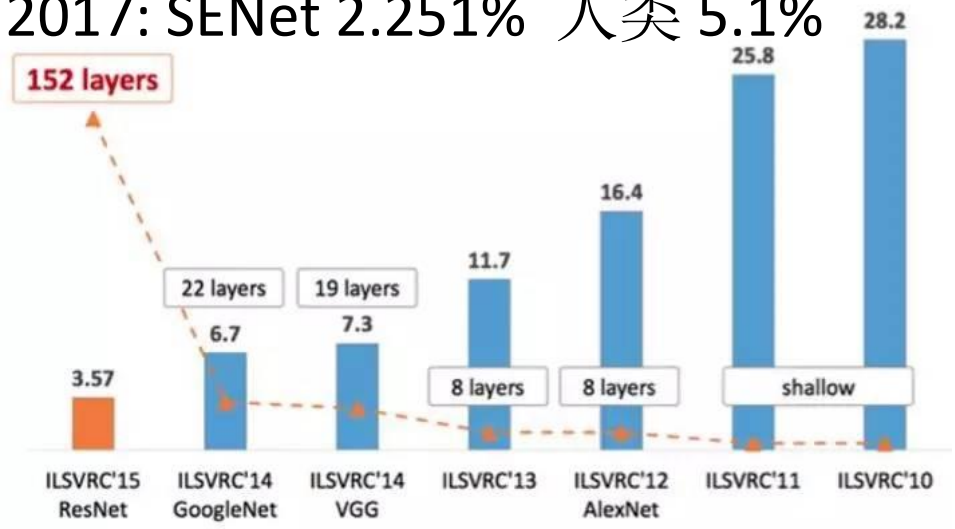




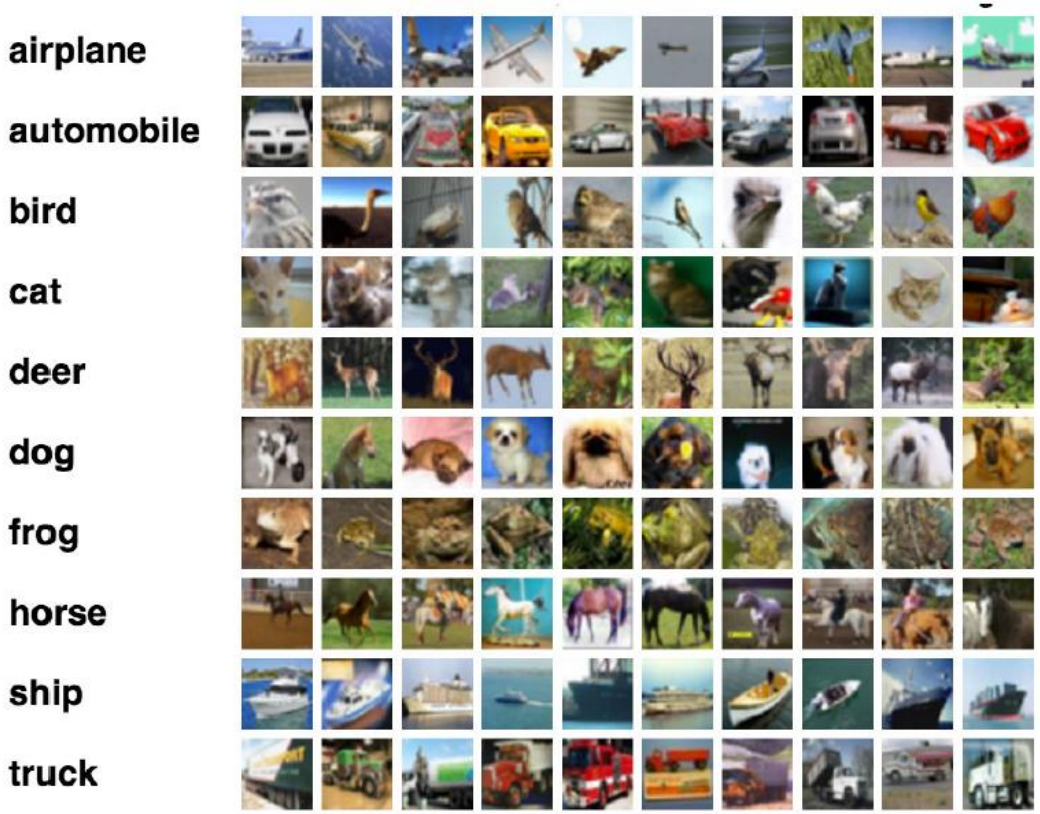


图像数据集

2017: SENet 2.251% 人类 5.1%



MNIST: training 60000， test 10000， best accuaray 99.77%

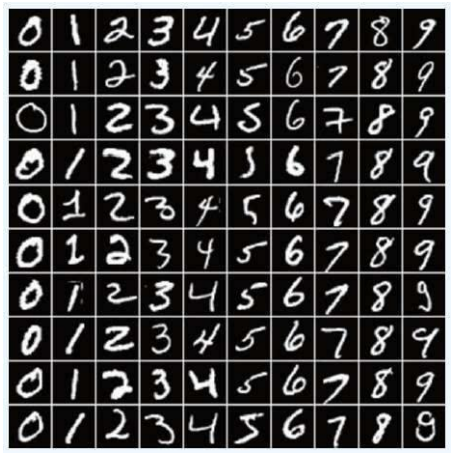


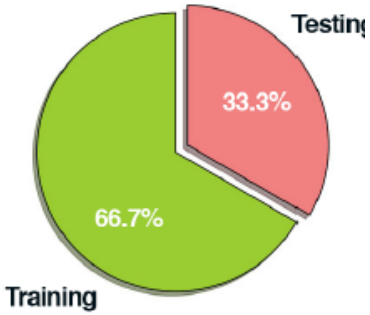
CIFAR-10: training 50000， test 10000， best accuaray 96.53%

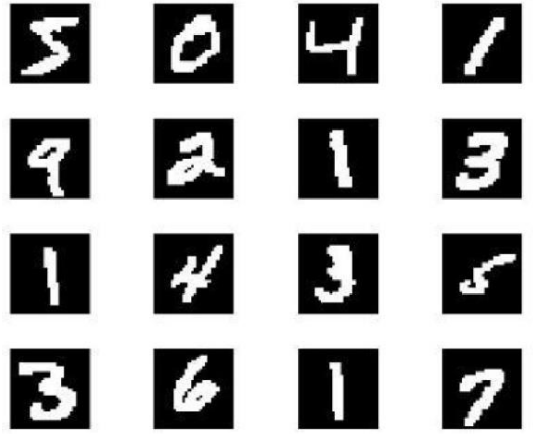
CIFAR-100: training 50000， test 10000， best accuaray 75.72%

开发深度(CNN)学习模型

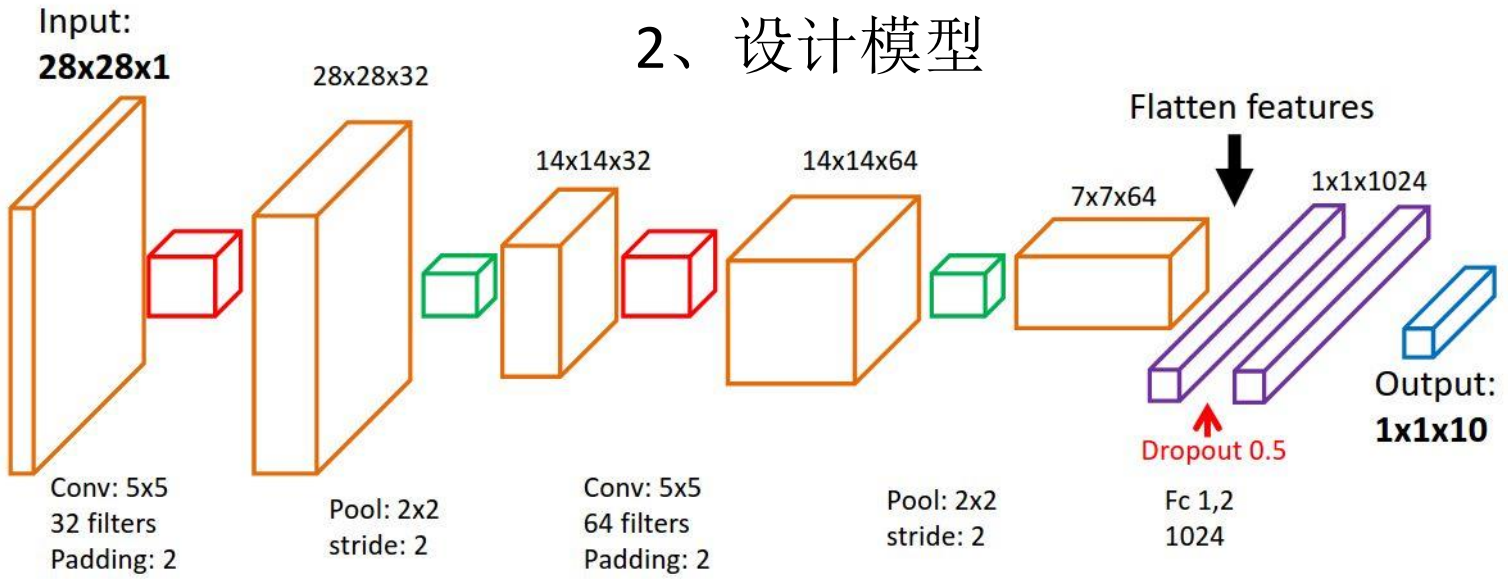
1、数据收集和处理







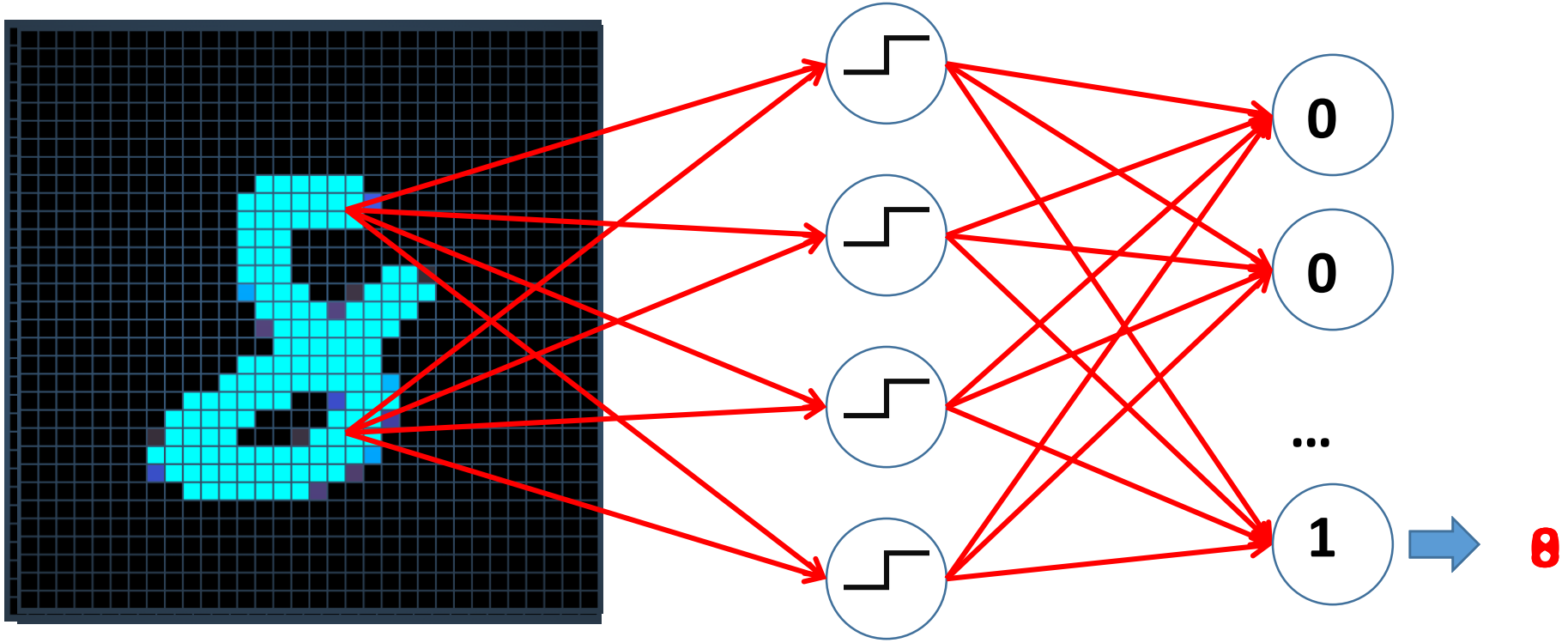
2、设计模型



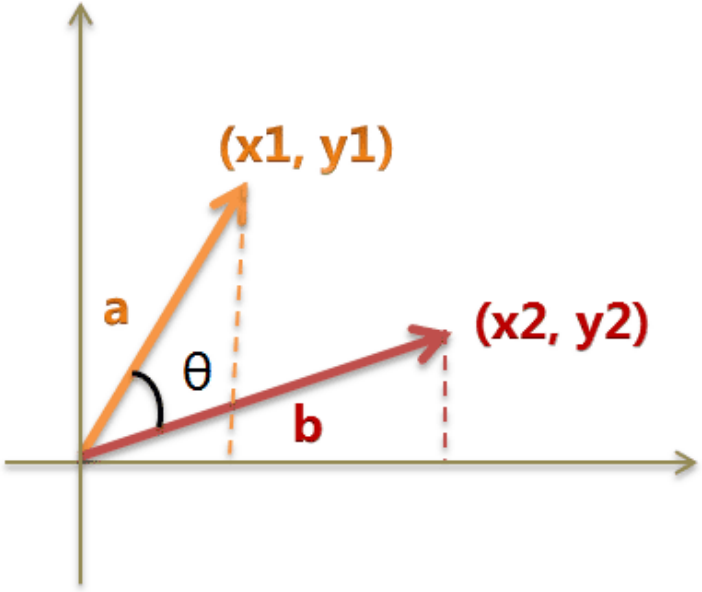
3、训练模型

4、验证模型

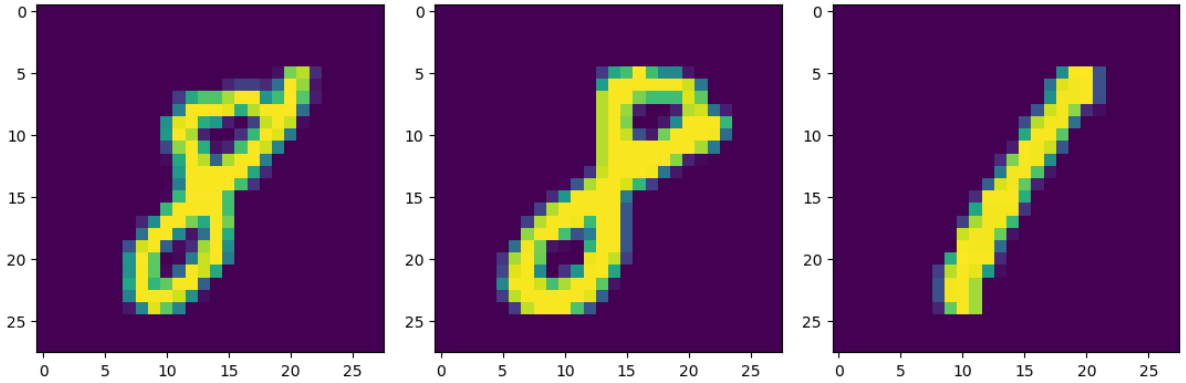
神经网络图像识别



神经网络图像识别-点积dot



$$\cos\theta = \frac{x_1x_2 + y_1y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$



