遥感图像分类

1.4 GoogLeNet

@tm9161

L4

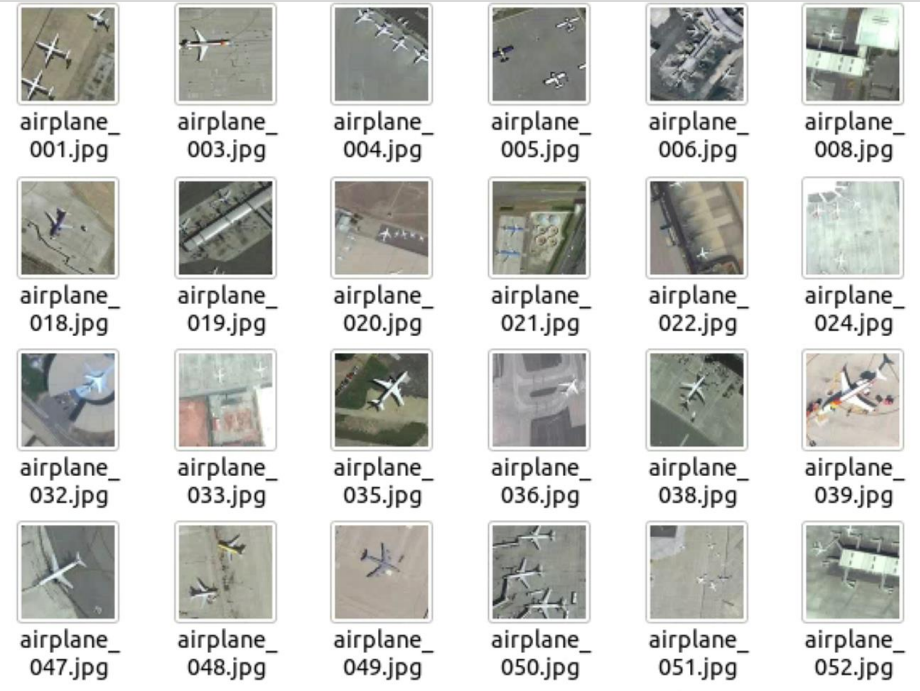
GoogLeNet

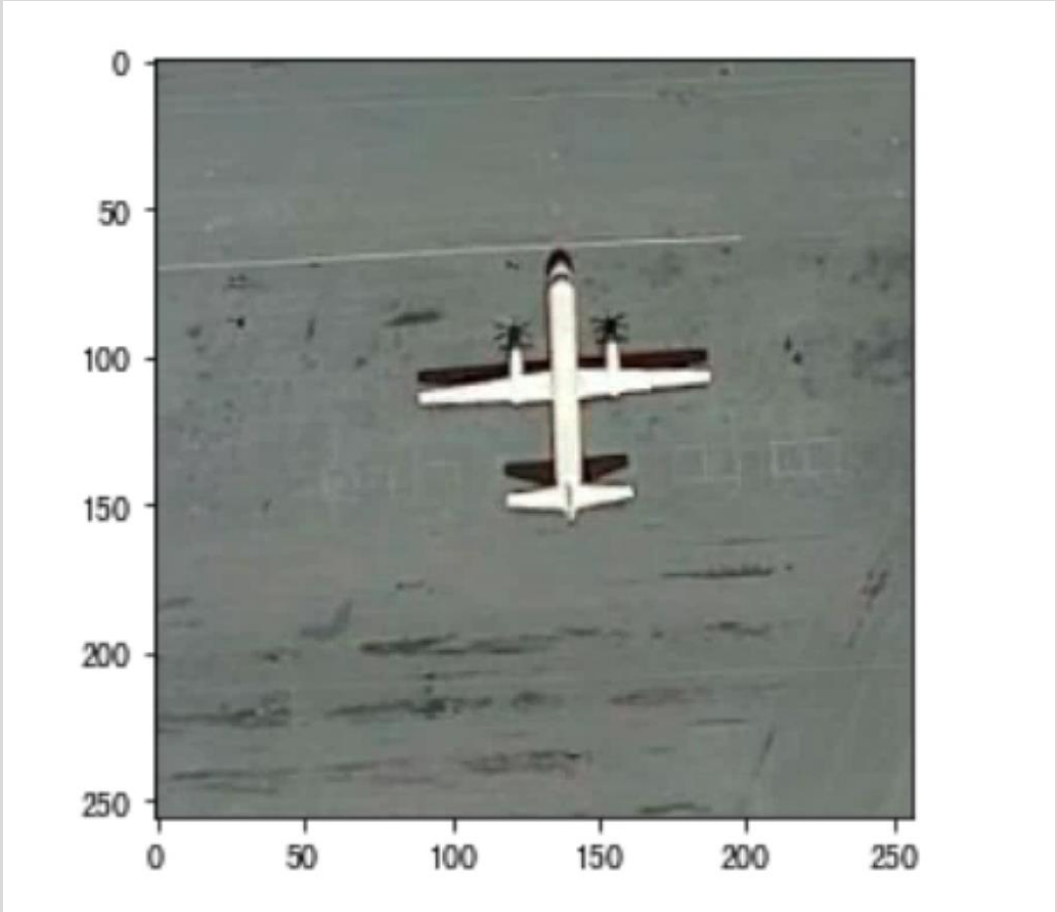
1.GoogLeNet 结构

2. GoogLeNet 创新

3. GoogLeNet 训练与预测

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.
```

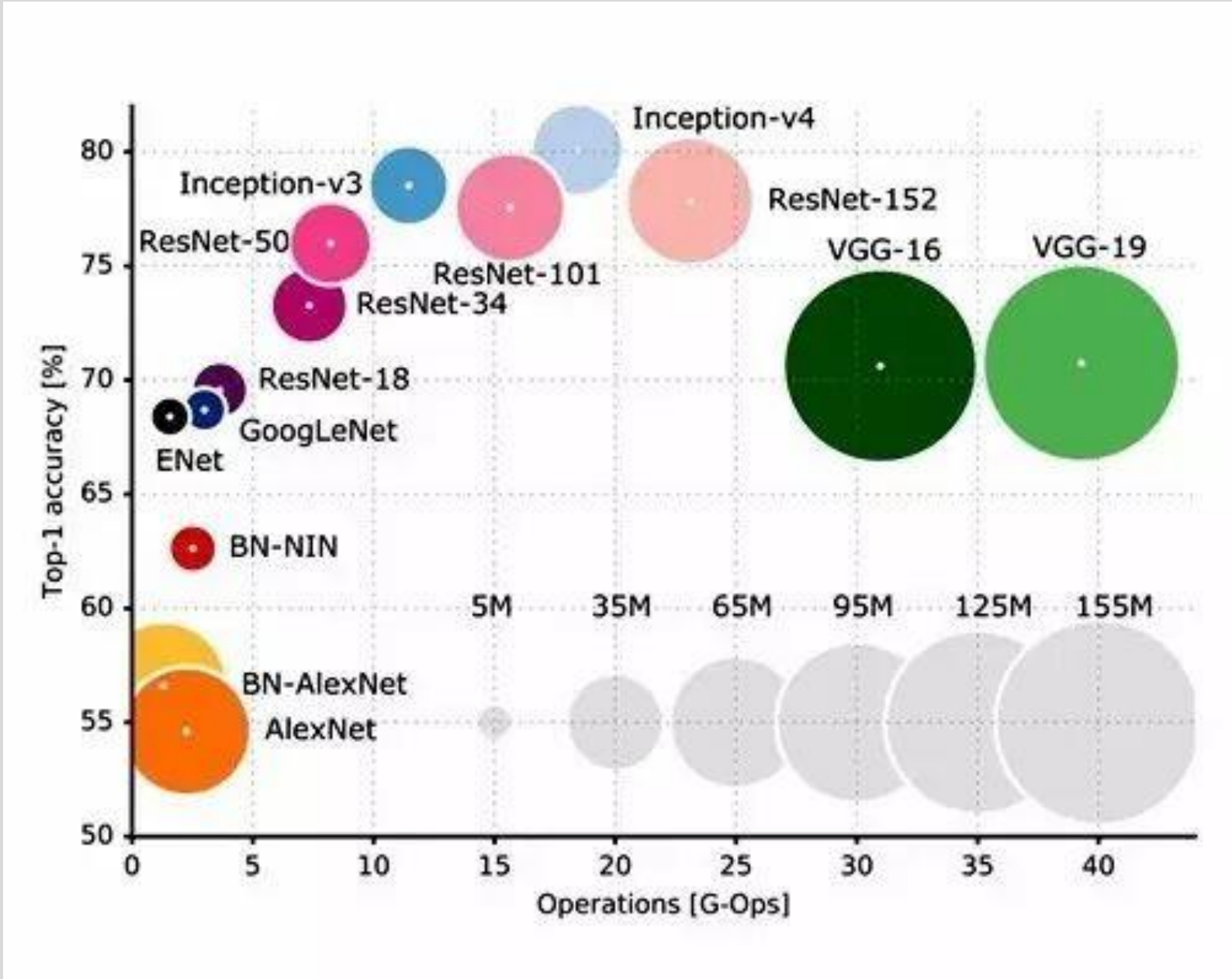
```
val_gen = test_images.flow_from_directory(directory=test_dir,
                                          batch_size=batch_size,
                                          shuffle=False,
