Fastai Lesson 3
Understanding Convolutions

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录 Contents

windows下远程连接服务器

- （1）启动ipython，$ ipython
- （2）创建远程连接密码In [1]: from notebook.auth import passwd; passwd()
  输入两次密码，将得到一个字符串，比如'sha1:d0270e88f33b:212384922803ca9bd49d1fb6c11ab42651ccb358'，该终端不要关掉，复制该字符串，后面会用到；
- （3）生成jupyter的配置文件，$ jupyter notebook --generate-config，将会在home目录下生成一个隐藏文件夹.jupyter，该文件夹中有一个jupyter的配置文件；
- （4）打开配置文件，$ vim ~/.jupyter/jupyter_notebook_config.py，复制以下内容粘贴到配置文件中：

```
c.NotebookApp.ip = '*'
c.NotebookApp.password = u' sha1:d0270e88f33b:212384922803ca9bd49d1fb6
c11ab42651ccb358'#这里需要改为第（2）步你自己电脑上生成的密钥字符串
```

windows下远程连接服务器

- 在远程服务器上，启动jupyter notebooks服务：

  jupyter notebook --no-browser --port=8889

- 在本地机器的Terminal中启动SSH：

  ssh -N -L localhost:8888:localhost:8889 remote_user@remote_host

- 打开浏览器，输入地址：http://localhost:8888/

- 输入设置的密码就可打开jupyter notebook

Kaggle数据集下载方法

1.使用kaggle cli从kaggle下载

- 下载kaggle cli

  ~$ pip install kaggle

- 由于每当kaggle网站发生更新时，kagglecli都会中断，因此要更新kaggle cli

  ~$ pip install -U kaggle-cli

- 下载完后可从kaggle下载数据

  ~$ kg download –u <username> -p <password> -c <competition>

使用kaggle cli

实际操作如下：

```
medialab@315-1080Ti:~$ pip install kaggle-cli
Collecting kaggle-cli
Collecting beautifulsoup4<4.7,>=4.6.0 (from kaggle-cli)
  Using cached https://files.pythonhosted.org/packages/9e/d4/10f46e5cfac773e22707237bfcd51bbffeaf0a576b0a847ec7ab15bd7ace/beautifulsoup4-4.6.0-py3-none-any.whl
Collecting MechanicalSoup<0.9,>=0.7.0 (from kaggle-cli)
  Using cached https://files.pythonhosted.org/packages/5c/2e/f63ed26b51e36efa4cc22cad18187fcb0a253f756d548c96bb931f13de98/MechanicalSoup-0.8.0-py2.py3-none-any.whl
Collecting lxml<4.1,>=4.0.0 (from kaggle-cli)
  Using cached https://files.pythonhosted.org/packages/a0/b5/4c6995f8f259f0858f79460e6d277888f8498ce1c1a466dfbb24f06ba83f/lxml-4.0.0-cp36-cp36m-manylinux1_x86_64.whl
Collecting cliff<2.9,>=2.8.0 (from kaggle-cli)
```

```
medialab@315-1080Ti:~$ pip install -U kaggle-cli
Requirement already up-to-date: kaggle-cli in ./miniconda3/lib/python3.6/site-packages (0.12.13)
Requirement not upgraded as not directly required: MechanicalSoup<0.9,>=0.7.0 in ./miniconda3/lib/python3.6/site-packages (from kaggle-cli) (0.8.0)
Requirement not upgraded as not directly required: cssselect<1.1,>=1.0.1 in ./miniconda3/lib/python3.6/site-packages (from kaggle-cli) (1.0.3)
```

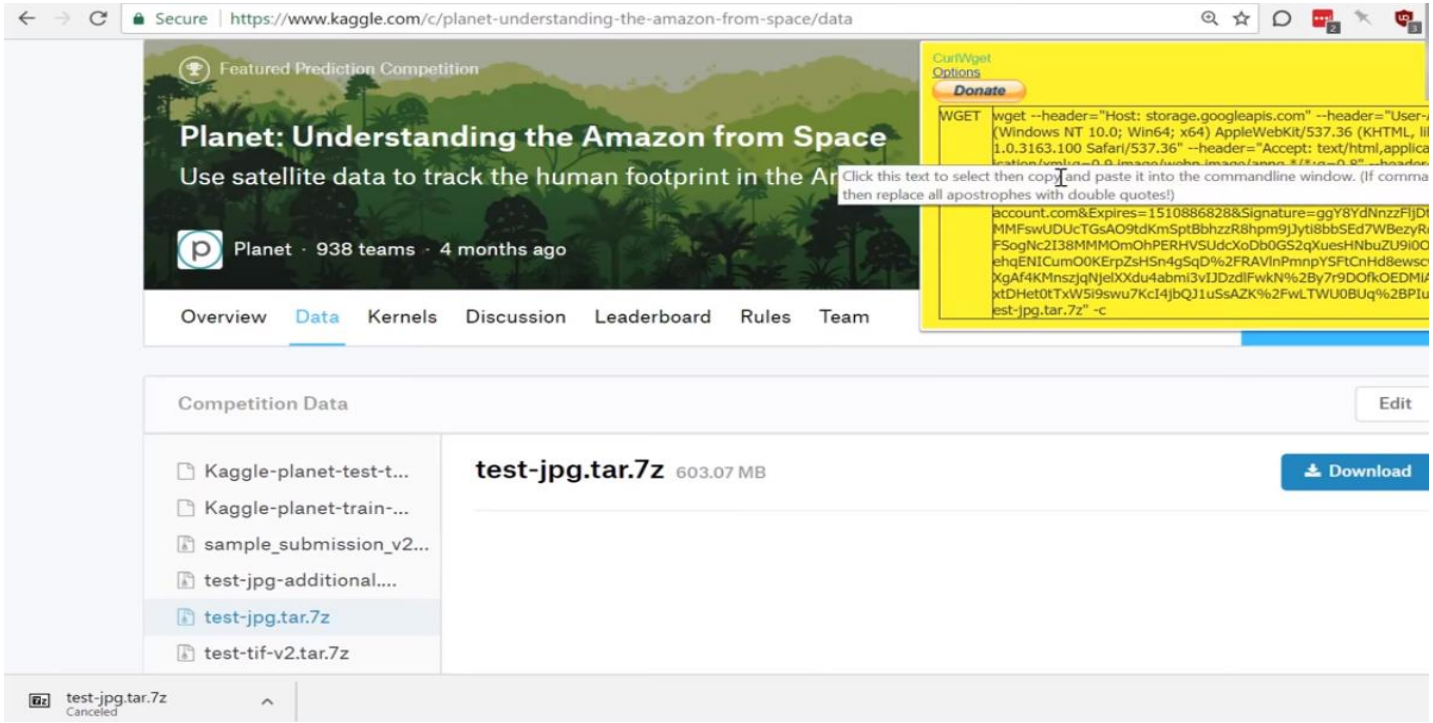
```
medialab@315-1080Ti:~$ kg download -u l1951931460qq.com -p swt57957319 -c planet-understanding-the-amazon-from-space -f test-jpg.tar.7z
downloading https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/download/test-jpg.tar.7z

test-jpg.tar.7z 100% |##############################| Time: 0:01:22   7.3 MiB/s
```