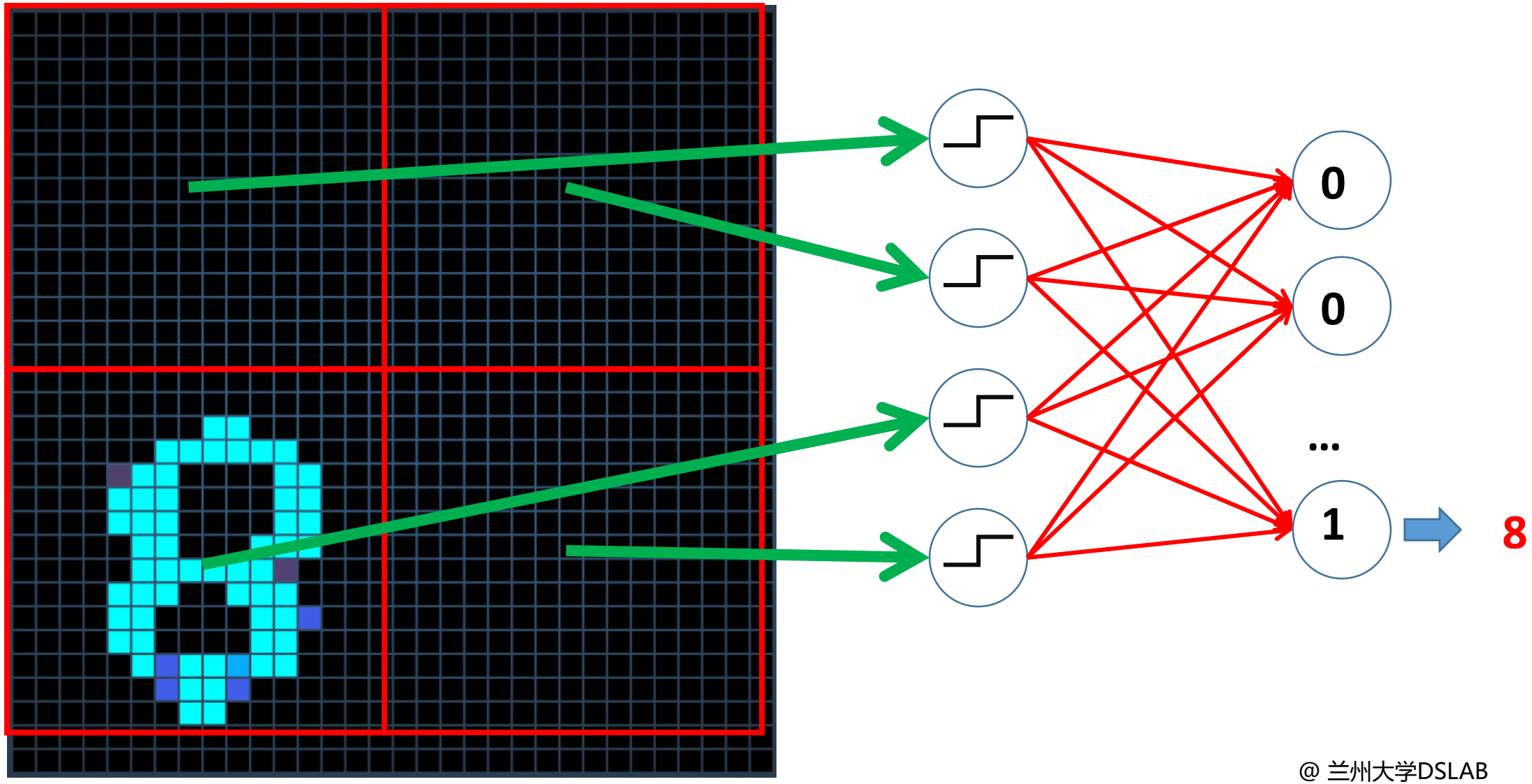
```
import keras
(x_train,y_train),(x_test,y_test)=keras.datasets.mnist.load_data()
x_train = x_train.reshape((-1,784))/255.0
digs = []
for i in range(10):
    digi = x_train[y_train==i][:1000]
    digs.append(digi)

for i in range(10):
    dotvalue = sum(sum(digs[8][0]*(digs[i])))/1000
    print(dotvalue)
```

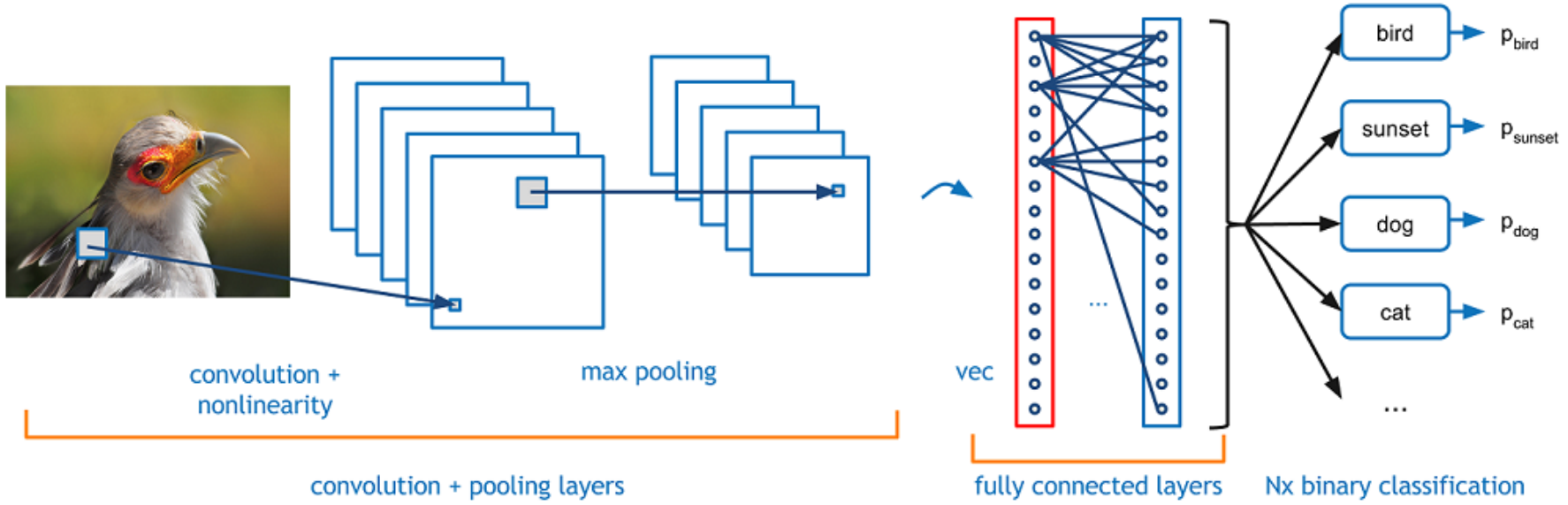
点积可以表征相似度

神经网络图像识别

- 增加样本
- 滑动窗口
- 卷积网络

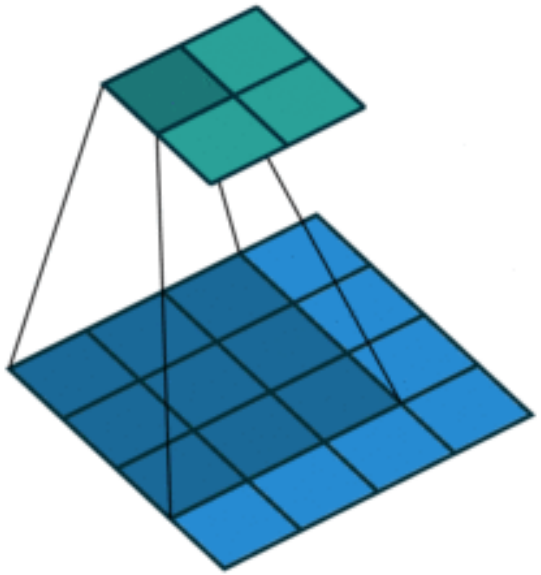


卷积神经四个组件

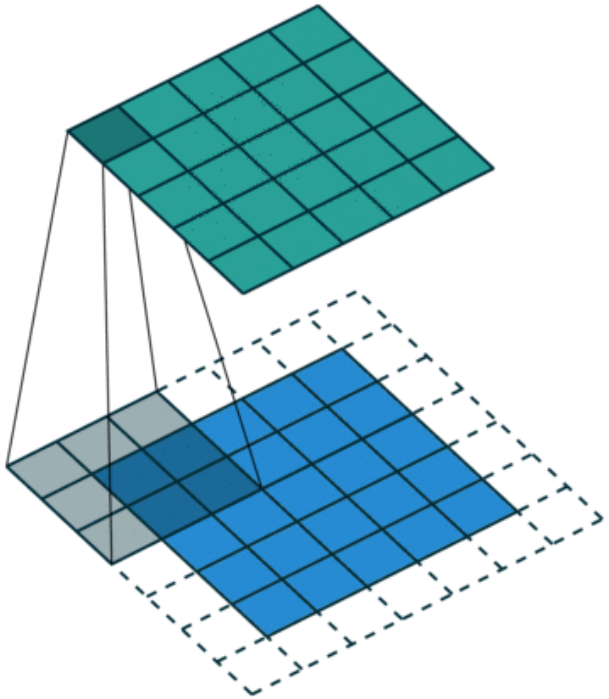


- 卷积层（Convolution）
- 非线性映射（ReLu）
- 池化层（Pooling）
- 分类层（FC）

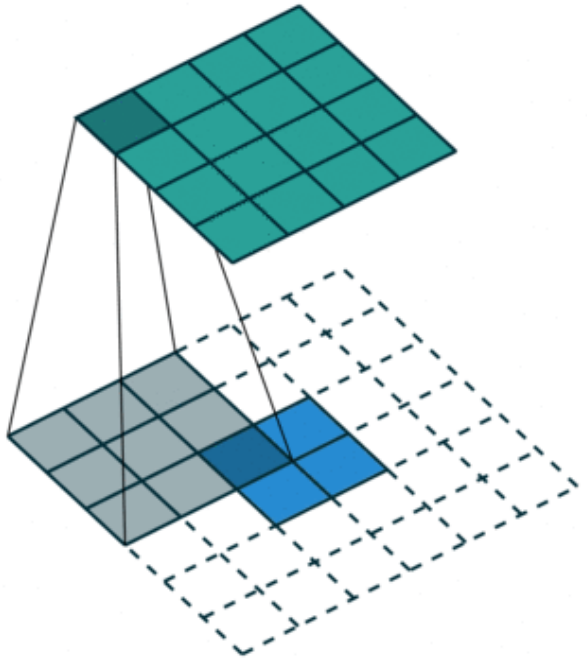
卷积层



padding=valid



padding=same



- input size i
- kernel size k
- stride s
- padding size p

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1.$$

- 参数共享
- 自动提取特征
- 平移不变性

卷积操作

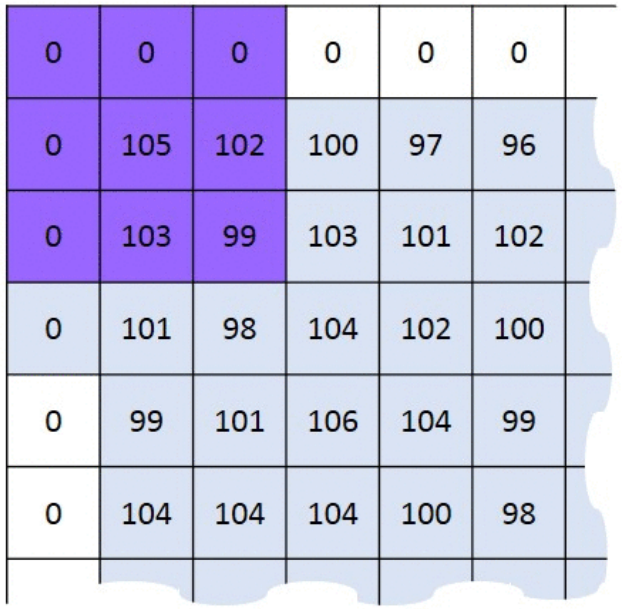
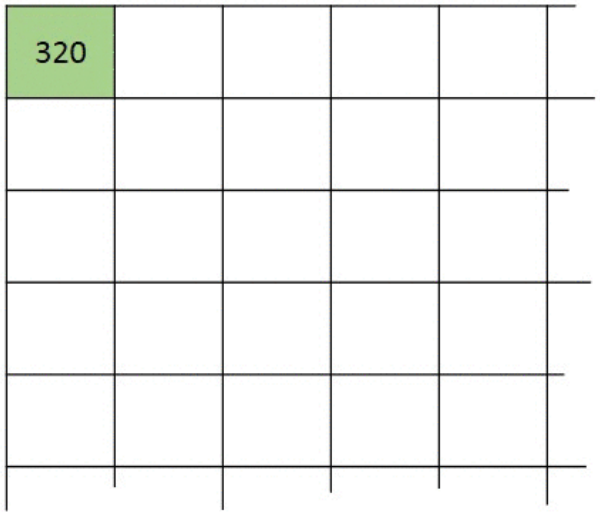


Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0



Output Matrix

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$

Convolution with horizontal and vertical strides = 1

多通道卷积操作

Input Image

红色通道

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

绿色通道

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

蓝色通道

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

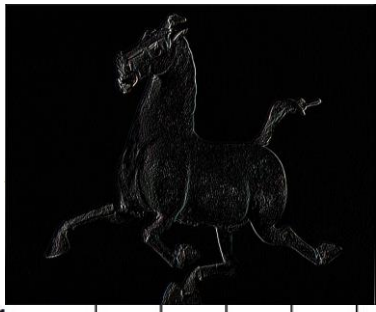
0	1	1
0	1	0
1	-1	1

Kernel Channel #3

卷积核有3个通道
权值个数3x3x3+1

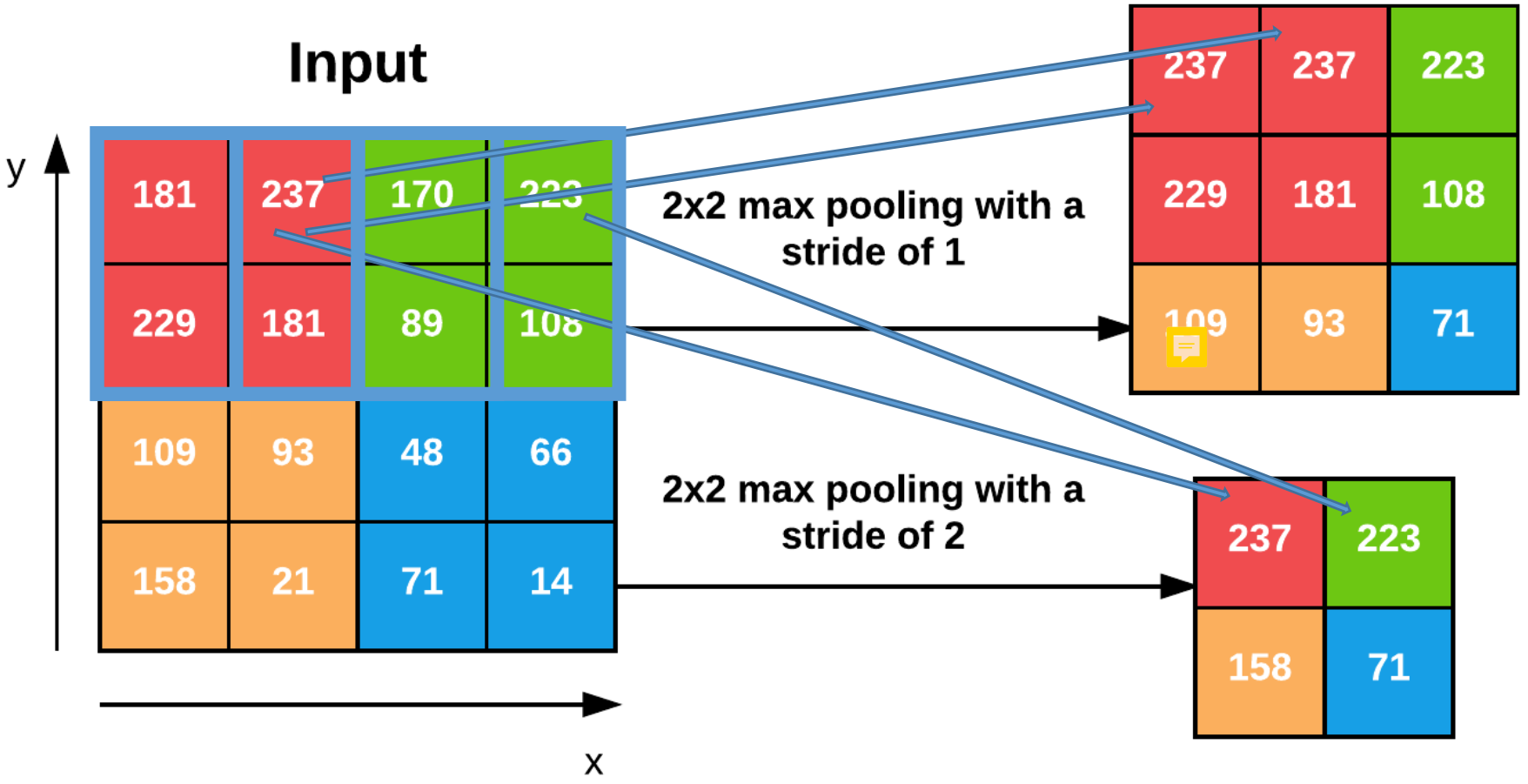
$$308 + (-498) + 164 + 1 =$$

Bias = 1



池化层

1. 一般池化
2. 重叠池化

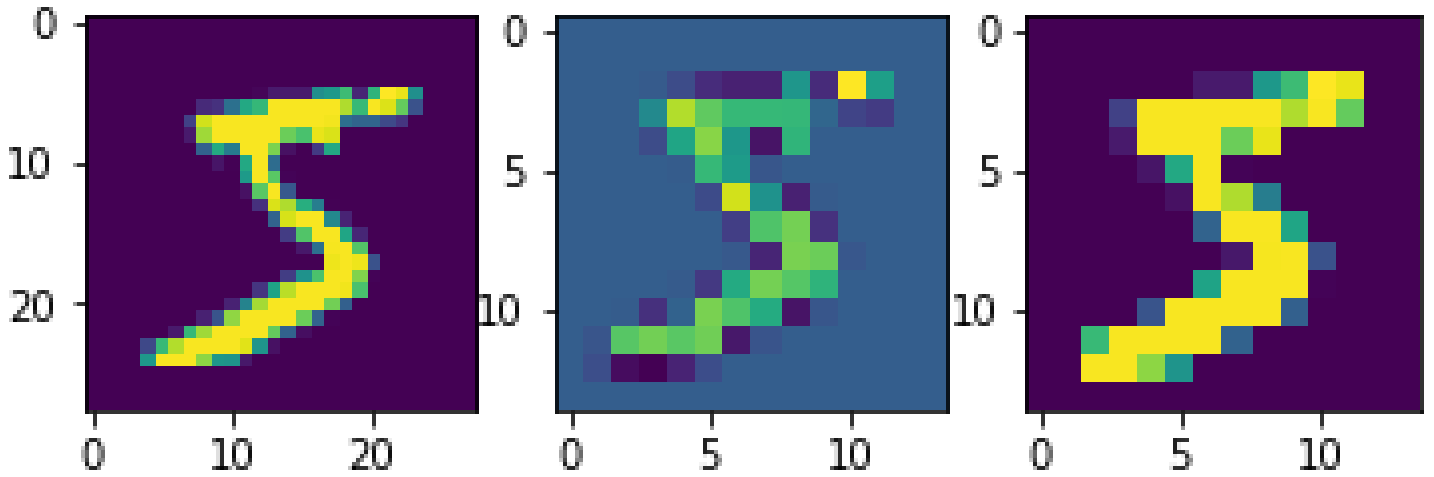


1. 池化无参数
2. 具有降维作用
3. 缩放不变

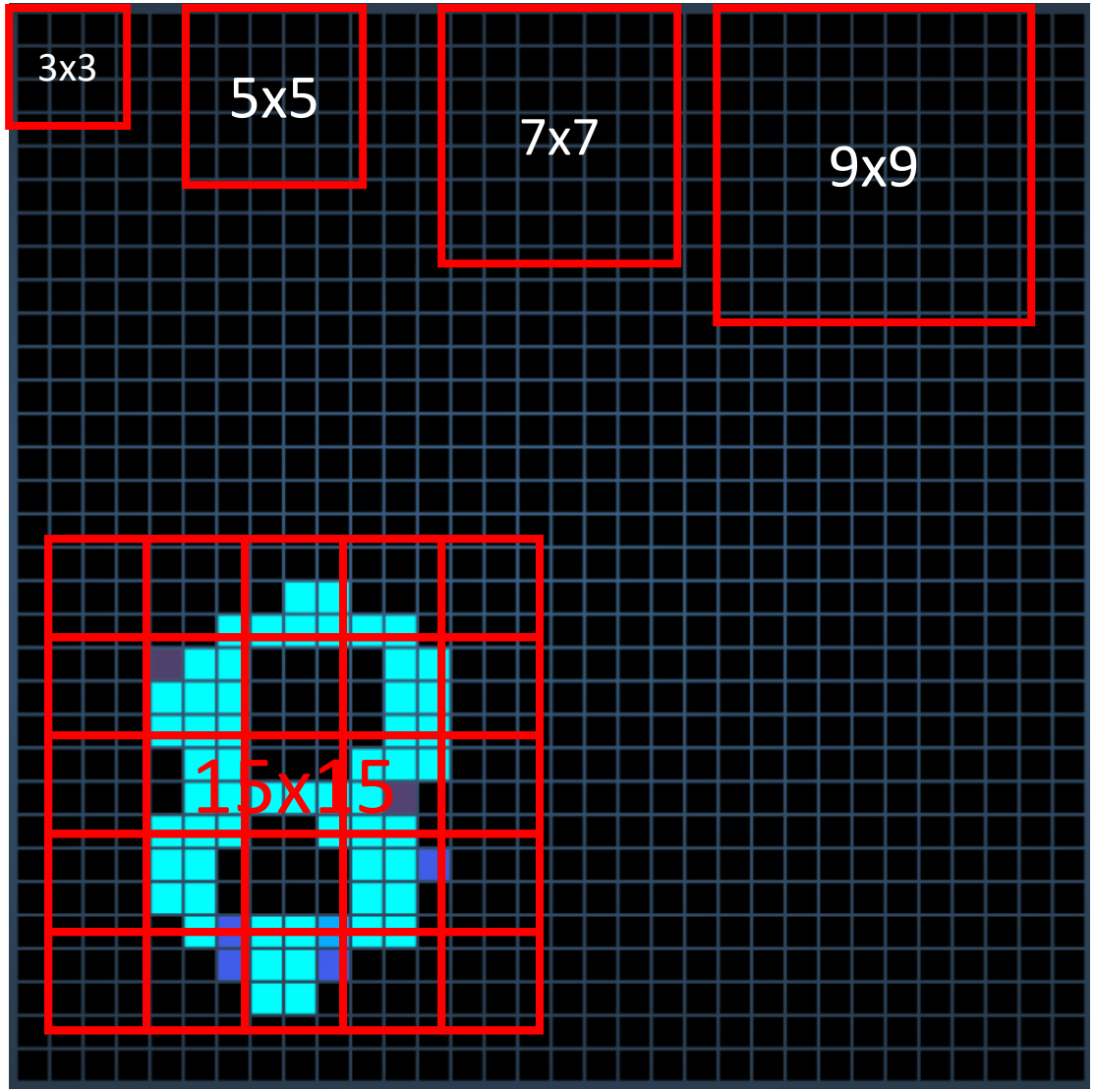
TensorFlow卷积池化

```
import tensorflow as tf
import numpy as np
(x_train,y_train),(x_test,y_test) = tf.keras.datasets.mnist.load_data()
x_train = x_train.reshape((-1,28,28,1))/255.0
x_test = x_test.reshape((-1,28,28,1))/255.0

filter = tf.constant([[-1,-1,-1],[-1,9,1],[-1,-1,-1]],dtype='float32')
filter = tf.reshape(filter,(3,3,1,1))
img = tf.constant(x_train[0:1]/255.0,dtype='float32')
conv_img = tf.nn.conv2d(img,filter,strides=[1,2,2,1],padding='SAME')
pool_img = tf.nn.max_pool(img,[1,2,2,1],[1,2,2,1],padding="VALID")
# window_size,stride
print(conv_img.shape)
plt.subplot(131)
plt.imshow(tf.squeeze(img))
plt.subplot(132)
plt.imshow(tf.squeeze(conv_img))
plt.subplot(133)
plt.imshow(tf.squeeze(pool_img))
plt.show()
```