                                          target_size=(im_size, im_size),
                                          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.
```

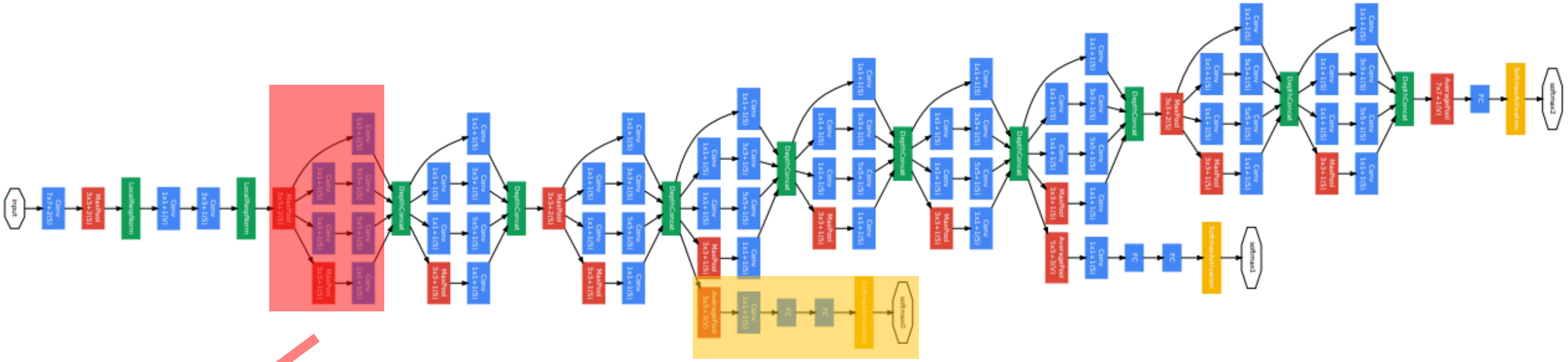
ILSVRC

GoogLeNet是在2014年由Google团队提出的，获得了当年ImageNet比赛中分类任务的**第一名**。

AlexNet、VGG等结构都是通过增大网络的深度来获得更好的训练效果，存在计算资源**消耗大**和**梯度消失**等问题，GoogLeNet提出的inception模块，**融合不同尺度**的特征信息。



GoogLeNet 结构



Inception模块

输出分支

池化层：