Kaggle数据集下载方法

2.使用Google Chrome的CurlWget扩展程序

在谷歌浏览器搜索CurlWget后添加到谷歌浏览器，搜索框出现一个黄色的标志表示添加成功，在谷歌浏览器找到需要下载的文件点击下载后取消下载。



使用CurlWget扩展程序

点击黄色按钮，把里面的命令复制到命令行，即可把数据下载到相应文件夹

CurlWget

WGET

```
wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8" --header="Accept-Language: zh-CN,zh;q=0.9" "https://storage.googleapis.com/kaggle-competitions-data/kaggle/6322/Kaggle-planet-train-tif.torrent?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1531733997&Signature=jOhdjcXh%2BF7h7tkXfspBIf5nPj7QVzYMHn7r2l6CHNfW9c8PfykFto%2FrsDbT4MBbFveQbk2frLnAOVtV%2Fnz%2BecmpT3Utsv6d%2FUwfgzJyuvYJf7CZuJD7Ig5MSjn5aQPnADxSCiqUWQiK4czr465VZ0gd8Qw0QJUPf0R5zHKS8aieA%2FqVnyA%2F3kGoHHjPoHCw8SKvhR6aA2bKCaZ2GLnitsJA5lDfq2CakhP5ro6892as2BxgSo5Vx1J1U8et47hjIjsw6zlrf%2BtK%2F%2FgqMbErTTiyDHwvwLCZXm08sNhVpKVmL521kDpyhr%2B%2BCgjPSuA4FCMJIhKi%2FodaVwE%2F4ym2%2Fw%3D%3D" -O "Kaggle-planet-train-tif.torrent" -c
```

Options

```
--2018-07-13 17:45:47--  https://storage.googleapis.com/kaggle-competitions-data/kaggle/6322/Kaggle-planet-train-tif.torrent?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1531733997&Signature=jOhdjcXh%2BF7h7tkXfspBIf5nPj7QVzYMHn7r2l6CHNfW9c8PfykFto%2FrsDbT4MBbFveQbk2frLnAOVtV%2Fnz%2BecmpT3Utsv6d%2FUwfgzJyuvYJf7CZuJD7Ig5MSjn5aQPnADxSCiqUWQiK4czr465VZ0gd8Qw0QJUPf0R5zHKS8aieA%2FqVnyA%2F3kGoHHjPoHCw8SKvhR6aA2bKCaZ2GLnitsJA5lDfq2CakhP5ro6892as2BxgSo5Vx1J1U8et47hjIjsw6zlrf%2BtK%2F%2FgqMbErTTiyDHwvwLCZXm08sNhVpKVmL521kDpyhr%2B%2BCgjPSuA4FCMJIhKi%2FodaVwE%2F4ym2%2Fw%3D%3D
正在解析主机 storage.googleapis.com (storage.googleapis.com)... 2404:6800:4008:803::2010, 172.217.24.48
正在连接 storage.googleapis.com (storage.googleapis.com)|2404:6800:4008:803::2010|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 1055292 (1.0M) [application/x-bittorrent]
正在保存至：“Kaggle-planet-train-tif.torrent”

Kaggle-planet-train 100%[===================>]   1.01M   903KB/s    in 1.1s

2018-07-13 17:45:49 (903 KB/s) - 已保存 “Kaggle-planet-train-tif.torrent” [1055292/1055292])
```

Keras与fastai的对比

```
# GPU 版本
>>> pip install --upgrade tensorflow-gpu

# CPU 版本
>>> pip install --upgrade tensorflow

# Keras 安装
>>> pip install keras -U --pre
```

Edit View Insert Cell Kernel Navigate Widgets Help Snippets Trusted

```
    shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(train_data_dir,
    target_size=(sz, sz),
    batch_size=batch_size, class_mode='binary')

validation_generator = test_datagen.flow_from_directory(validation_data_dir,
    shuffle=False,
    target_size=(sz, sz),
    batch_size=batch_size, class_mode='binary')
```

```
Found 23000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
```

In [6]:
```
base_model = ResNet50(weights='imagenet', include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)
```

In [7]:
```
model = Model(inputs=base_model.input, outputs=predictions)
for layer in base_model.layers: layer.trainable = False
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

In [8]:
```
%%time
model.fit_generator(train_generator, train_generator.n // batch_size, epochs=3, workers=4,
        validation_data=validation_generator, validation_steps=validation_generator.n // batch_size)
```

```
Epoch 1/3
```

Keras与fastai对比

```
In [8]: %%time
model.fit_generator(train_generator, train_generator.n // batch_size, epochs=3, workers=4,
        validation_data=validation_generator, validation_steps=validation_generator.n // batch_size)
```

```
Epoch 1/3
359/359 [==============================] - 144s 401ms/step - loss: 0.1960 - acc: 0.9499 - val_loss: 0.1714 - val_acc: 0.9632
Epoch 2/3
359/359 [==============================] - 140s 390ms/step - loss: 0.0859 - acc: 0.9694 - val_loss: 0.0585 - val_acc: 0.9869
Epoch 3/3
359/359 [==============================] - 134s 372ms/step - loss: 0.0640 - acc: 0.9770 - val_loss: 0.0615 - val_acc: 0.9839
CPU times: user 20min 31s, sys: 1min 20s, total: 21min 52s
Wall time: 6min 57s
```

Here's how to train and evalulate a *dogs vs cats* model in 3 lines of code, and under 20 seconds:

```
In [13]: # Uncomment the below if you need to reset your precomputed activations
!rm -rf {PATH}tmp
```

```
In [13]: arch=resnet34
data = ImageClassifierData.from_paths(PATH, tfms=tfms_from_model(arch, sz))
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 3)
```

Epoch 100% 3/3 [00:17<00:00, 5.90s/it]

```
[ 0.       0.04955  0.02605  0.98975]
[ 1.       0.03977  0.02916  0.99219]
[ 2.       0.03372  0.02929  0.98975]
```

Keras需要更多的代码和不同的参数，fastai在指定体系结构后就能自动设置必要的参数，Keras训练时间更长而且模型训练后的效果没有fastai好。

向Kaggle提交结果：以dogbreed为例

以lesson 1中的dogbreed任务为例
data.class按字母顺序给出所有类别的名称

```
In [25]: data.classes

Out[25]: ['affenpinscher',
          'afghan_hound',
          'african_hunting_dog',
          'airedale',
          'american_staffordshire_terrier',
          'appenzeller',
          'australian_terrier',
          'basenji',
          'basset',
          'beagle',
          'bedlington_terrier',
          'bernese_mountain_dog',
          'black-and-tan_coonhound',
          'blenheim_spaniel',
          'bloodhound',
          'bluetick',
          'border_collie',
          'border_terrier',
          'borzoi',
```

data.test_ds包含测试集数据，文件名在data.test_ds.fnames中