卷积核大小

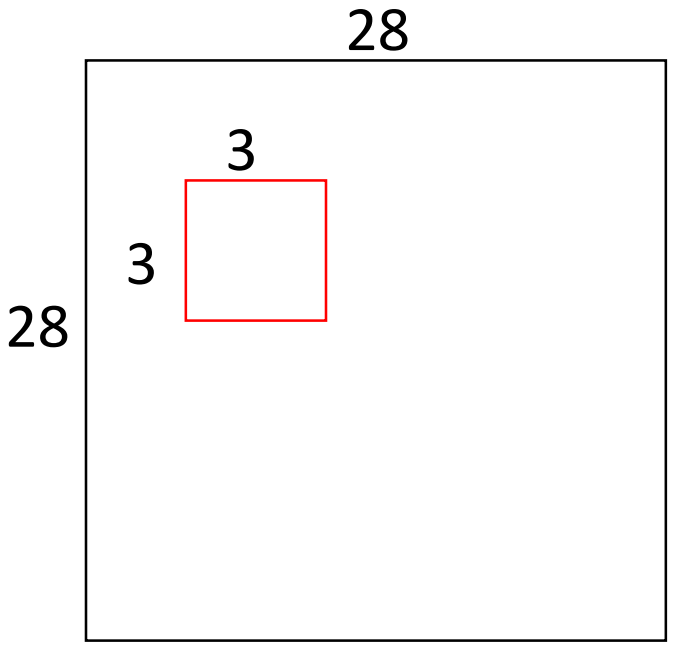


- 卷积核通常使用奇数 3x3, 5x5, 7x7,9x9
- 卷积核通常小而深

参数为 3 x 3 x C1xC2

参数为 5 x 5 x C1xC2

卷积输出计算实例



input depth = 3
output depth = 8

padding	stride	width	height	depth
same	1	28	28	8
valid	1	26	26	8
valid	2	13	13	8

权值计算

width height channel

32 x 32 x 3

20 kernels

conv size= 8x8

padding = 1

stride = 2

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1.$$

h=w=(32+2-8) / 2+1 = 14

width height channel

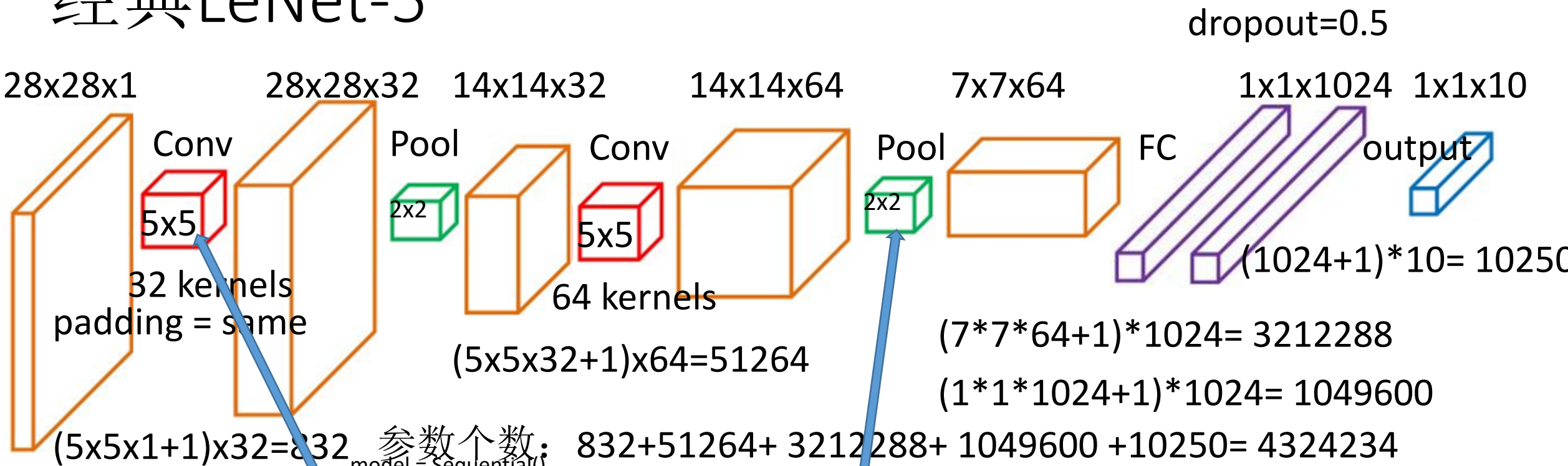
14 x 14 x 20

不使用参数共享，卷积层有多少参数？采用参数共享，卷积层有多少参数？

不共享参数：(8 * 8 * 3 + 1) * (14 * 14 * 20) = 756560

共享参数：(8 * 8 * 3 + 1) * 20 = 3860

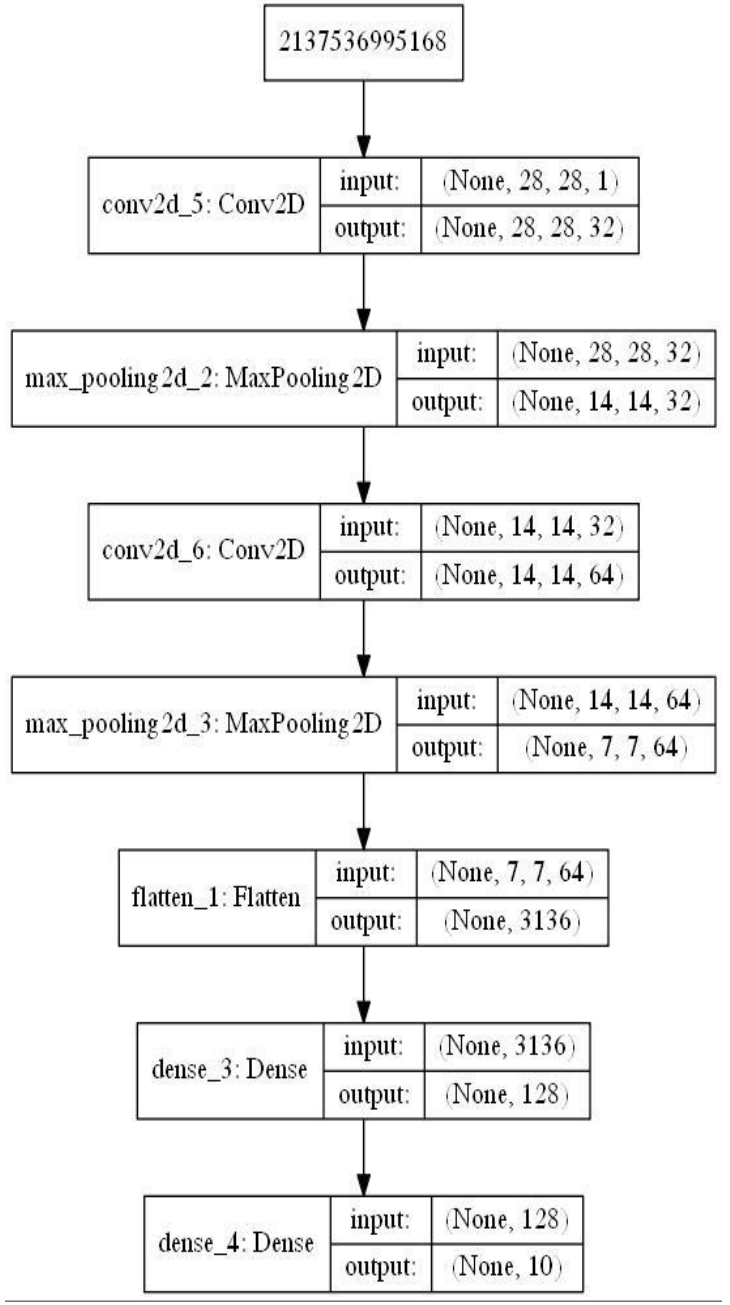
经典LeNet-5




```
model = Sequential()
model.add(Conv2D(32, (5,5), padding='same', activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5,5), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1024, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

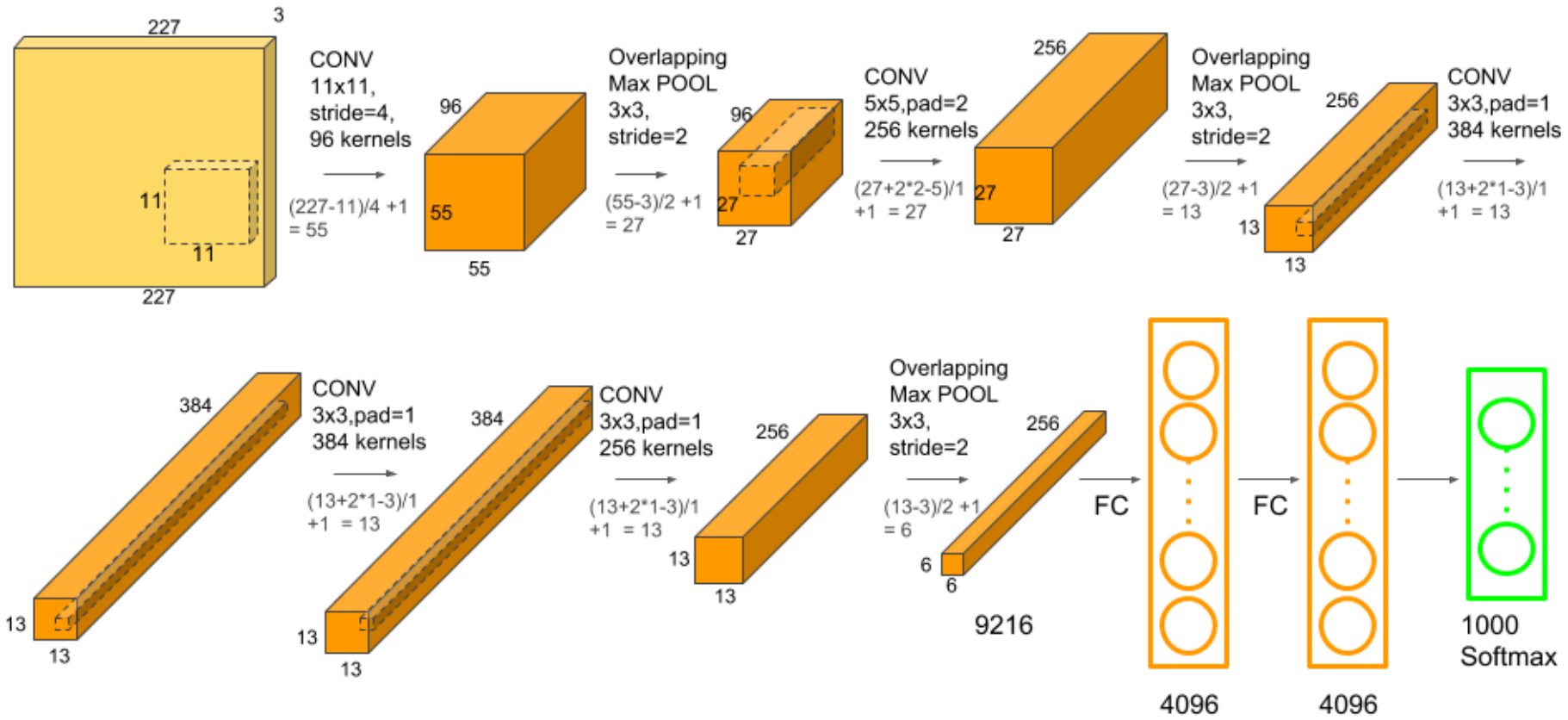
CNN手写数字识别

```
import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
(x_train,y_train),(x_test,y_test) = tf.keras.datasets.mnist.load_data()
#tf.keras.datasets.fashion_mnist.load_data()
x_train = x_train.reshape((-1,28,28,1))/255.0
x_test = x_test.reshape((-1,28,28,1))/255.0
y_train = tf.keras.utils.to_categorical(y_train,10)
y_test = tf.keras.utils.to_categorical(y_test,10)
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',padding='same',input_shape=((28,28,1))))
model.add(MaxPool2D(2))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer=tf.keras.optimizers.Adam(),
    loss=tf.keras.losses.categorical_crossentropy,metrics=['accuracy'])
model.fit(x_train,y_train,batch_size=128,epochs=20)
print(model.evaluate(x_test,y_test))
```



AlexNet (8层)

1. ReLU函数
2. DropOut 0.5（FC）
3. 重叠池化减少了top-5和top-1错误率的0.4%和0.3%



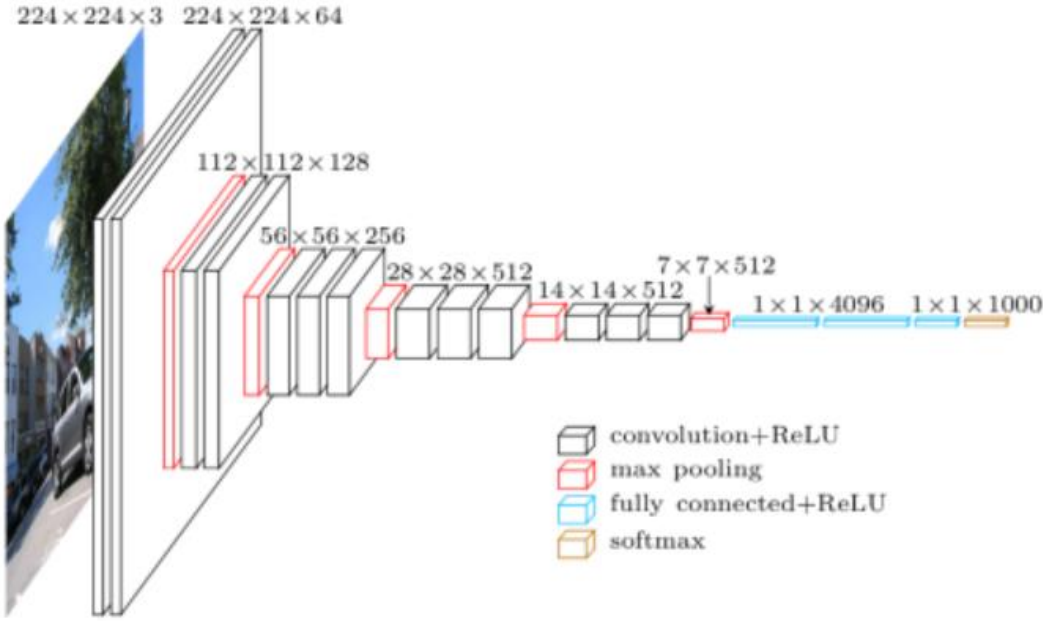
Conv-Pool-Conv-Pool-Conv-Conv-Conv-Pool-FC-FC-Softmax

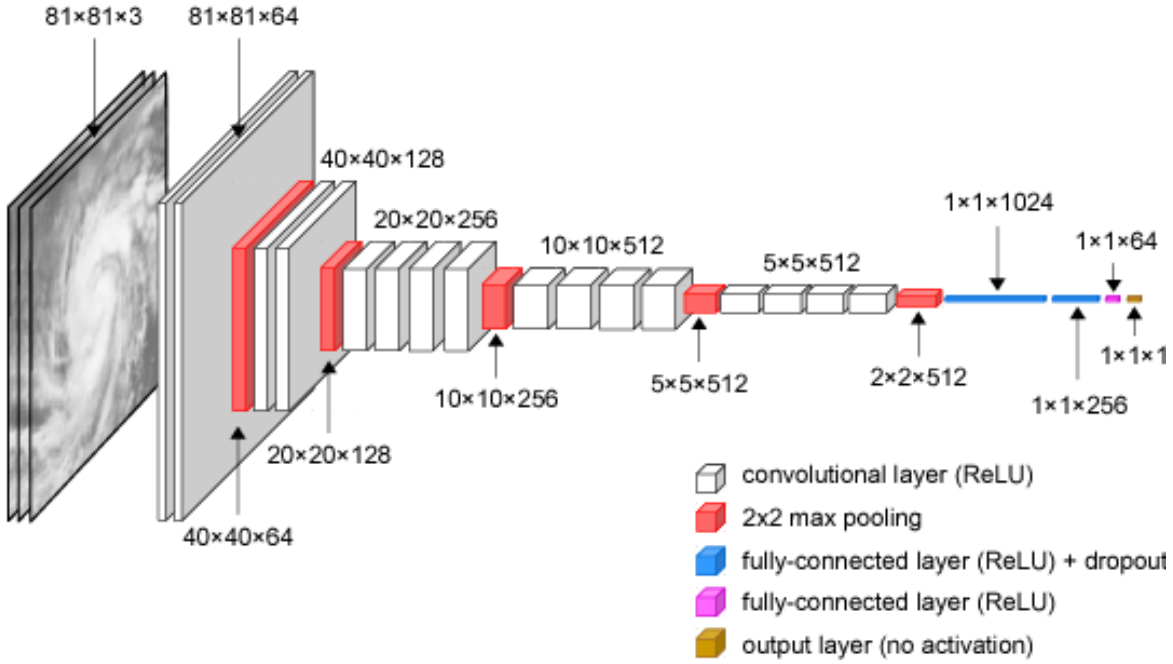
VGG16和VGG19

牛津视觉几何组

1. 卷积核3×3
2. 池化2×2，步长2

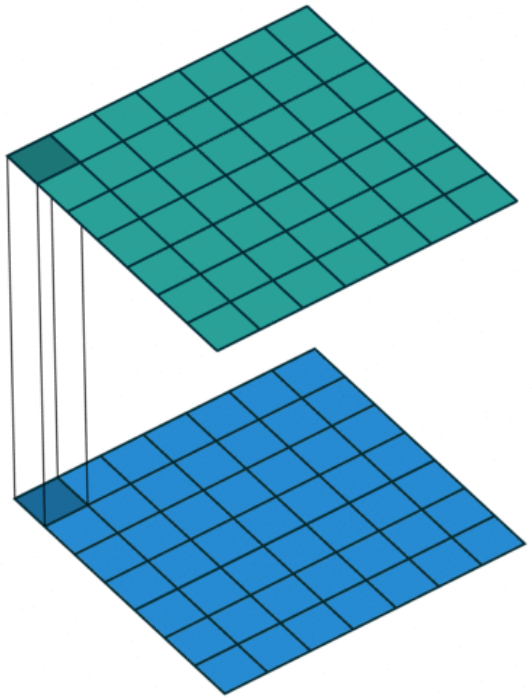
```
from tensorflow.keras.applications import vgg19
model = vgg19.VGG19()
from tensorflow.keras.preprocessing import image
cat = image.load_img('./cat.jpg',target_size=(224,224))
cat.show()
cati = image.img_to_array(cat)
cati = cati.reshape((1,)+cati.shape)
vgg19.decode_predictions(model.predict(vgg19.preprocess_input(cati)))
```



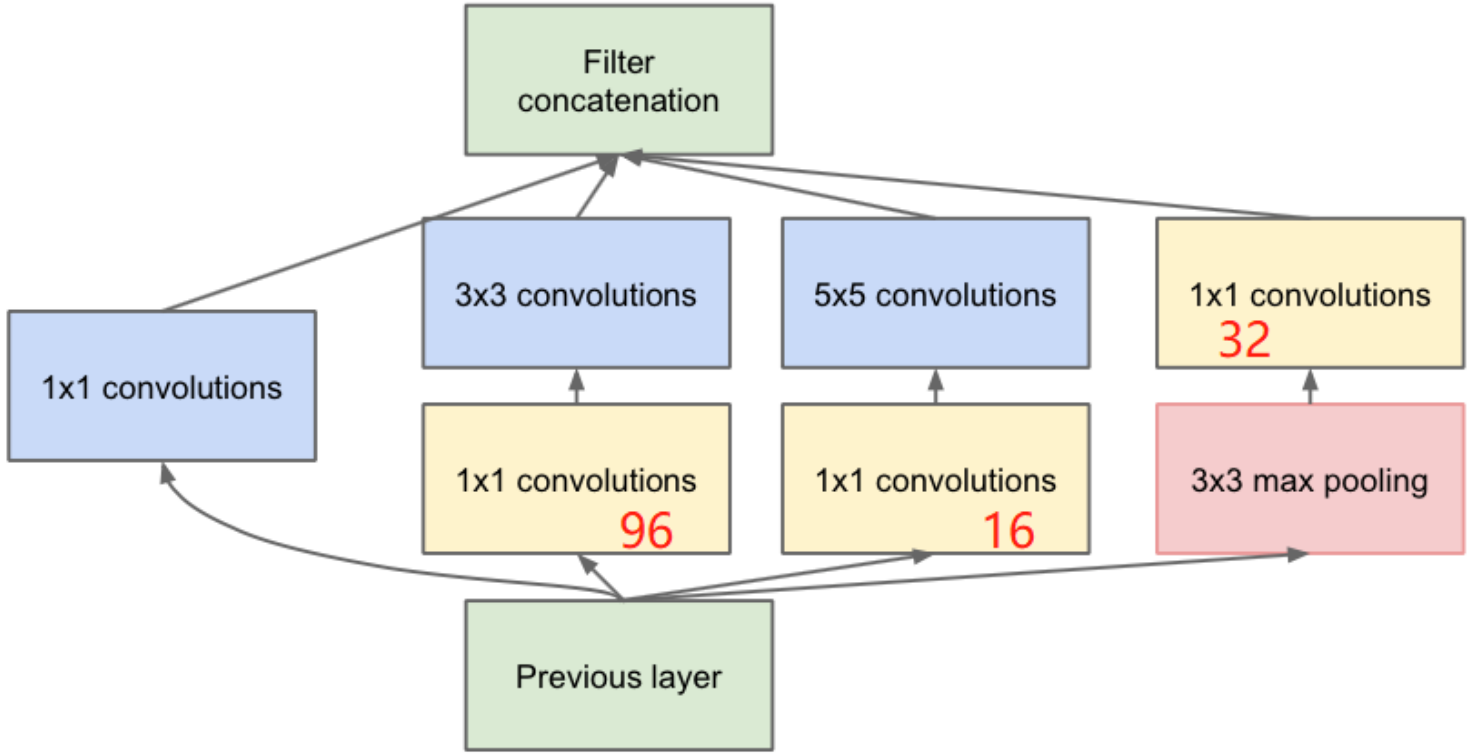