MaxPool 3x3+1(S)

AveragePool 5x5+3(V)

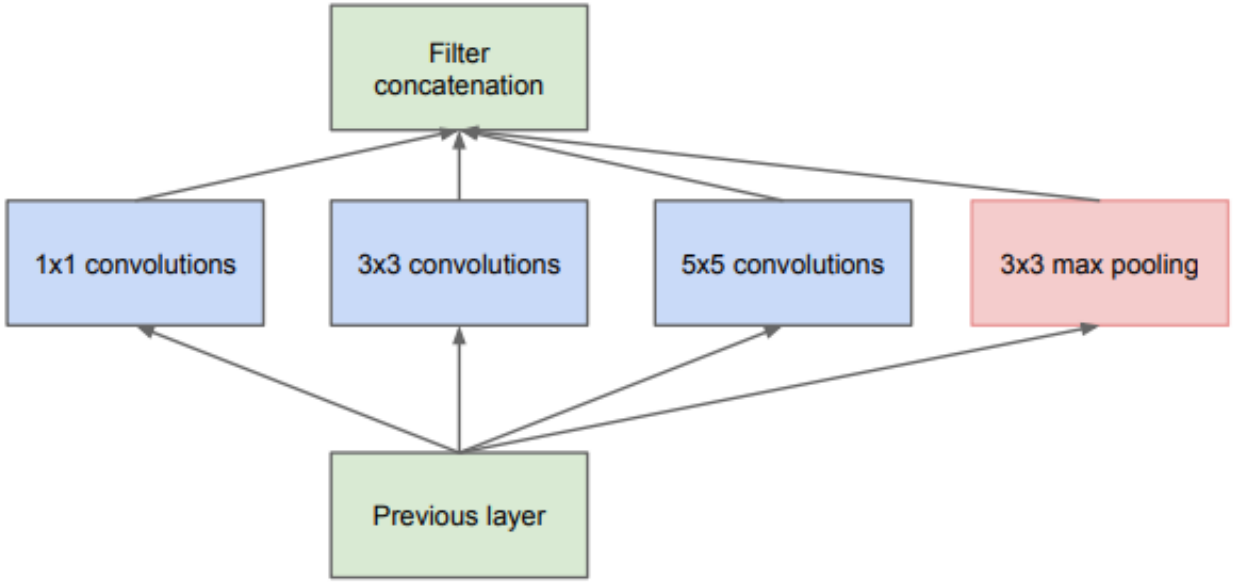
卷积层：

Conv 1x1+1(S)

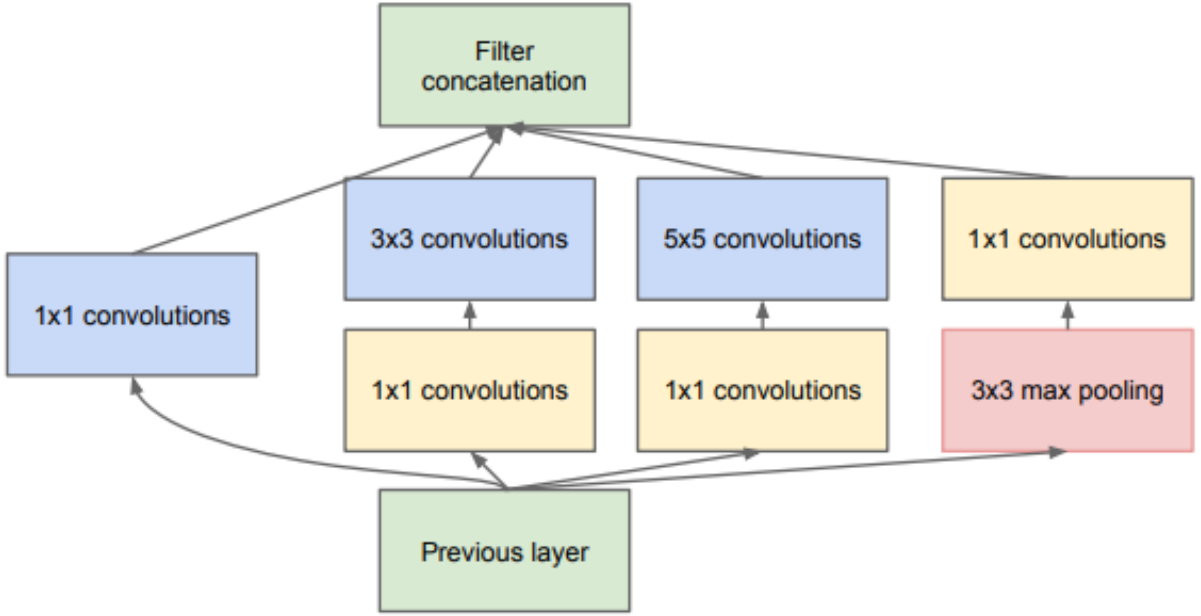
Conv 3x3+1(S)

Conv 5x5+1(S)

Inception模块



(a) Inception module, naïve version



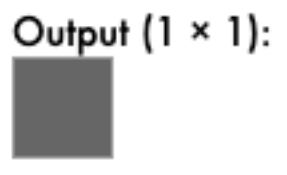
(b) Inception module with dimension reductions

输入为28*28*192（*不考虑偏置项）
直接32个5X5卷积参数：5*5*192*32=105600
先使用16个1X1卷积降维，再使用32个5X5卷积参数：1*1*192*16+5*5*16*32=15872

感受视野 Receptive Field