```
In [24]: data.test_ds.fnames

Out[24]: ['test/765e7af138f98a55beb08ca4fd304cc8.jpg',
          'test/0348bdc26f3ddab7bb839fe245f9062f.jpg',
          'test/90167f4ade65f5ba4308404cc9342f25.jpg',
          'test/30e54ef6f9e66673cd594138b7302a7f.jpg',
          'test/24193dad6b66cb25cca7e84d731d4290.jpg',
          'test/ccbb9dd96ed2b8a635b4389eb85c6c96.jpg',
          'test/3f0552a49c585a645fcb6145163d56cc.jpg',
          'test/1c0dda70b0facf724280c3529937c848.jpg',
          'test/4e05e0a32ce41086abcb5868d97fc293.jpg',
          'test/1326e98b42f5026ecd51389c4afde535.jpg',
          'test/4d3ff3175f121743c13c7406486be193.jpg',
```

向Kaggle提交结果：以dogbreed为例

```
log_preds,y = learn.TTA(is_test=True)
probs = np.exp(log_preds)
```

```
probs.shape
```

```
(10357, 120)
```

```
ds = pd.DataFrame(probs)
ds.columns = data.classes
```

```
ds.insert(0, 'id', [o[5:-4] for o in data.test_ds.fnames])
```

```
ds.head()
```

...

```
SUBM = f'{PATH}subm/'
os.makedirs(SUBM, exist_ok=True)
ds.to_csv(f'{SUBM}subm.gz', compression='gzip', index=False)
```

```
FileLink(f'{SUBM}subm.gz')
```

data/dogbreed/subm/subm.gz

预测样本时使用TTA对测试图片进行数据增强操作

模型的预测结果默认转化为对数，因此用np.exp()将其转化为普通概率

Probs.shape看出在120种分类中有10357张图片

pd.DataFrame创建日期帧，并将种类名称分配给列

Insert在0列插入新的一列，命名为id,剪切掉了文件名开始的test/和结尾的.jpg

使用.to_csv()将数据帧写入csv文件，使用compression='gzip'进行压缩

然后notebook就出现了一个csv文件

使用FileLink可以将文件从服务器下载到计算机

对一个样本进行预测

```
fn=data.val_ds.fnames[0]
```

```
fn
```

```
'train/000bec180eb18c7604dcecc8fe0dba07.jpg'
```

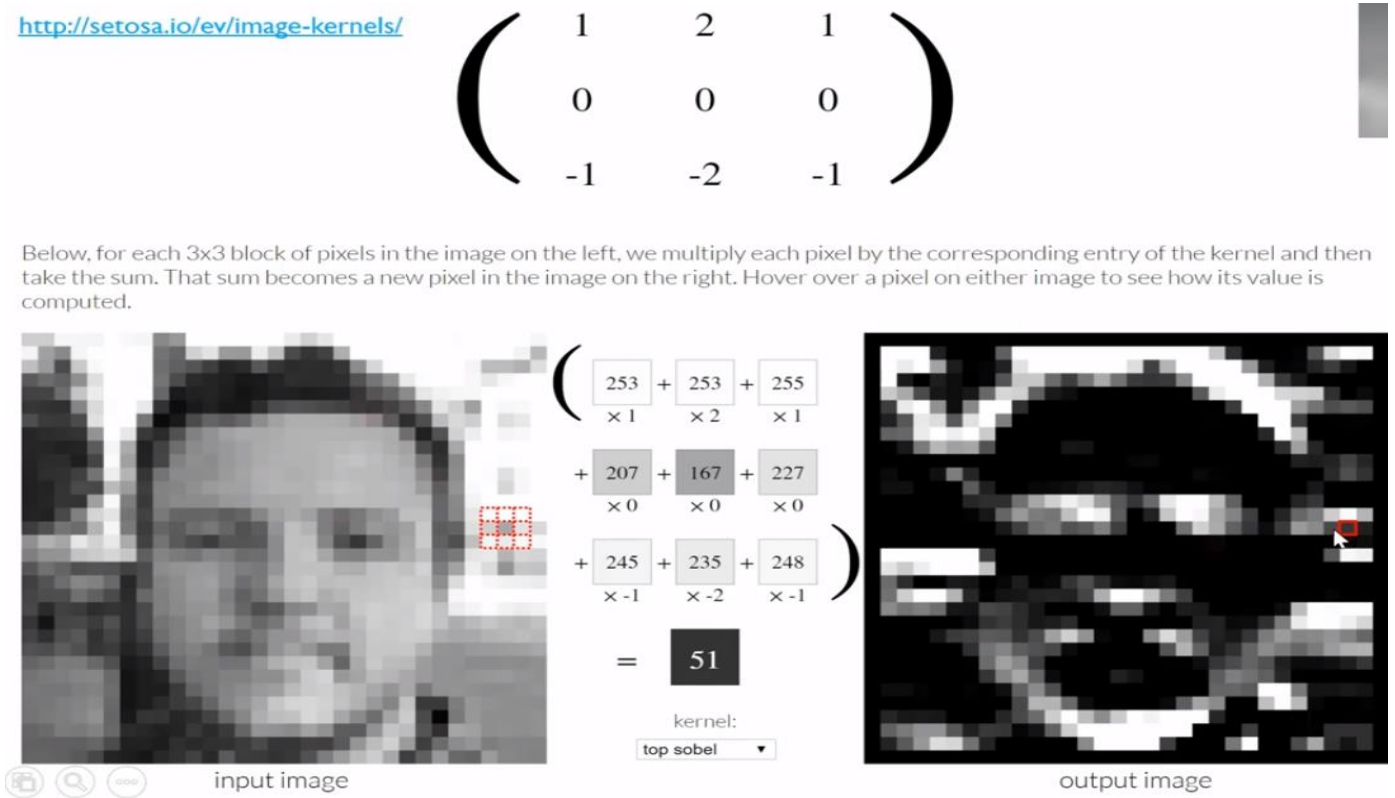
```
Image.open(PATH+fn)
```