```
from tensorflow.keras.applications.vgg16 import VGG16
model = VGG16(weights='imagenet', include_top=False)
```

GoogleNet Inception V1

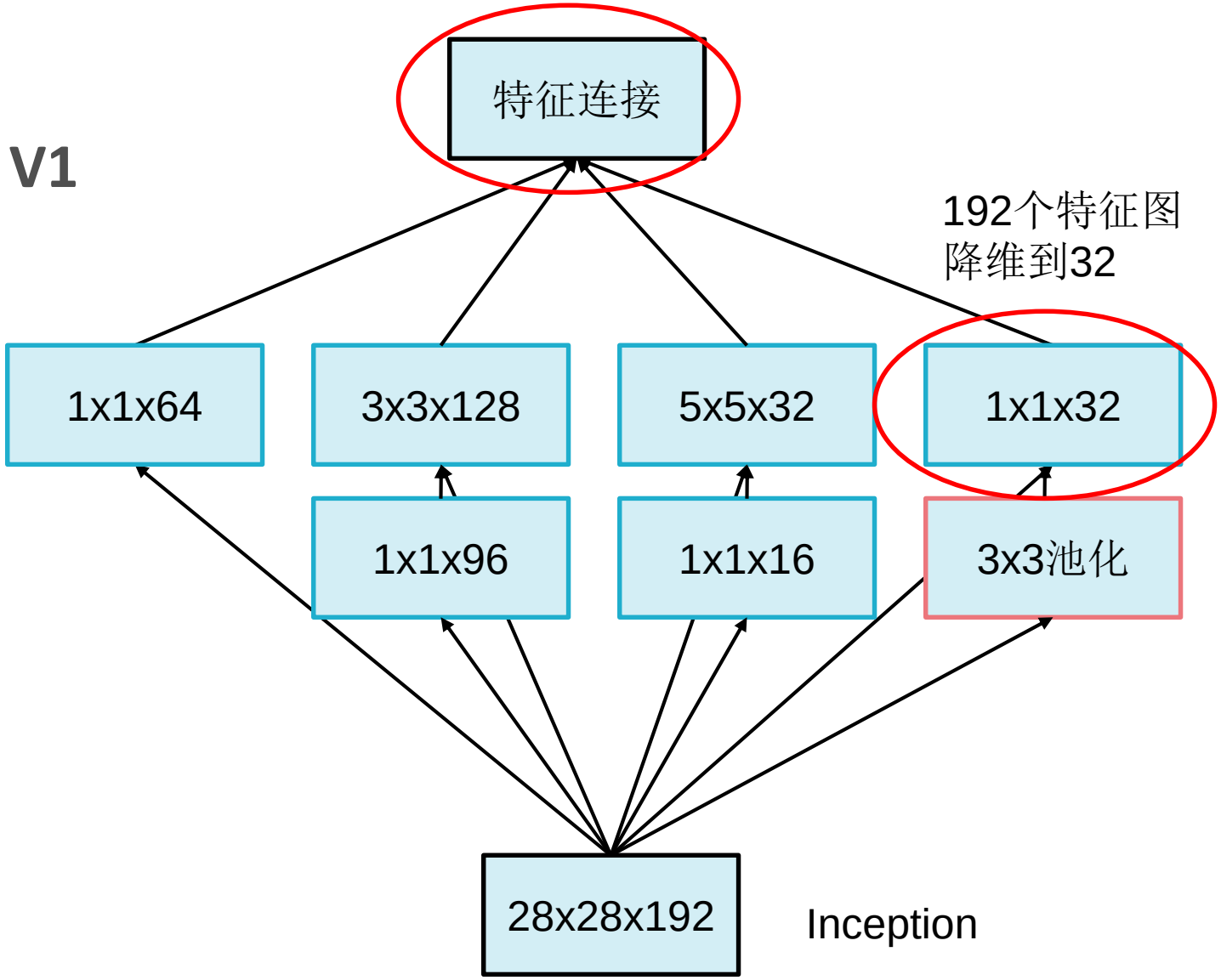


tf.keras.layers.concatenate



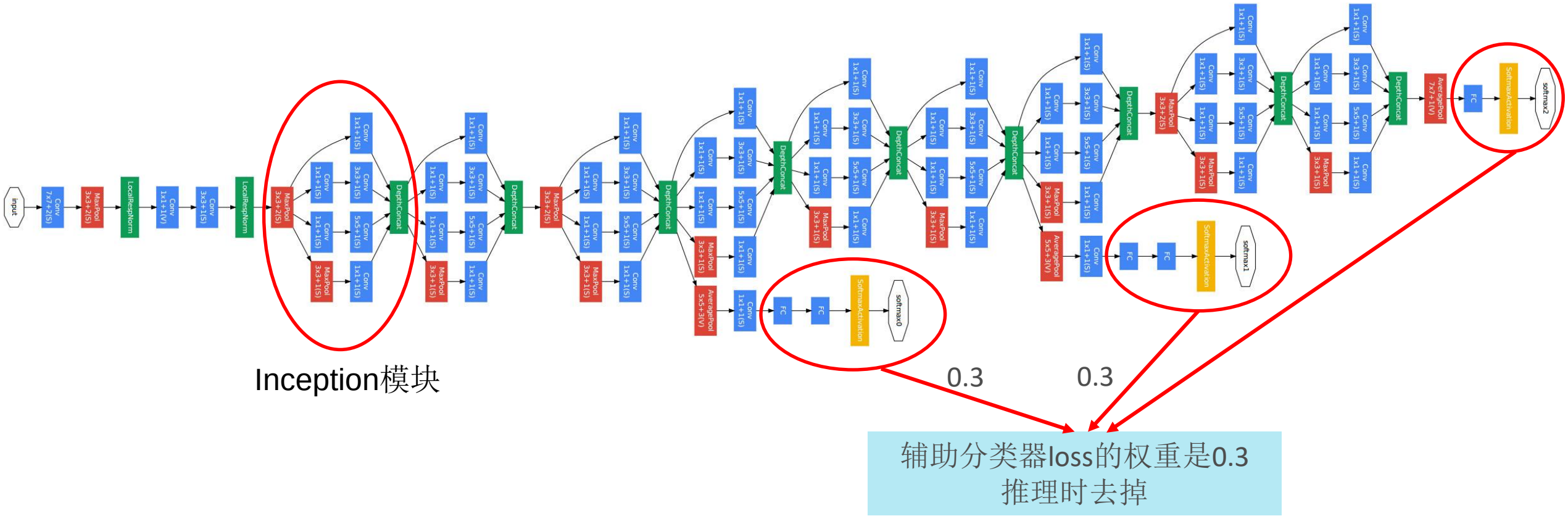
- 实现跨通道的交互和信息整合
- 可以把不同特征图组合
- 全连接可以看作1x1卷积操作

GoogleNet Inception V1



1×1×192×64+3×3×192×128+5×5×192×32= 387072

GoogleNet (22层)

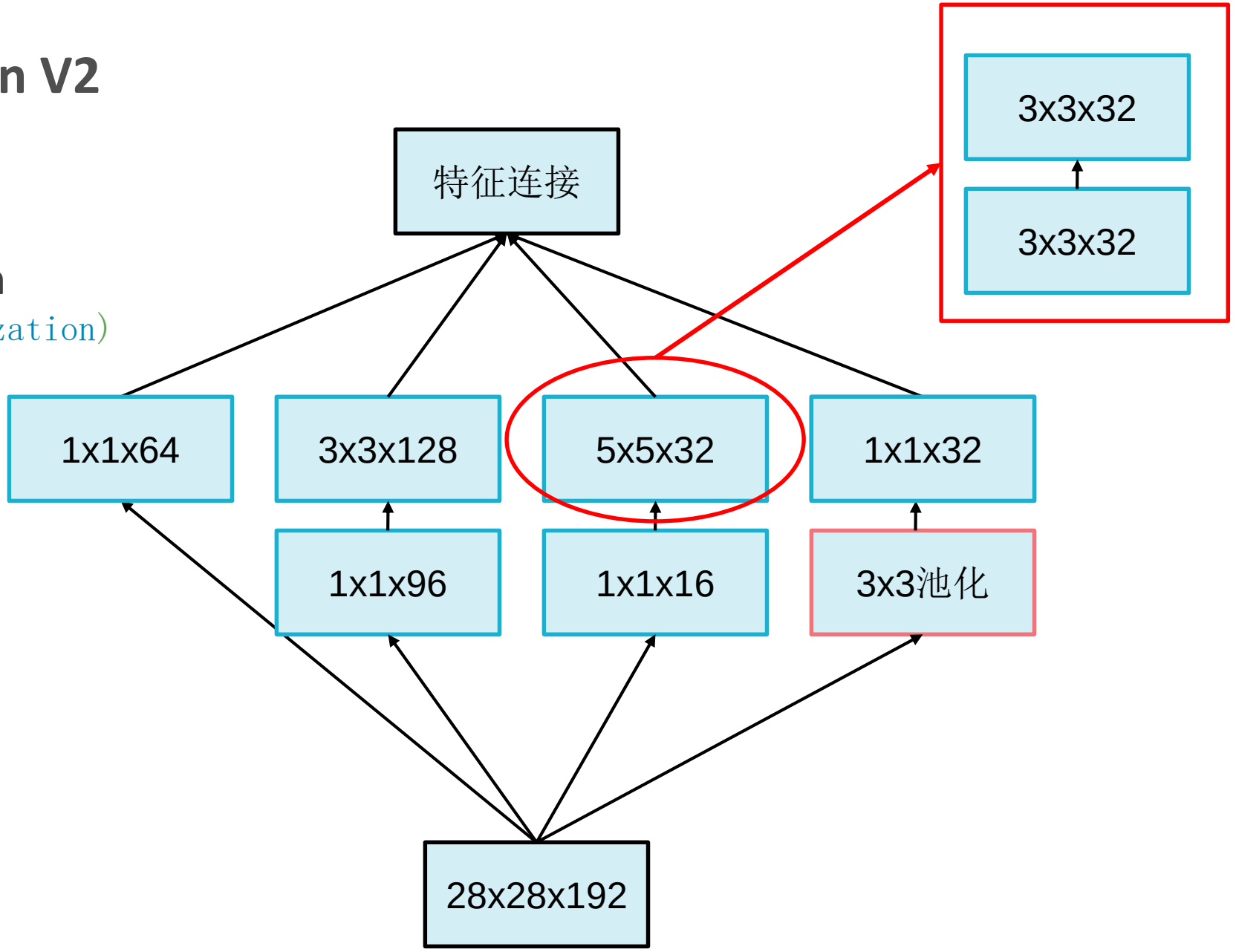


GoogleNet Inception V2

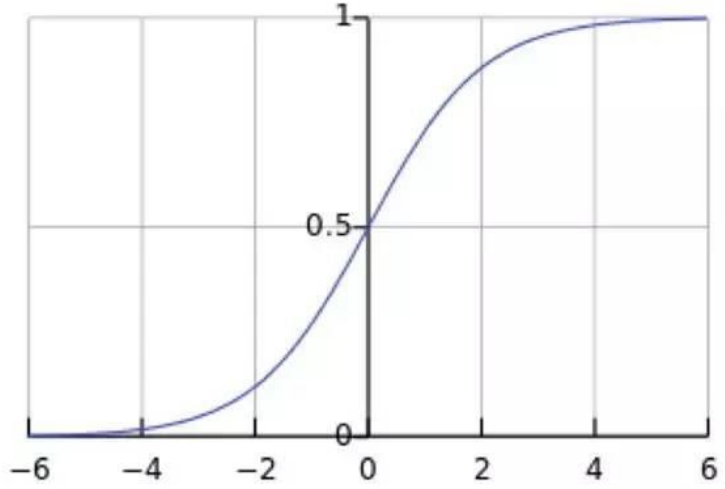
1. Batch Normalization

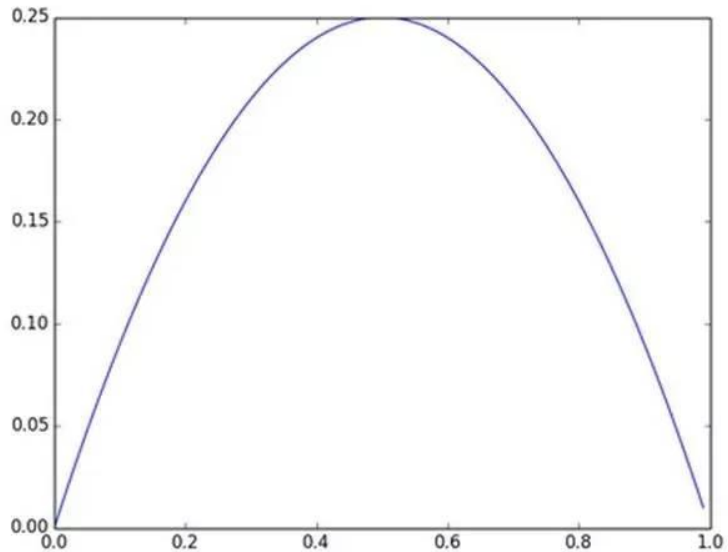
```
model.add(BatchNormalization)
```

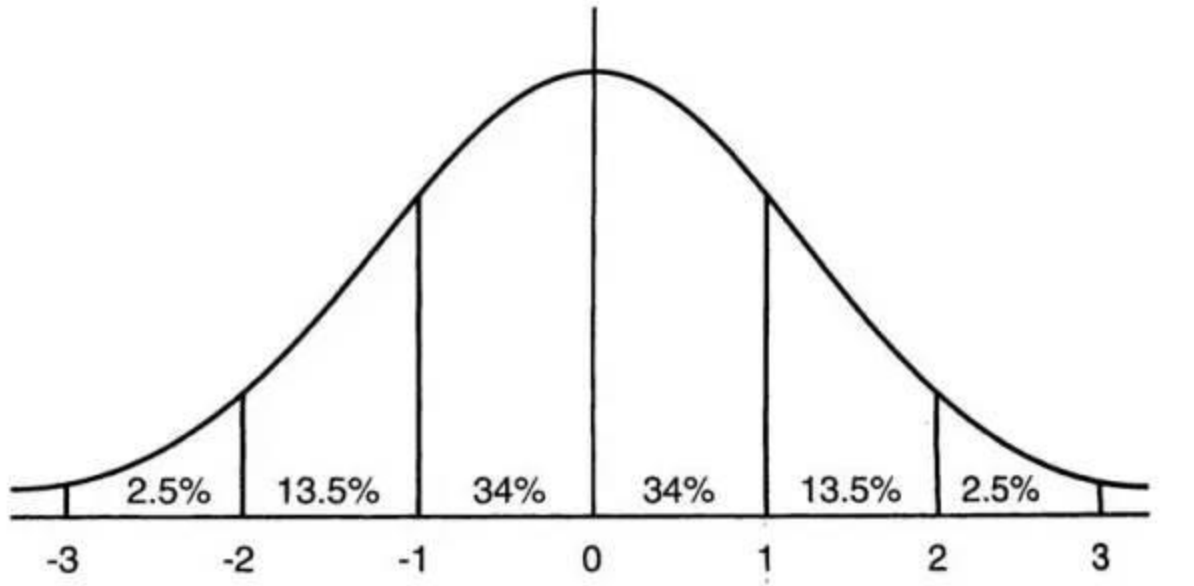
2. 3x3卷积核



Batch Normalization

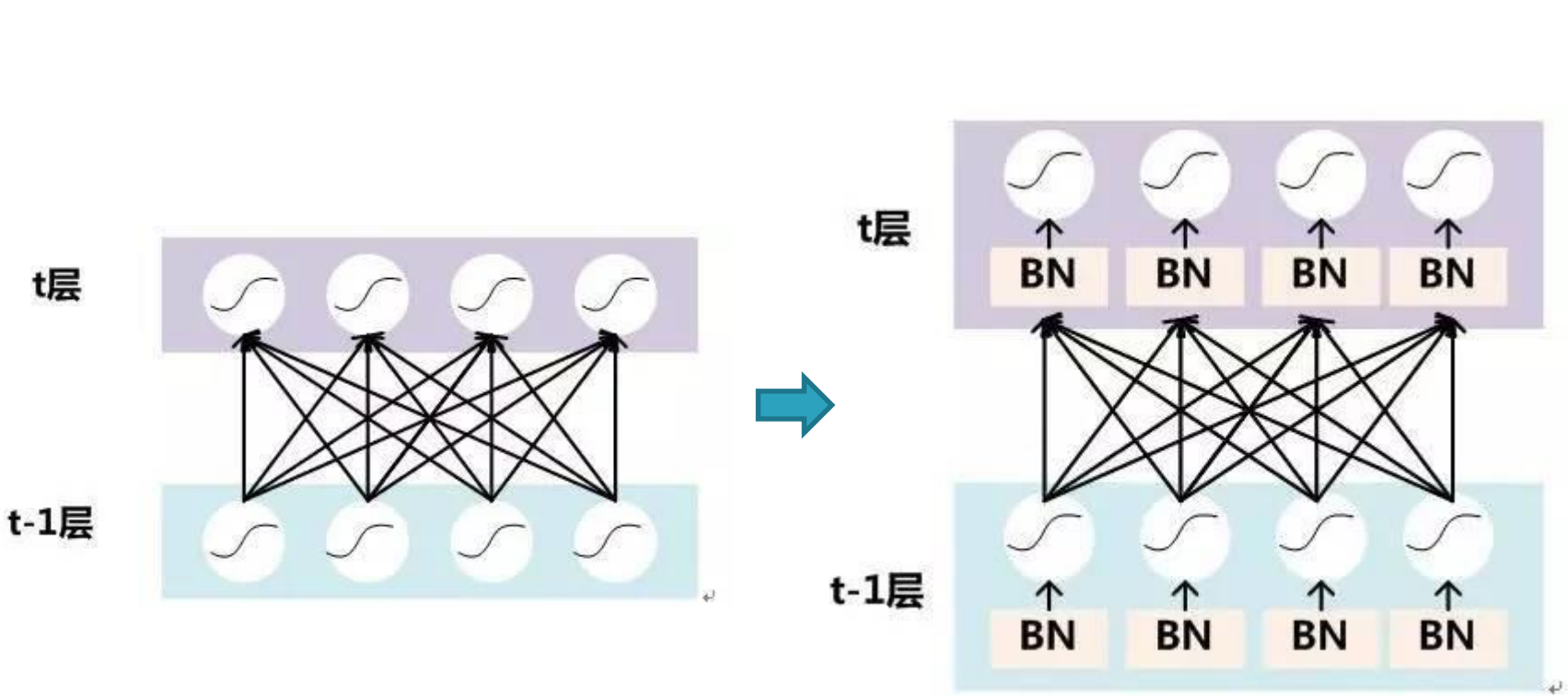


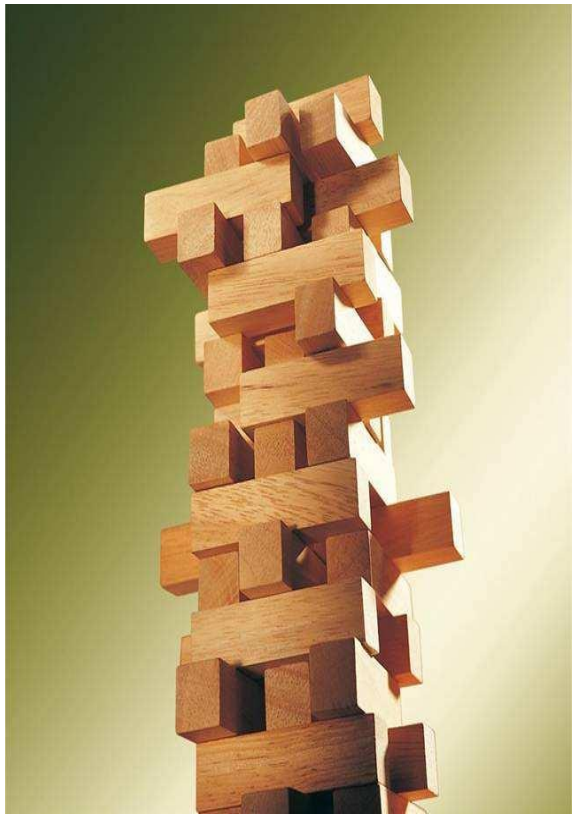




64%的概率在[-1,1], 95%的概率在[-2,2]

Batch Normalization





Batch Normalization

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i = \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$

$$z = W * x + b$$

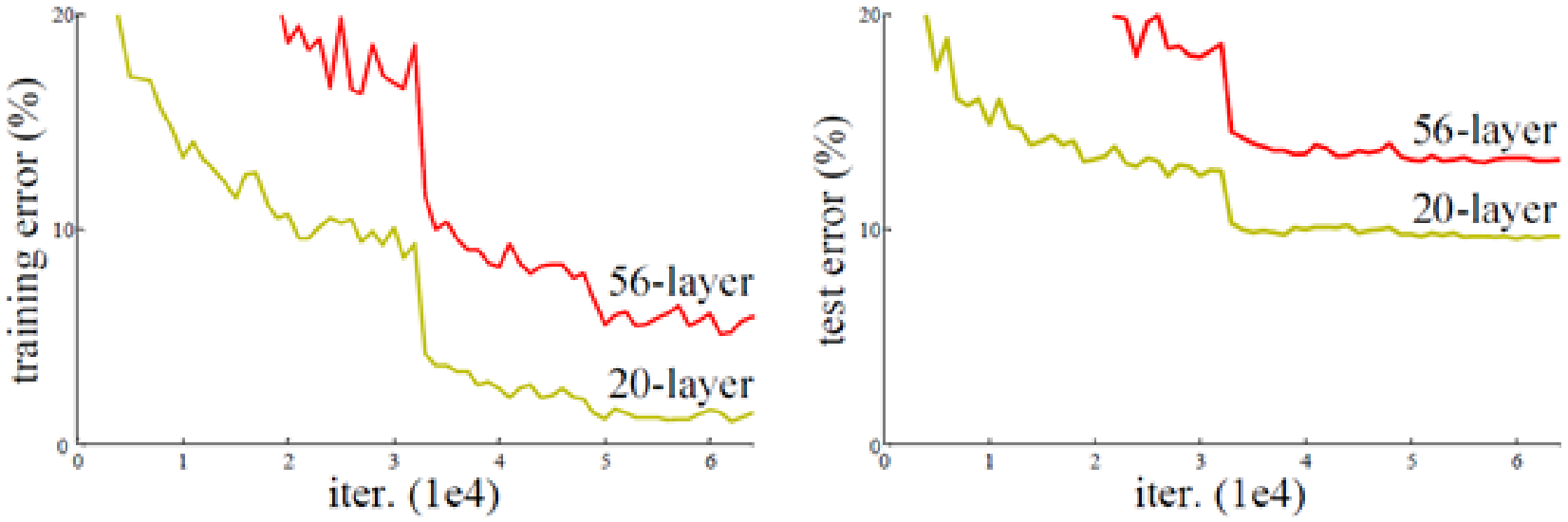
$$\mathrm{out} = \gamma \cdot \frac{z - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

$$w_{\mathrm{fold}} = \gamma \cdot \frac{W}{\sqrt{\sigma^2 + \epsilon}}$$

$$b_{\mathrm{fold}} = \gamma \cdot \frac{b - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

减少了偏置参数

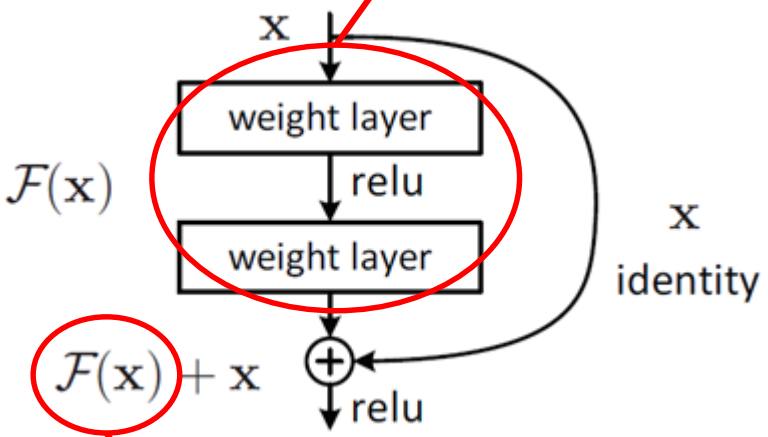
深度网络退化



深度网络退化（CIFAR-10）

ResNet (152层)

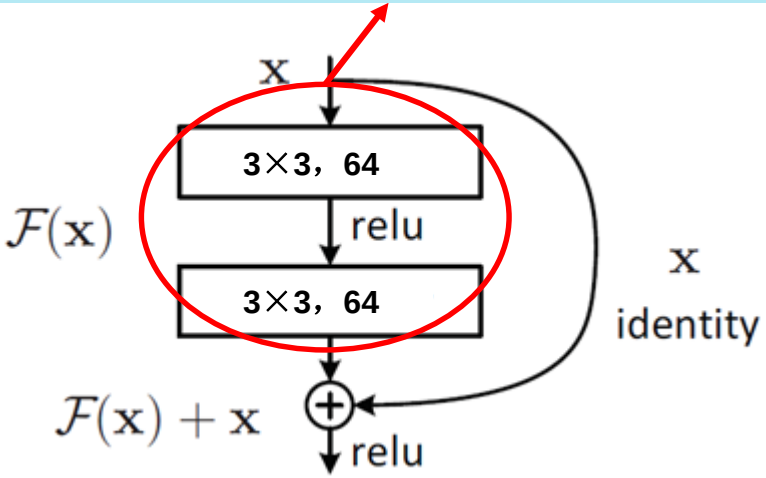
为啥要两层？



tf.keras.layers.add

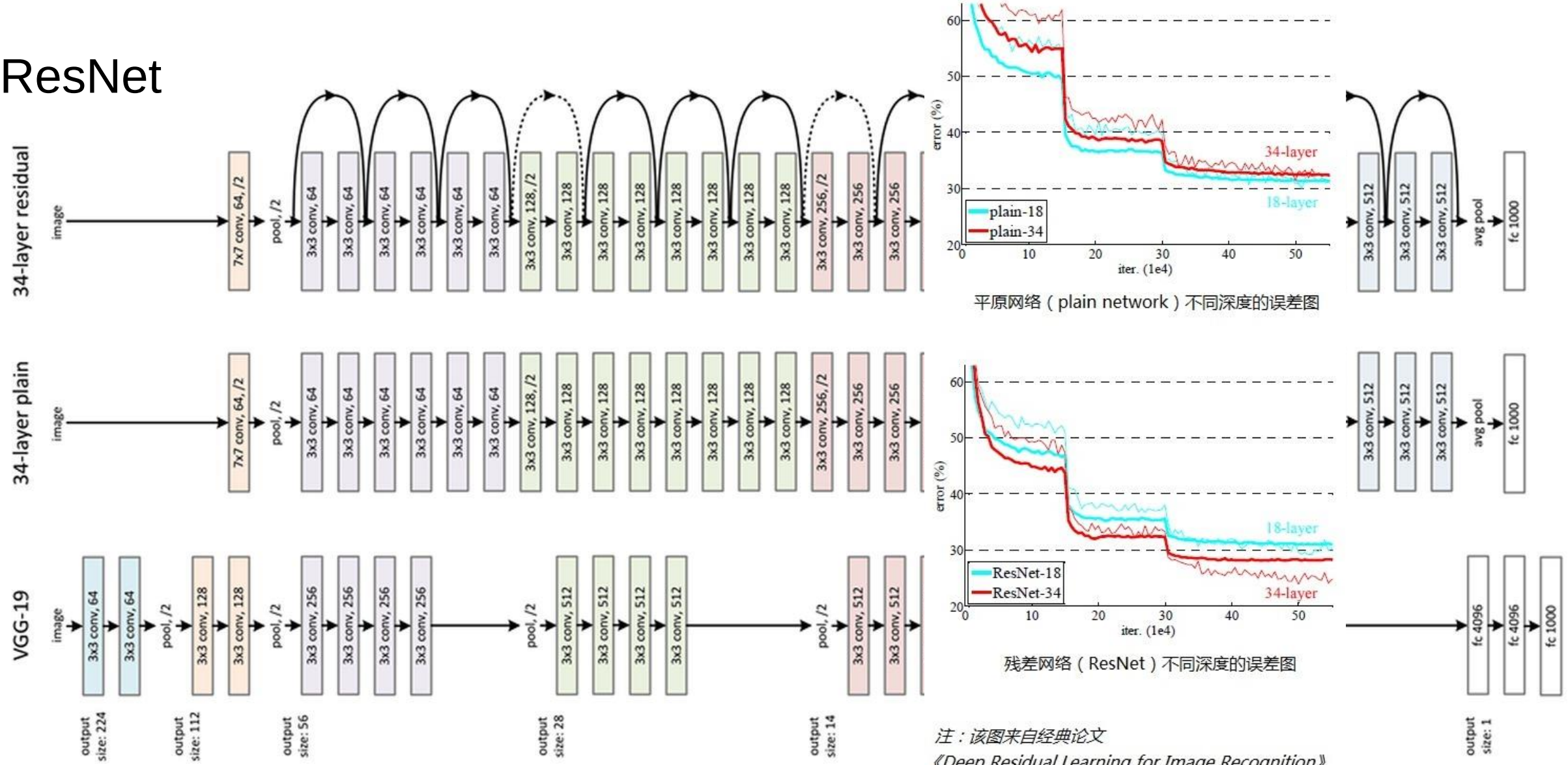
新加层恒为0时，不会使得网络变坏
因此深度网络一定可以优于浅层

1. 2个卷积层的输出与输入形状一样
2. 快捷通道采用1x1卷积变换