定义：输出层一个元素对应输入层区域的大小。
计算：感受视野=（上一层感受视野 - 1）*步长 +卷积核尺寸

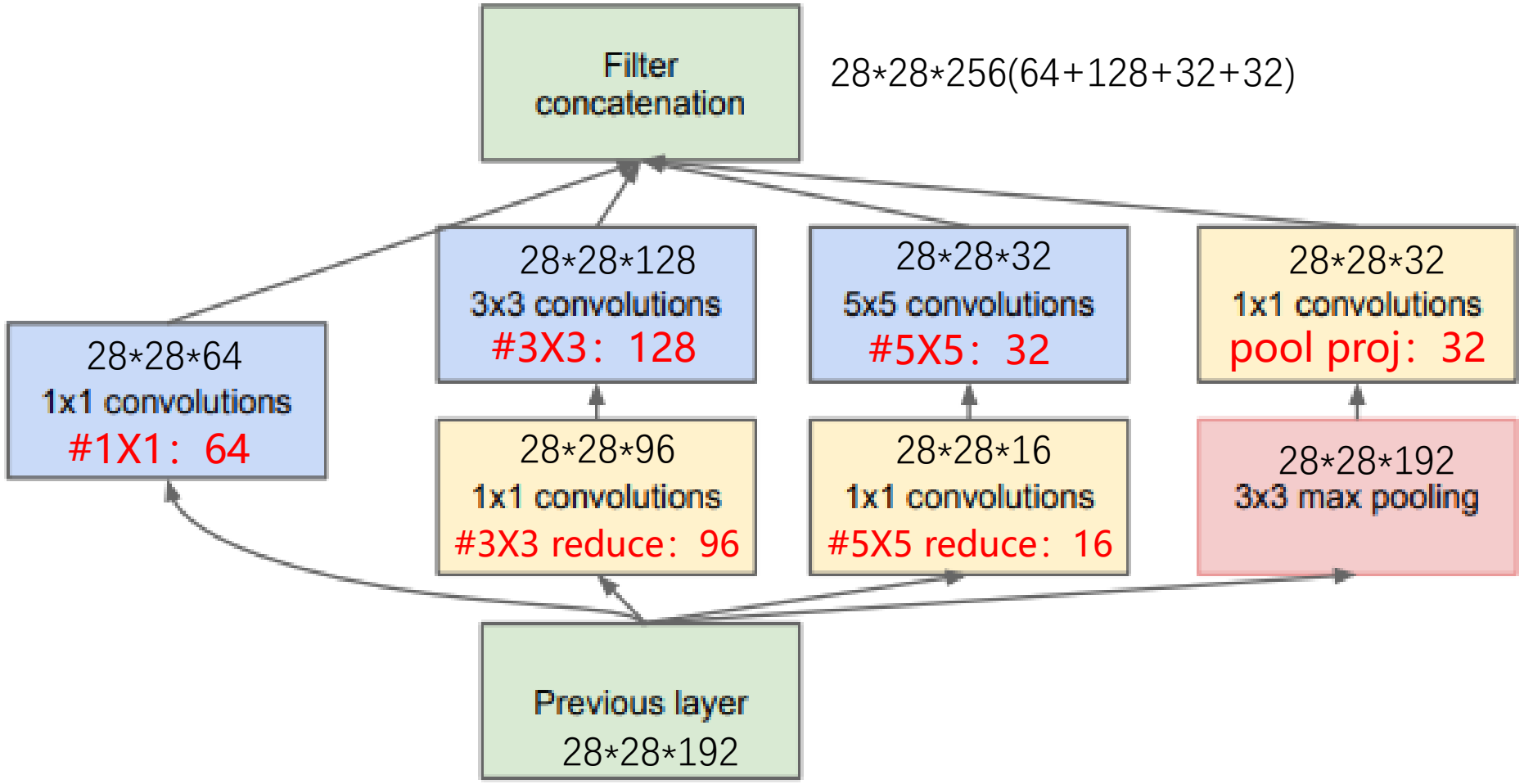
VGGNet提出：
堆叠两个3*3卷积核替代一个5*5卷积核；
堆叠三个3*3卷积核替代一个7*7卷积核。



相同感受视野，训练参数量减少。

https://ezyang.github.io/convolution-visualizer/index.html

Inception模块



type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M

Padding问题

TensorFlow中 padding = ‘same’

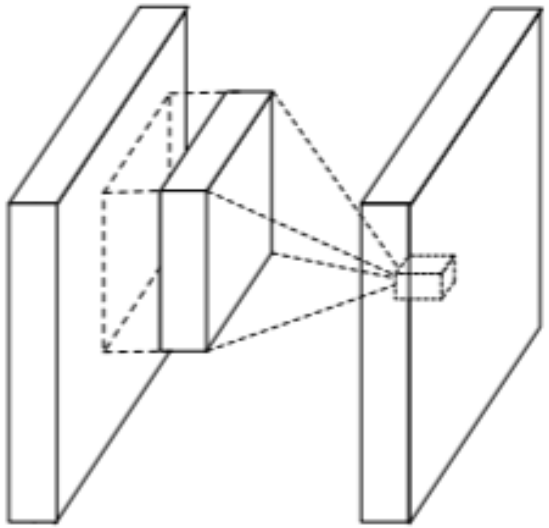
输出图像的长和宽= 输入图像 / 步长 （结果向上取整）

*如果步长为1，卷积、池化操作不改变图像的长宽。

例子：