```
trn_tfms, val_tfms=tfms_from_model(arch, sz)
```

```
im = val_tfms(Image.open(PATH+fn)
preds=learn.predict_array(im[None])
np.argmax(preds)
```

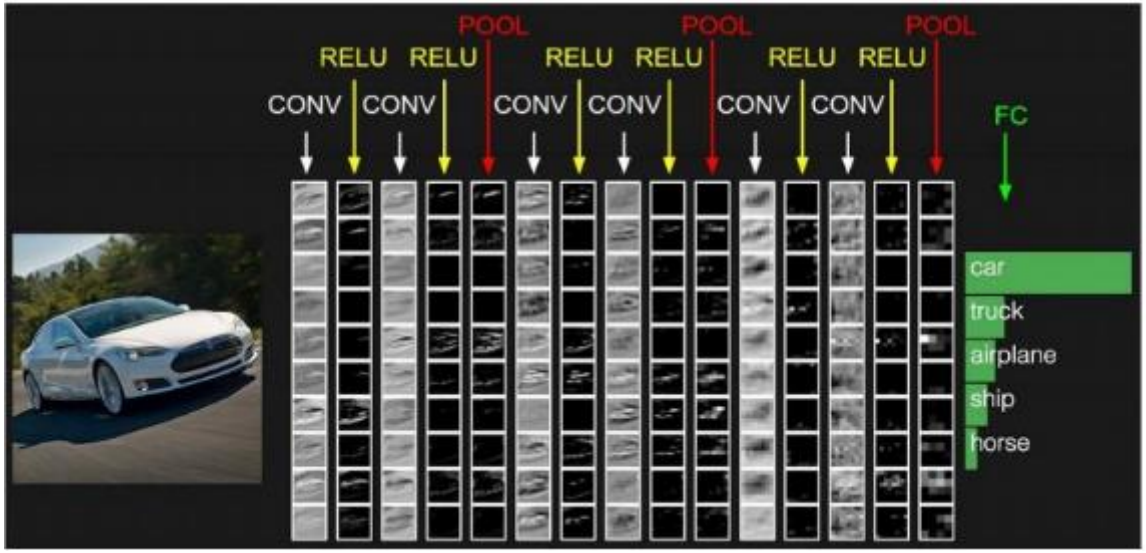
首先在验证集中打开一张图片
使用Image.open显示图片
使用predict_arry()进行计算预测
tfms_from_model函数是将训练和验证样本进行数据转换
使用im[None]将三维张量变为四维张量

CNN详解



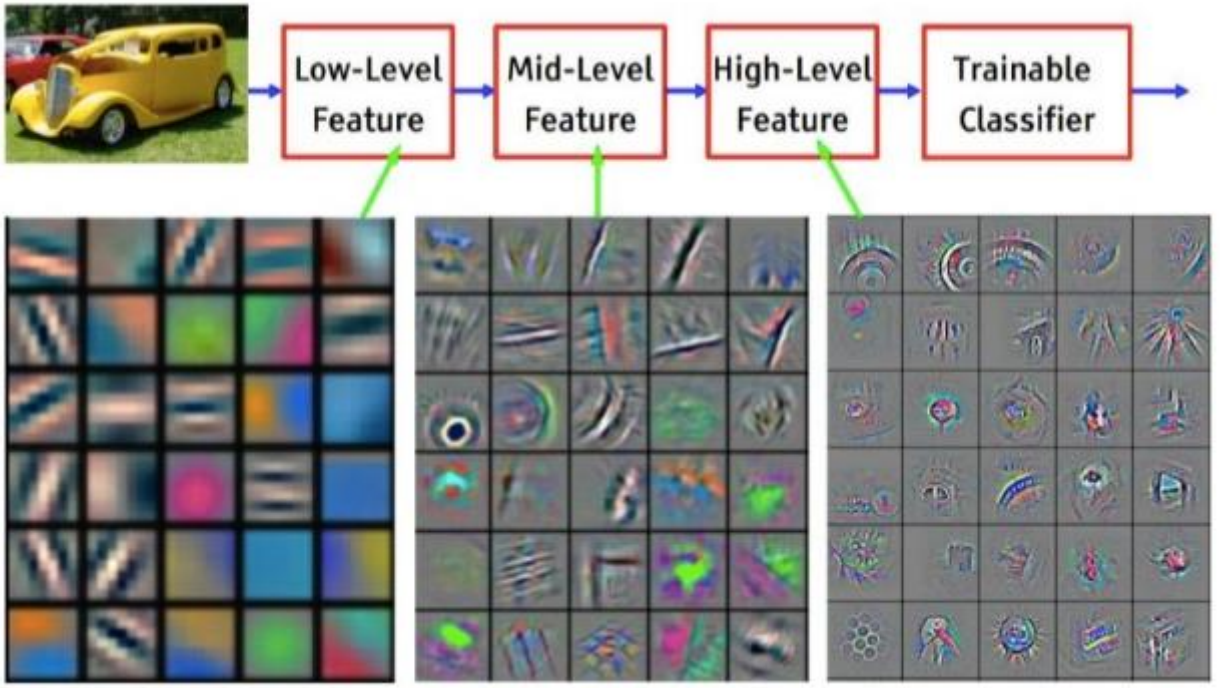
在lesson1的演示中看到了卷积的过程，是用一个3x3的矩阵依次点乘移动框内的数后相加的和作为输出

CNN详解



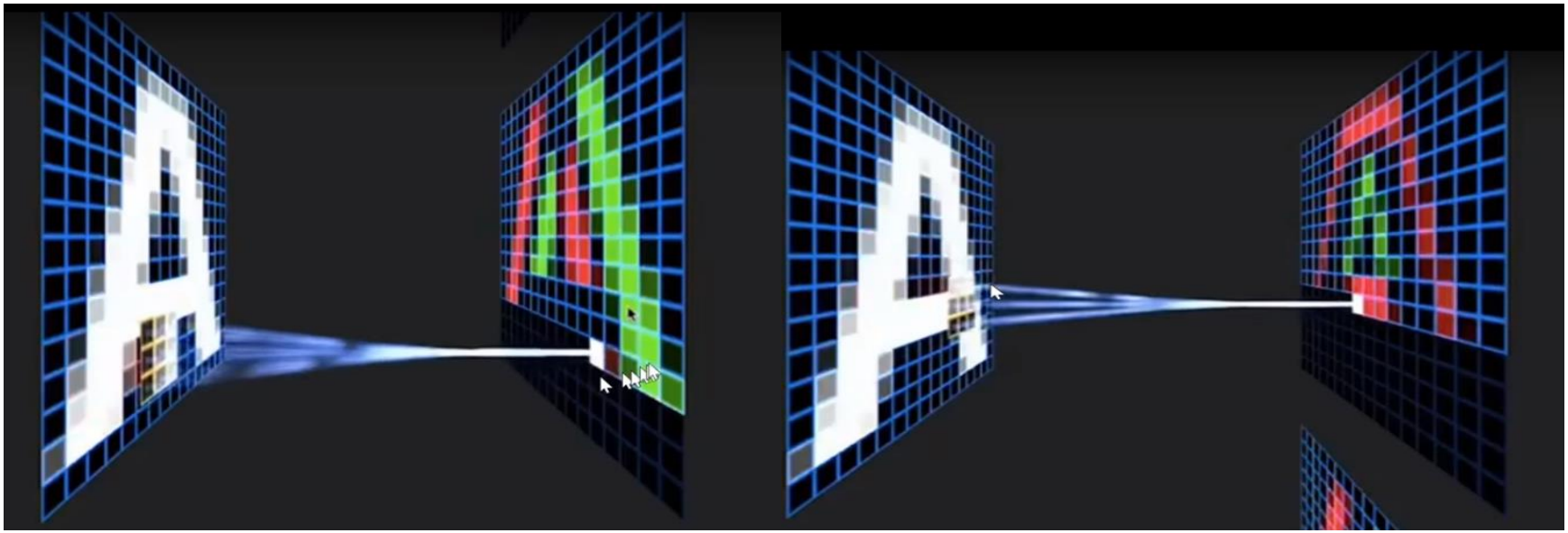
CNN的框架：包含多个卷积层、激活层、池化层以及全连接层

CNN详解



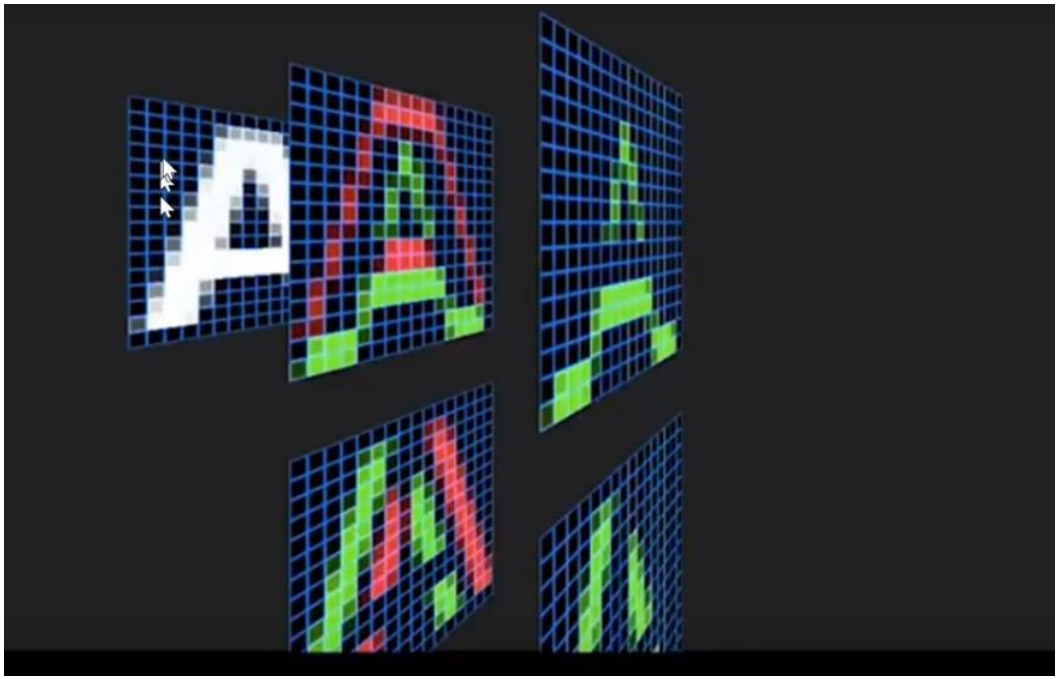
卷积的过程实现的是提取特征

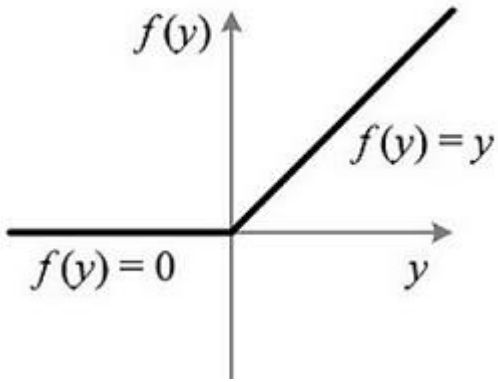
CNN详解



左侧为黑色，右侧为白色，每个3x3区域元素相乘后相加，黑色变为白色的区域计算得到的值为正值，显示为绿色；白色变为黑色的区域计算得到的值为负值，显示为红色，第二个卷积核上白下黑，进行卷积计算后得到新的红绿像素

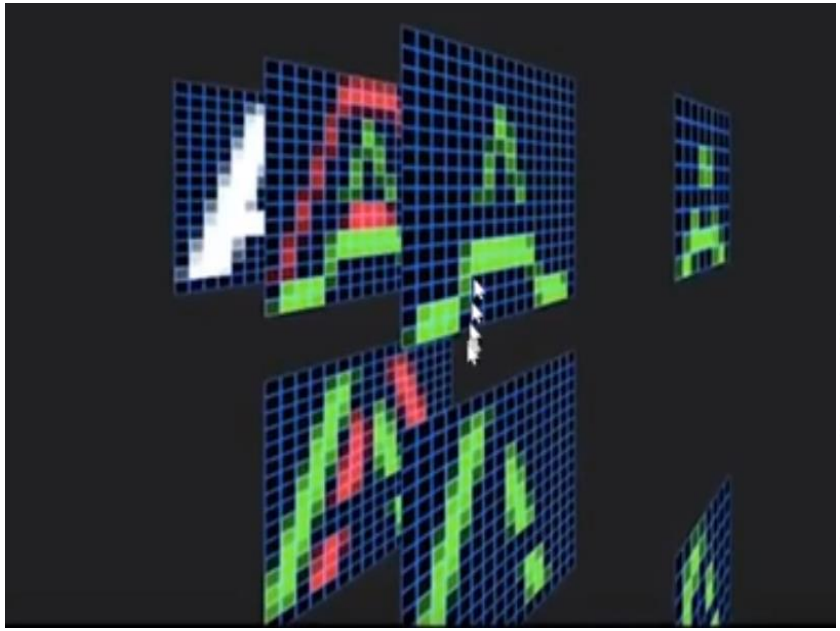
CNN详解





激活层：卷积得到第二层，并用ReLU激活函数去除负值

CNN详解



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