```
def residual_block(x: Tensor):
    y = Conv2D(64,(3,3),padding='same')(x)
    y = relu_bn(y)
    y = Conv2D(64,(3,3),padding='same')(y)
    out = Add()([x, y])
    out = relu_bn(out)
    return out
```

ResNet

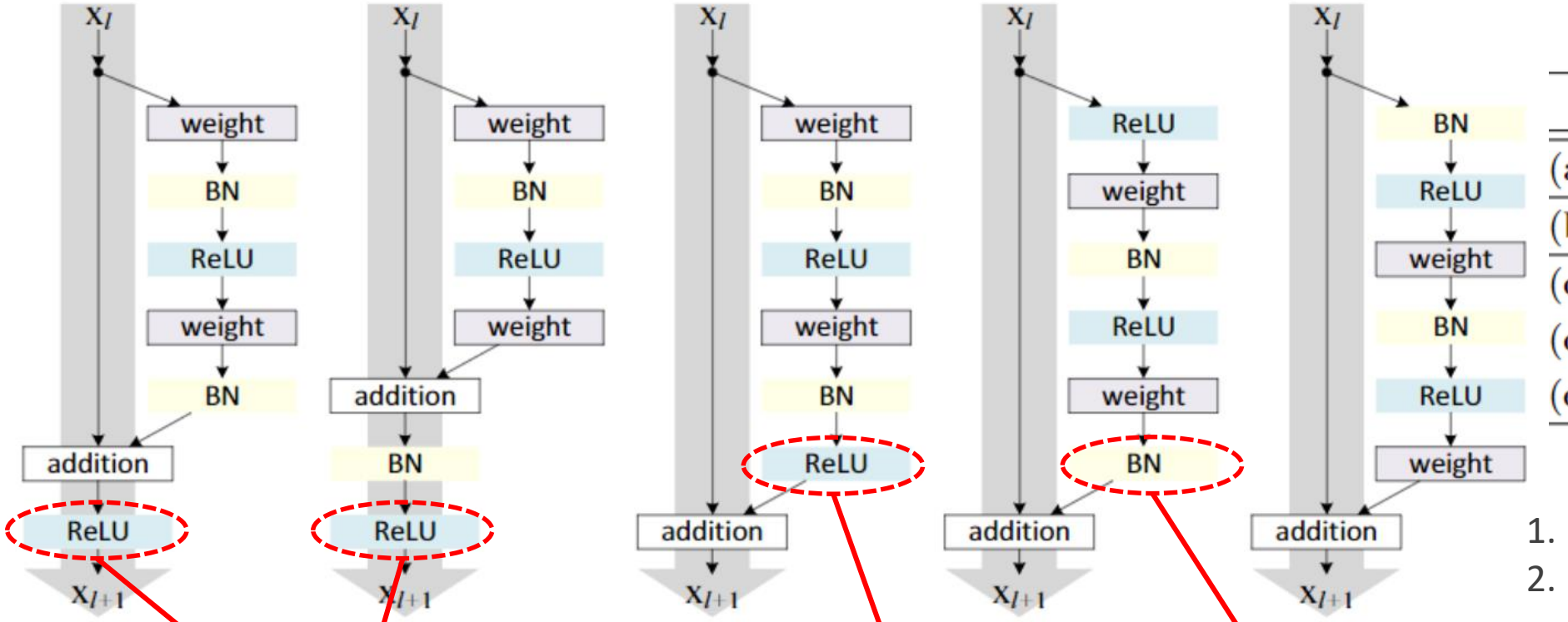


平原网络（plain network）不同深度的误差图

残差网络（ResNet）不同深度的误差图

注：该图来自经典论文
《Deep Residual Learning for Image Recognition》

ResNet



	ResNet-110	ResNet-164
(a)	6.61	5.93
(b)	8.17	6.50
(c)	7.84	6.14
(d)	6.71	5.91
(e)	**6.37**	**5.46**

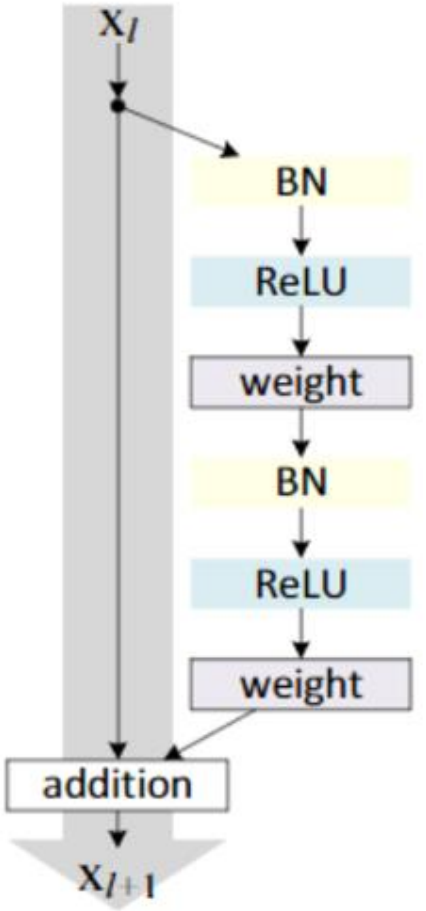
1. 少于100层原始结构可能更好
2. 大于100层畅通无阻可能更好

不加ReLU信息传递更好

ReLu放在最后
会限制网络能力

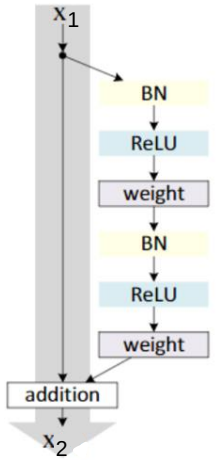
BN适合放在ReLU
前面

ResNet

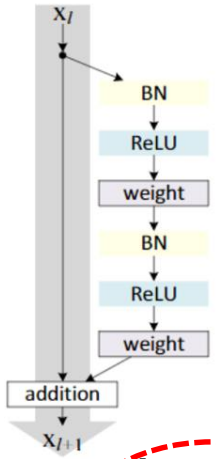


残差
$F(X^{(l)})$

$$X^{(l+1)} = X^{(l)} + F(X^{(l)})$$



$F(X^{(1)})$



$F(X^{(l)})$

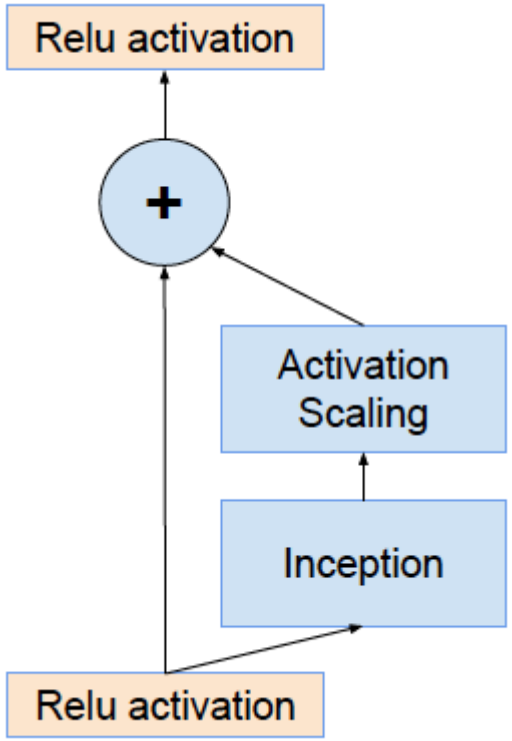
$$X^{(l+1)} = X^{(1)} + \sum_{k=1}^{l} F(X^{(k)})$$

相当于残差模型集成

$$\frac{\partial J}{\partial X^{(1)}} = \frac{\partial J}{\partial X^{(l+1)}} \frac{\partial X^{(l+1)}}{\partial X^{(1)}} = \frac{\partial J}{\partial X^{(l+1)}} \left(1 + \sum_{k=1}^{l} \frac{\partial F(X^{(k)})}{\partial X^{(1)}}\right)$$

改善梯度消失问题
信息前后向传播更加顺畅

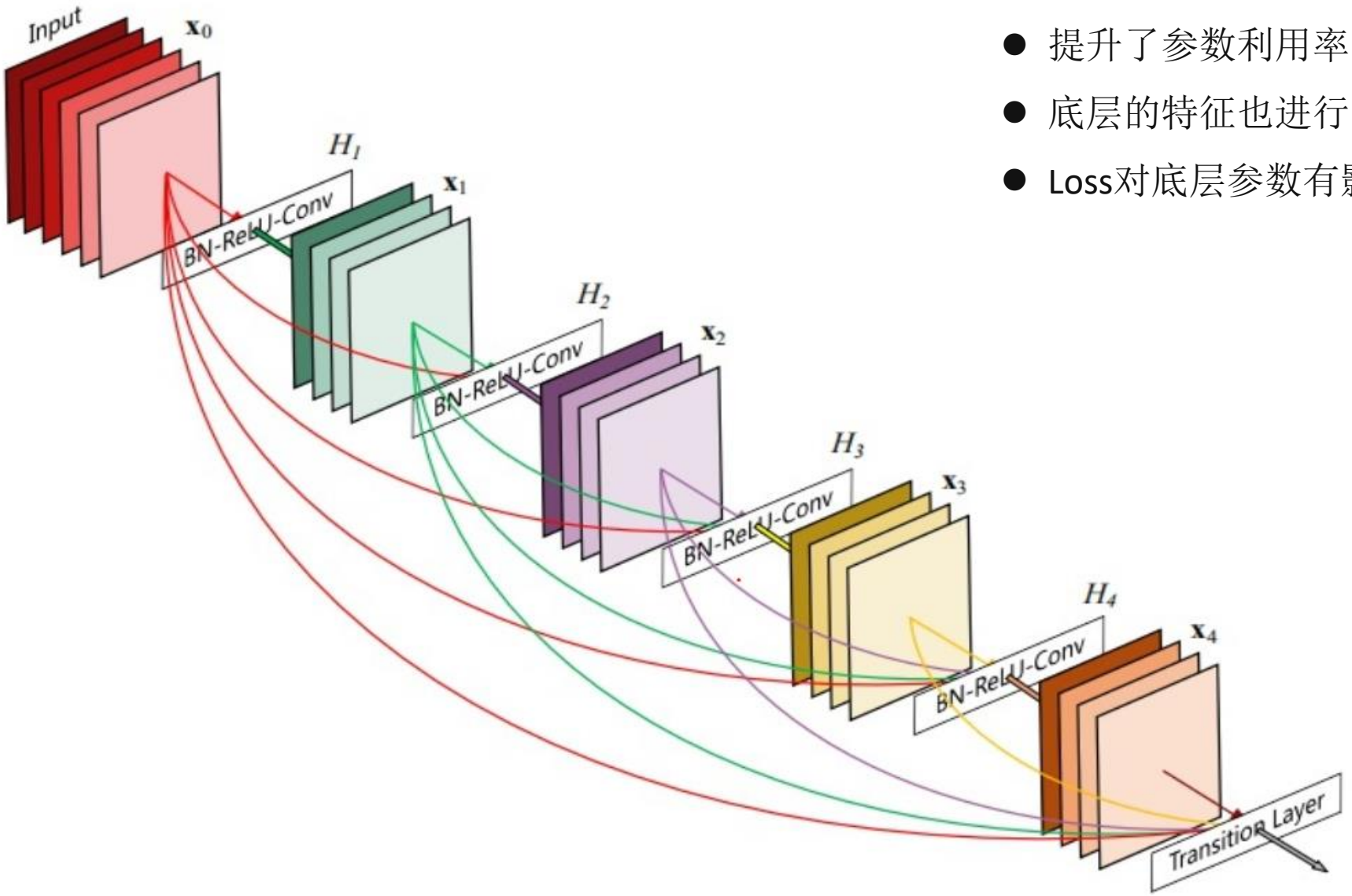
GoogleNet Inception-ResNet