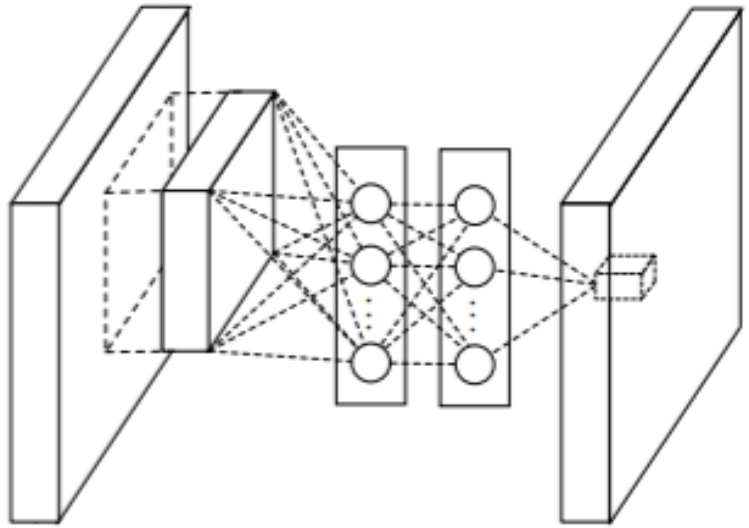
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								

NIN网络



dense (Dense)	(None, 1024)	25691136	
dropout_1 (Dropout)	(None, 1024)	0	
dense_1 (Dense)	(None, 1024)	1049600	
dense_2 (Dense)	(None, 5)	5125	

Total params: 35,966,341
Trainable params: 35,966,341
Non-trainable params: 0



average_pooling2d_2 (AveragePoo	(None, 1, 1, 1024)	0	model_10[0][0]
flatten_2 (Flatten)	(None, 1024)	0	average_pooling2d_2[0][0]
dropout_4 (Dropout)	(None, 1024)	0	flatten_2[0][0]
model_3 (Functional)	(None, 5)	2168965	model_2[0][0]
model_7 (Functional)	(None, 5)	2171013	model_6[0][0]
dense_4 (Dense)	(None, 5)	5125	dropout_4[0][0]
aux_1 (Softmax)	(None, 5)	0	model_3[0][0]
aux_2 (Softmax)	(None, 5)	0	model_7[0][0]
aux_3 (Softmax)	(None, 5)	0	dense_4[0][0]

Total params: 10,318,655
Trainable params: 10,318,655
Non-trainable params: 0

使用了全局平均池化代替全连接层，避免全连接层带来的大量训练参数。

模型搭建