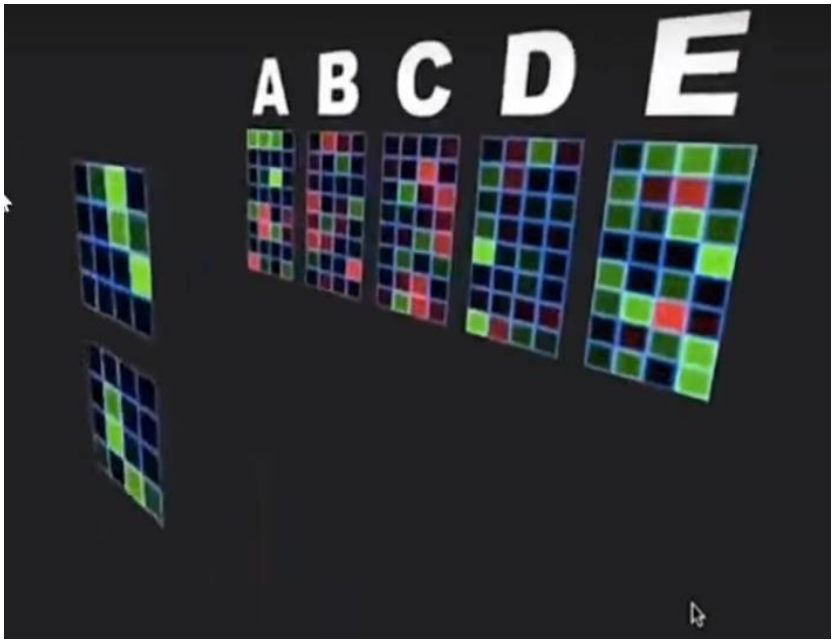
max pool with 2x2 filters
and stride 2

→

6	8
3	4

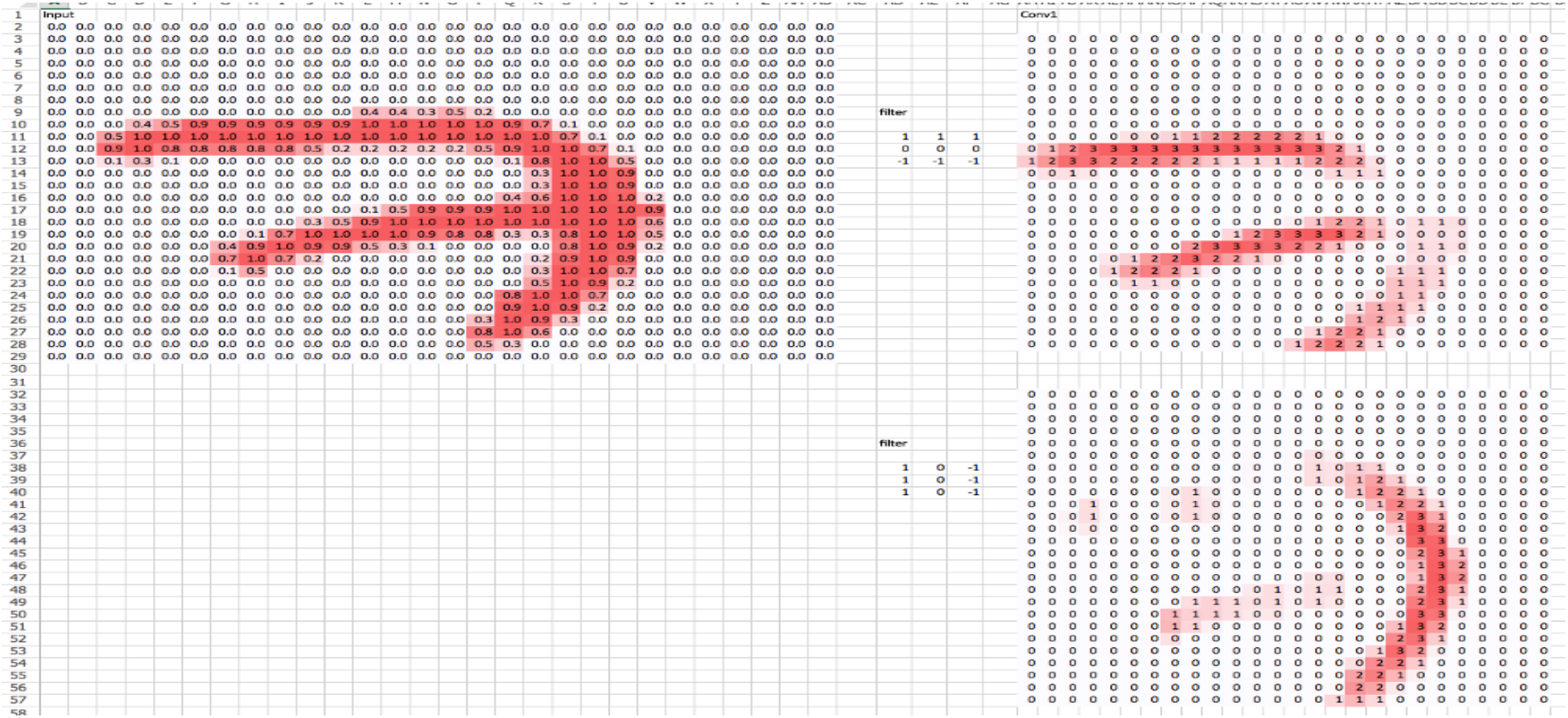
池化层（以最大池化为例）：对于每个2*2的窗口选出最大的数作为对应位置的输出值

CNN详解



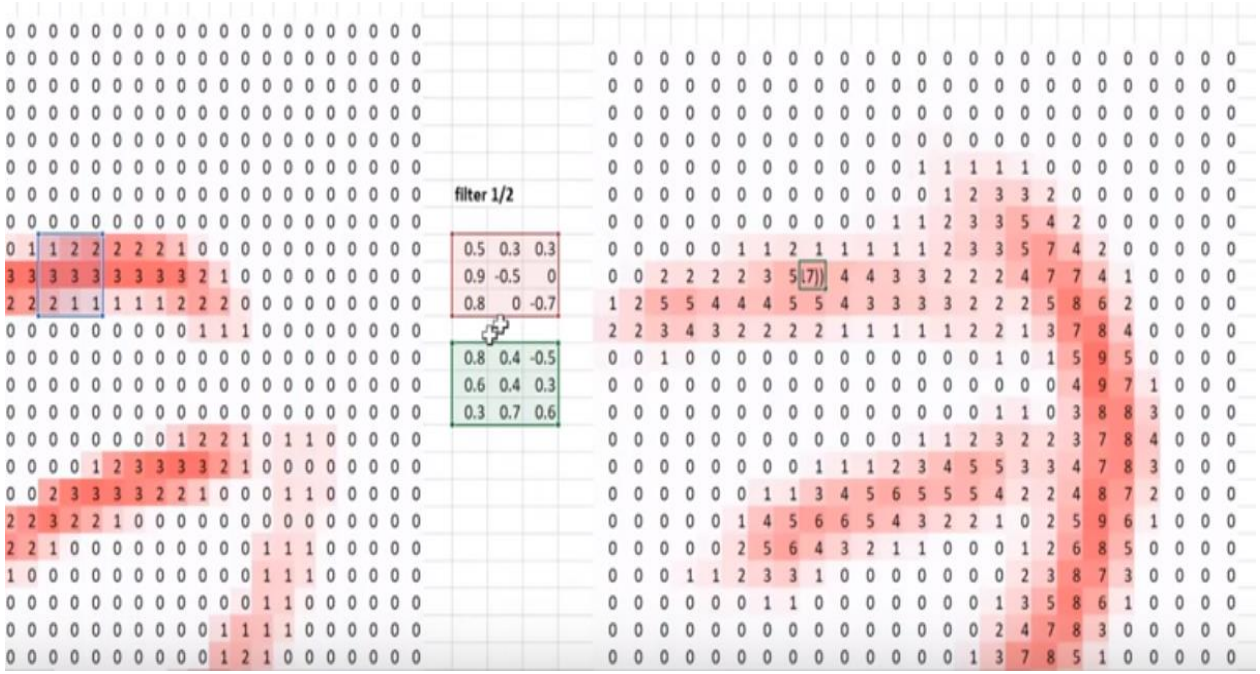
将得到的图像的每个元素以某种对应模式处理后与待分种类进行比较，相似度最大的作为预测分类

CNN详解



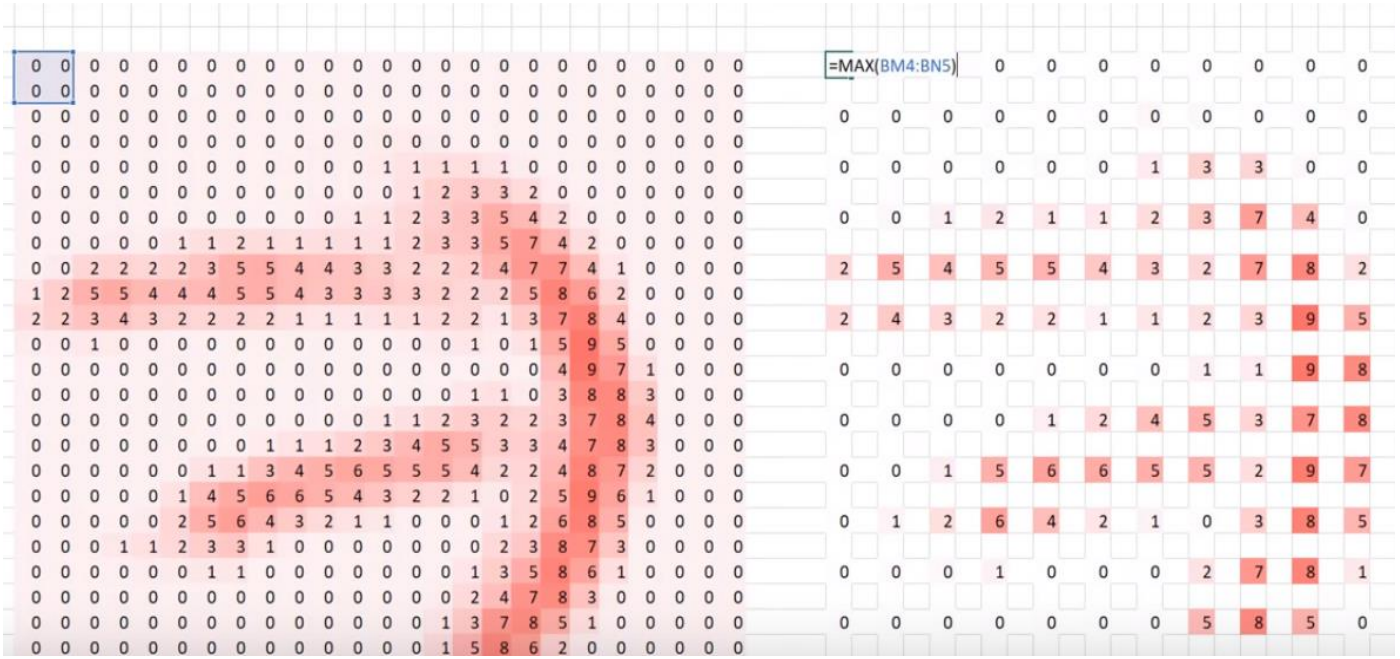
上面的卷积核检测出了原图中的水平边缘，而下面的卷积核检测出了原图中的垂直边缘。

CNN详解



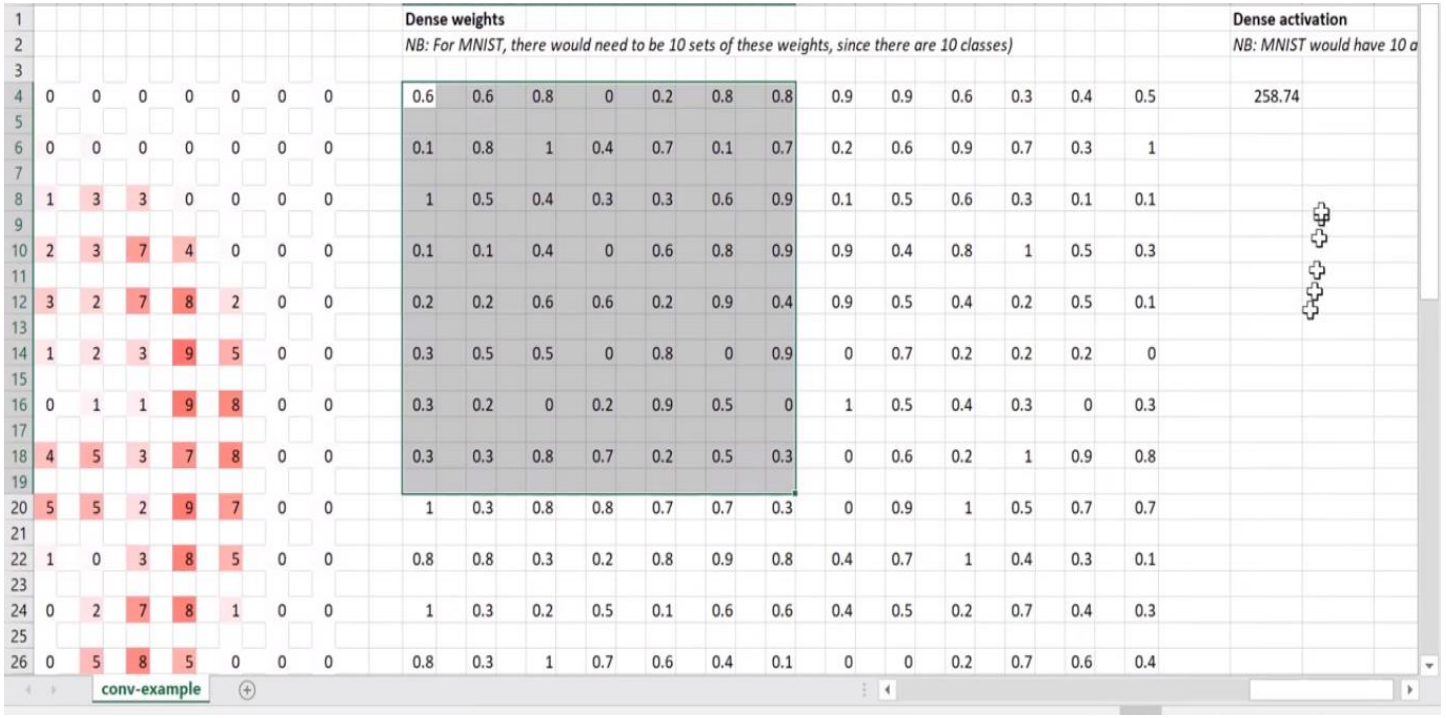
第一层的输出是两个矩阵，在深度学习的计算框架中我们会将这两个矩阵拼接起来，存为一个三维的张量，在fastai（pytorch）中，这里的张量其shape为(2,w,h)，其中w和h分别为输出的宽和高，因此进行第二层卷积时，这一层的卷积核是一个shape为(2,3,3)的张量

CNN详解



Maxpooling：用于压缩数据和参数的量，减小过拟合。

CNN详解



								Dense weights													Dense activation
								NB: For MNIST, there would need to be 10 sets of these weights, since there are 10 classes)													*NB: MNIST would have 10 a*
4	0	0	0	0	0	0	0	0.6	0.6	0.8	0	0.2	0.8	0.8	0.9	0.9	0.6	0.3	0.4	0.5	258.74