```
def forward(self, x):
    out = self.conv2d(x)
    # 这里可以是卷积层、可以是Inception模块等任意sub-network
    out = out * self.scale + x  # 乘以一个比例系数0.1再相加
    out = self.relu(out)
    return out
```

1. Inception更宽
2. ResNet更深
3. Inception-ResNet又宽又深

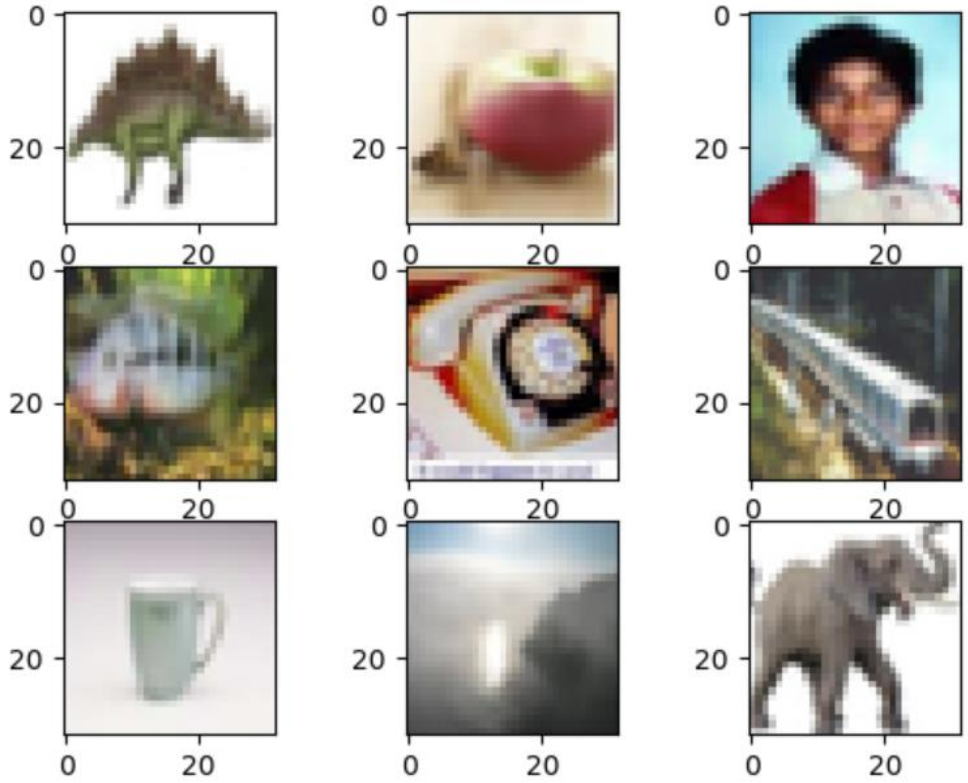
DenseNet



- 提升了参数利用率
- 底层的特征也进行了充分的传递，泛化好
- Loss对底层参数有影响，容易训练

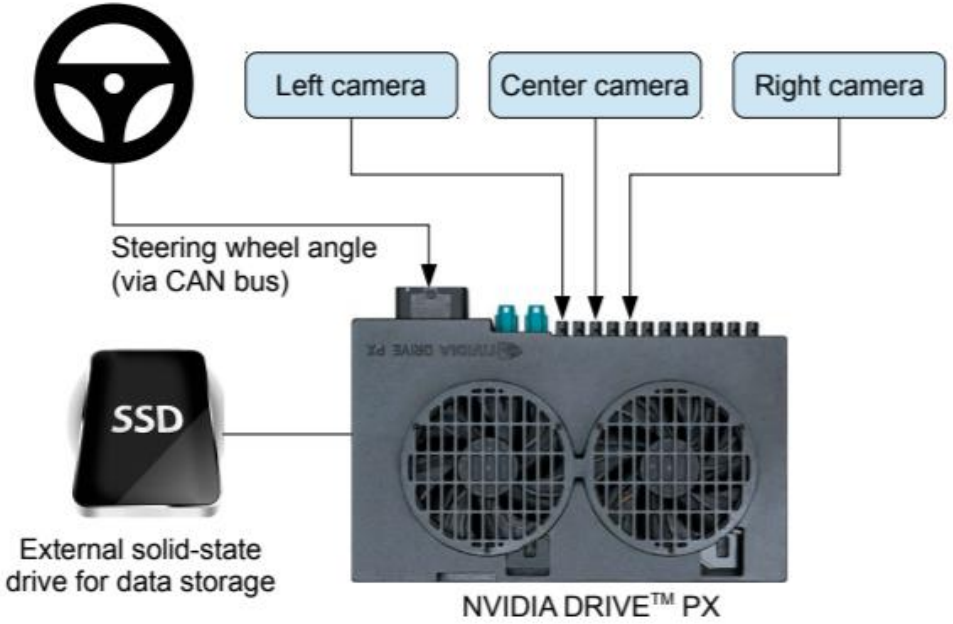
CIFAR-100实战