```
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

```
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
```

Inception模块

```python
def Inception (ch1x1, ch3x3red, ch3x3, ch5x5red, ch5x5, pool_proj, input_):
    inputs = tf.keras.layers.Input(shape=input_.shape[1:])
    x1 = tf.keras.layers.Conv2D(ch1x1, kernel_size=1, activation="relu")(inputs)

    x21 = tf.keras.layers.Conv2D(ch3x3red, kernel_size=1, activation="relu")(inputs)
    x22 =  tf.keras.layers.Conv2D(ch3x3, kernel_size=3, padding="same", activation="relu")(x21)

    x31 = tf.keras.layers.Conv2D(ch5x5red, kernel_size=1, activation="relu")(inputs)
    x32 = tf.keras.layers.Conv2D(ch5x5, kernel_size=5, padding="same", activation="relu")(x31)

    x41 = tf.keras.layers.MaxPool2D(pool_size=3, strides=1, padding="same")(inputs)
    x42 = tf.keras.layers.Conv2D(pool_proj, kernel_size=1, activation="relu")(x41)
    outputs = tf.concat((x1, x22, x32, x42),axis=-1)

    return tf.keras.Model(inputs=inputs, outputs=outputs)
```

长宽相同，最后一个维度拼接。

```python
#3a
x = Inception(64, 96, 128, 16, 32, 32, x)(x)
```

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M

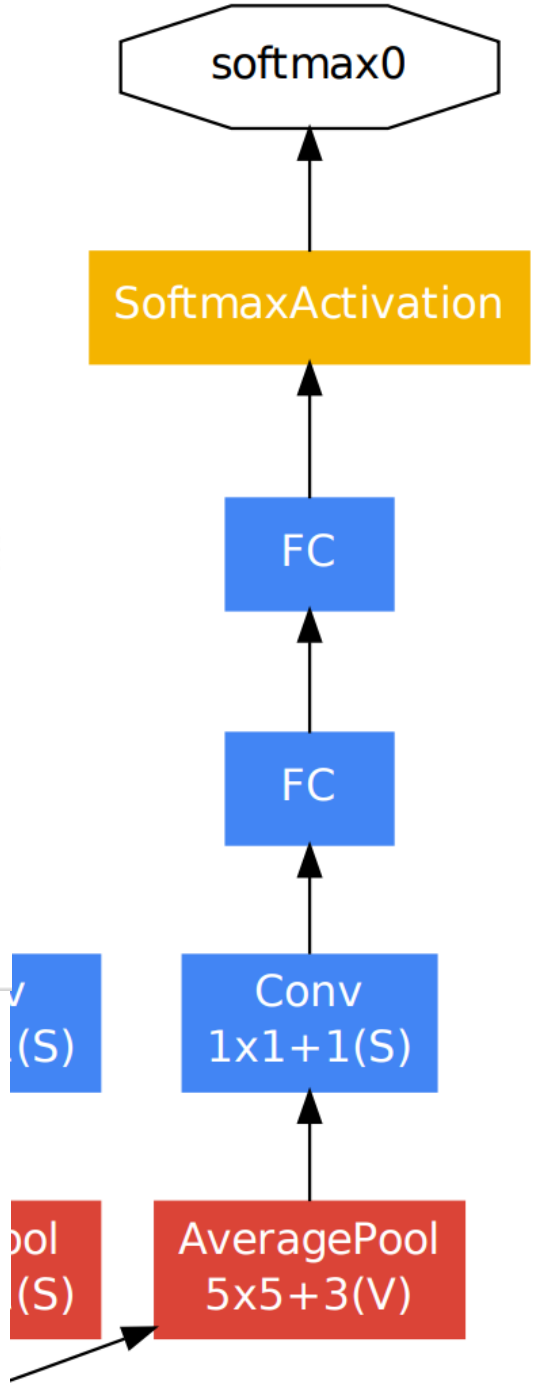
辅助分类器

```python
def InceptionAux (num_classes, input_):
    inputs = tf.keras.layers.Input(shape=input_.shape[1:])
    x = tf.keras.layers.AvgPool2D(pool_size=5, strides=3)(inputs)
    x = tf.keras.layers.Conv2D(128, kernel_size=1, activation="relu")(x)

    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dropout(rate=0.7)(x)
    x = tf.keras.layers.Dense(1024, activation="relu")(x)
    x = tf.keras.layers.Dropout(rate=0.7)(x)
    x = tf.keras.layers.Dense(num_classes)(x)

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

```python
#aux1
aux11 = InceptionAux(class_num,x)(x)
aux1 = tf.keras.layers.Softmax(name="aux_1")(aux11)
```



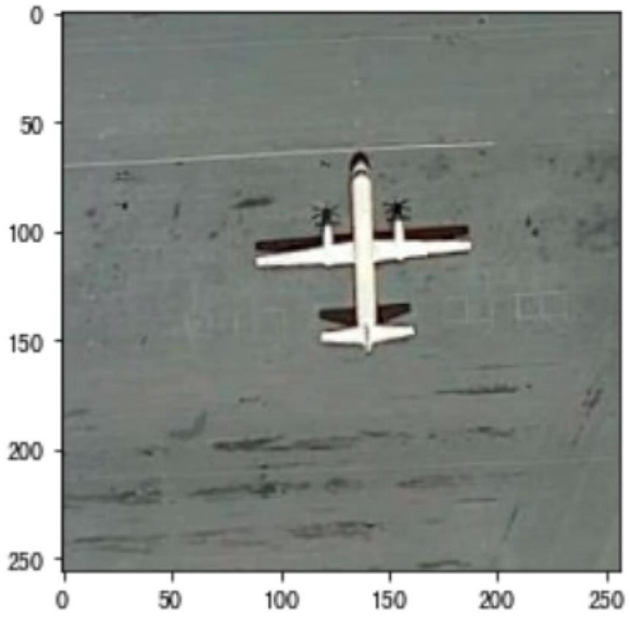
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

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