6	0	0	0	0	0	0	0	0.1	0.8	1	0.4	0.7	0.1	0.7	0.2	0.6	0.9	0.7	0.3	1	
8	1	3	3	0	0	0	0	1	0.5	0.4	0.3	0.3	0.6	0.9	0.1	0.5	0.6	0.3	0.1	0.1	
10	2	3	7	4	0	0	0	0.1	0.1	0.4	0	0.6	0.8	0.9	0.9	0.4	0.8	1	0.5	0.3	
12	3	2	7	8	2	0	0	0.2	0.2	0.6	0.6	0.2	0.9	0.4	0.9	0.5	0.4	0.2	0.5	0.1	
14	1	2	3	9	5	0	0	0.3	0.5	0.5	0	0.8	0	0.9	0	0.7	0.2	0.2	0.2	0	
16	0	1	1	9	8	0	0	0.3	0.2	0	0.2	0.9	0.5	0	1	0.5	0.4	0.3	0	0.3	
18	4	5	3	7	8	0	0	0.3	0.3	0.8	0.7	0.2	0.5	0.3	0	0.6	0.2	1	0.9	0.8	
20	5	5	2	9	7	0	0	1	0.3	0.8	0.8	0.7	0.7	0.3	0	0.9	1	0.5	0.7	0.7	
22	1	0	3	8	5	0	0	0.8	0.8	0.3	0.2	0.8	0.9	0.8	0.4	0.7	1	0.4	0.3	0.1	
24	0	2	7	8	1	0	0	1	0.3	0.2	0.5	0.1	0.6	0.6	0.4	0.5	0.2	0.7	0.4	0.3	
26	0	5	8	5	0	0	0	0.8	0.3	1	0.7	0.6	0.4	0.1	0	0	0.2	0.7	0.6	0.4	

conv-example

全连接层：对于分类任务，如果需要分类的类别数为m，那么全连接层的输出向量维数便为m

CNN详解

	A	B	C	D	E	F	G	H	I	J
1		output	exp	softmax						