```
from tensorflow.keras.datasets import cifar100
import matplotlib.pyplot as plt
from PIL import Image
(X_train, y_train), (X_test, y_test) = cifar100.load_data()
for i in range(1, 10):
    plt.subplot(330 + i)
    plt.imshow(Image.fromarray(X_train[i]))
plt.show()
```

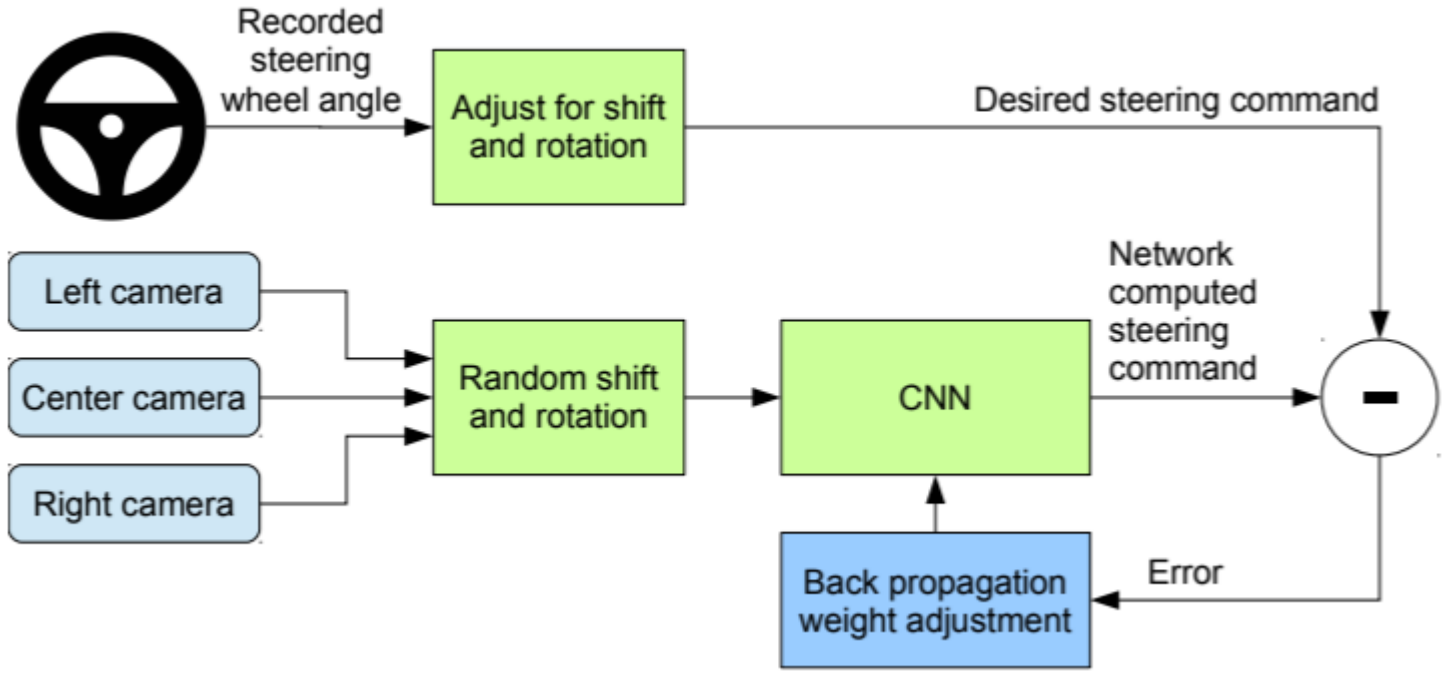


	SGD			Adam		
	NoRegular	BatchNorm	DropOut	NoRegular	BatchNorm	DropOut
VGG16	0.4273	0.6073	0.5938	0.4071	0.6232	0.5758
ResNet18	0.5714	0.637	0.6227	0.5535	0.6358	0.551
InceptionV2	0.5012	0.5615	0.4908	0.4946	0.6037	0.4948

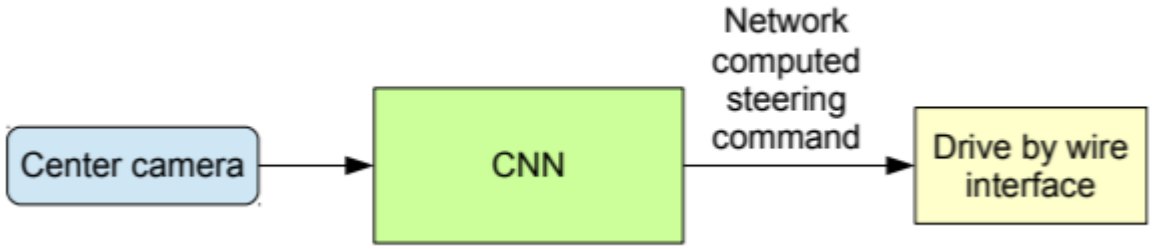
自动驾驶模型



训练模型



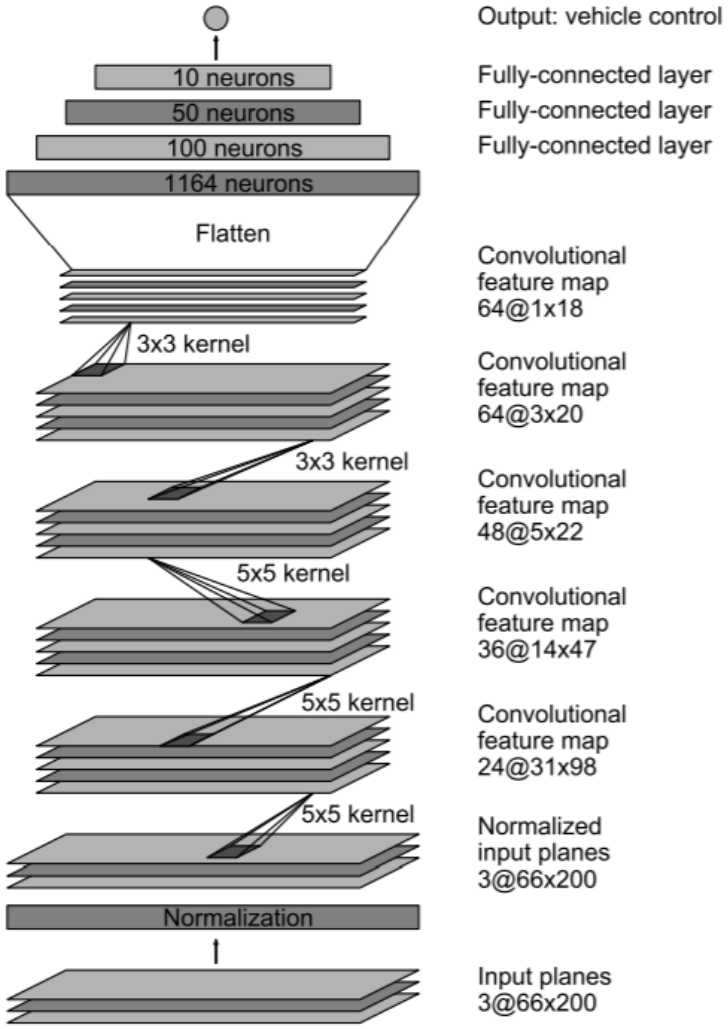
测试模型



自动驾驶模型

链接: https://pan.baidu.com/s/1c8H6fQaXzbQiJLhTZiuJgQ 提取码: nvx7 复制这段内容后打开百度网盘手机App，操作更方便哦

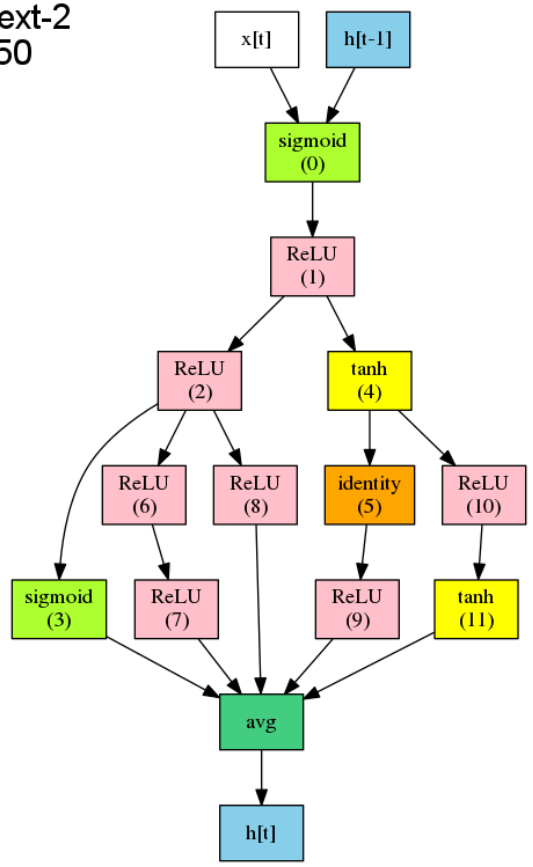
https://srikanthpagadala.github.io/serve/carnd-behavioral-cloning-p3-report.html



```
model = Sequential()
model.add(BatchNormalization(input_shape=(dshape[1], dshape[2], dshape[3])))
model.add(Convolution2D(24, 5, 5, border_mode='valid', activation='relu', subsample=(2, 2)))
model.add(Convolution2D(36, 5, 5, border_mode='valid', activation='relu', subsample=(2, 2)))
model.add(Convolution2D(48, 5, 5, border_mode='valid', activation='relu', subsample=(2, 2)))
model.add(Convolution2D(64, 3, 3, border_mode='valid', activation='relu'))
model.add(Convolution2D(64, 3, 3, border_mode='valid', activation='relu'))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(50, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(1))
model.compile(loss='mse', optimizer=adam(lr=0.0001))
```

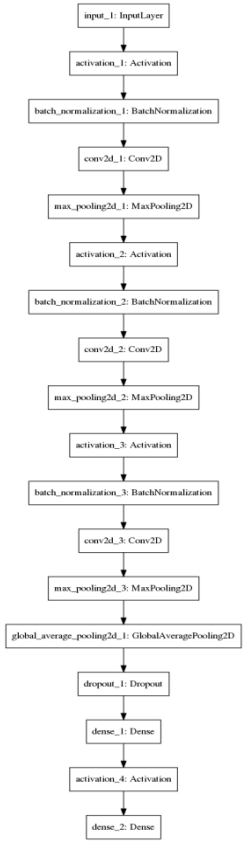
Efficient Neural Architecture Search(ENAS)

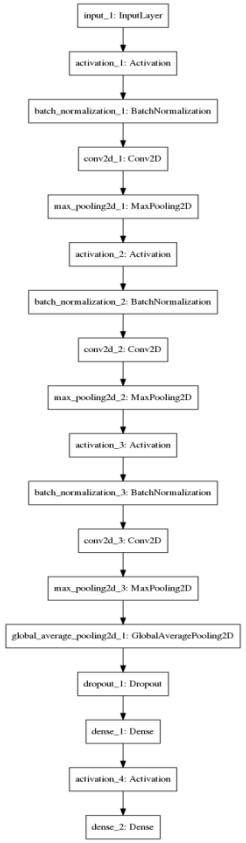
WikiText-2
step: 50




```
from keras.datasets import mnist
from autokeras import ImageClassifier
if __name__ == '__main__':
    (x_train,y_train),(x_test,y_test) = mnist.load_data()
    x_train=x_train.reshape(x_train.shape+(1,))
    x_test=x_test.reshape(x_test.shape+(1,))
    model=ImageClassifier(verbose=True,augment=False)
    model.fit(x_train,y_train,time_limit=0.1*60*60)
    print(model.evaluate(x_test,y_test))
```

AutoKeras





	LeNet5	0.1 hour	0.5 hour	0.8 hour
参数个数	107,786	79,822	153,870	11,535,114
预测精度	0.9919	0.9918	0.9934	0.9964

MCDNN, Ciresan et al. , CVPR 2012, 0.9977

Thank you!

Questions?