```
1 predict2 = model2.predict(img)
```

```
1 predict2
```

```
[array([[9.99999762e-01, 5.59262229e-12, 1.05985144e-10, 2.29200666e-07,
         1.62928525e-12]], dtype=float32),
 array([[1.0000000e+00, 1.5900411e-16, 2.0583806e-12, 1.8326143e-11,
         1.8823484e-12]], dtype=float32),
 array([[1.0000000e+00, 2.0445290e-16, 2.9851439e-13, 6.4077632e-10,
         2.0063951e-13]], dtype=float32)]
```

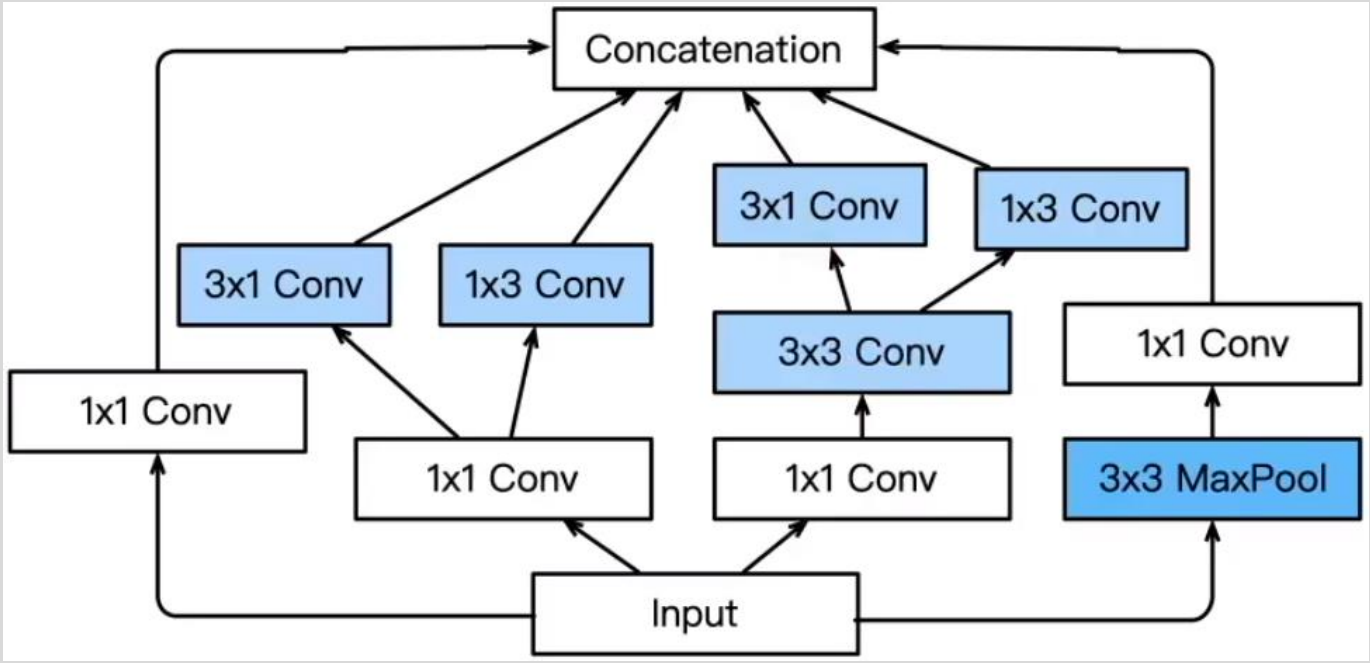
```
1 label = ['airplane','bridge','palace','ship','stadium']
```

Inception V2 V3 V4

V2、V3：

1.提出BN层：将每一层的输出都规范化到一个N(0,1)的正态分布。

2.两个3X3卷积代替5X5、1Xn和nX1非对称的卷积来代替nXn。



V4：残差结构

参考资料：

1. GoogLeNet网络详解
https://www.bilibili.com/video/BV1z7411T7ie

2.含并行连结的网络 GoogLeNet / Inception V3
https://www.bilibili.com/video/BV1b5411g7Xo

3.Going deeper with convolutions
https://arxiv.org/pdf/1409.4842.pdf

4.从零开始介绍深度学习算法和代码实现
https://courses.d2l.ai/zh-v2/

5. Network In Network
https://arxiv.org/pdf/1312.4400.pdf