2	cat	-2.39	0.09	0.00						
3	dog	2.06	19.23	0.92						
4	plane	0.49	1.63	0.08						
5	fish	-3.90	0.02	0.00						
6	building	-4.08	0.02	0.00						
7			20.99	1.00						
8										
9										
10										
11										
12										
13										
14										
15										

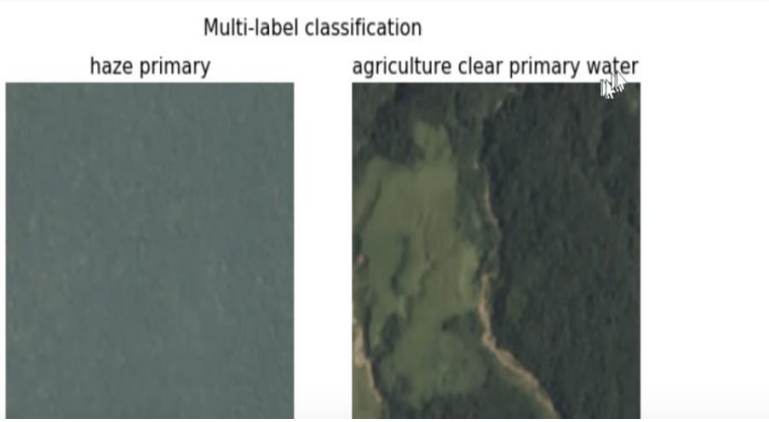
Softmax：在得到一列数后，先进行取指数全部变为正数，然后把所有的数加起来作为基数，把每个类的指数值除以基数，选取最大的一个数对应的类作为预测

CNN详解

	A	B	C	D	E
1		output	exp	softmax	actuals
2	cat	1.60	4.94	0.48	0
3	dog	1.10	3.01	0.30	1
4	plane	-0.54	0.58	0.06	0
5	fish	-3.71	0.02	0.00	0
6	building	0.50	1.65	0.16	0
7			10.20	1.00	

对于single-label classifier来说，经过softmax操作后得到了一个概率向量，该向量中的每个值均在0到1之间，并且它们的和为1，这样便可以和经过one-hot编码的label计算交叉熵损失。使用softmax的好处是其可以相对突出某一类的概率，从而选取概率最大的数所对应的类别作为预测类别。

多标签分类



一张图片属于两个或多个类

这里以planet数据集为例简单介绍一下多标签分类。

由于这里的数据为卫星图片，因此数据增强算法使用transforms_top_down，允许对图片进行任意角度的旋转操作。

```
# Set up our get_data function
# This sets up our transforms and returns an image classifier which we can use to browse data, etc.
# Notice this time we can set top-down transforms since we're using satellite data
def get_data(image_size):
    transforms = tfms_from_model(f_model, image_size, aug_tfms=transforms_top_down, max_zoom=1.05)
    return ImageClassifierData.from_csv(PATH, 'train-jpg', label_csv, tfms=transforms,
                                        suffix='.jpg', val_idxs=val_idxs, test_name='test-jpg')
```

多标签分类

x,y = next(iter(data.val_dl))

dl是数据加载器，提供一个minibatch，iter()函数从数据加载器中创建迭代器，next()函数产生新的minibatch，x是该批中的图像，y是对应的标签。
在notebook中可以用shift+tab查看变量的默认值

```
y
```

```
    1     0     0  ...     0     1     1
    0     0     0  ...     0     0     0
    0     0     0  ...     0     0     0
  ...           ⋱           ...
    0     0     0  ...     0     0     0
    0     0     0  ...     0     0     0
    1     0     0  ...     0     0     0
[torch.FloatTensor of size 64x17]
```

代码中batch_size的默认值是64，得到的标签数组大小是64x17,所以有17个不同的类。

多标签分类

```
list(zip(data.classes, y[0]))
```

```
[('agriculture', 1.0),
 ('artisinal_mine', 0.0),
 ('bare_ground', 0.0),
 ('blooming', 0.0),
 ('blow_down', 0.0),
 ('clear', 1.0),
 ('cloudy', 0.0),
 ('conventional_mine', 0.0),
 ('cultivation', 0.0),
 ('habitation', 0.0),
 ('haze', 0.0),
 ('partly_cloudy', 0.0),
 ('primary', 1.0),
 ('road', 0.0),
 ('selective_logging', 0.0),
 ('slash_burn', 1.0),
 ('water', 1.0)]
```

zip()命令将第一个列表和第二个列表的对应元素组合在一起，可以看出该图像对应的标签为agriculture，clear，primary，water

plt.imshow(data.val_ds.denorm(to_np(x))[0]*1.4)
实现的是将图片亮度调整为原来的1.4倍

多标签分类

```
# Start off training on small images
image_size=64
```

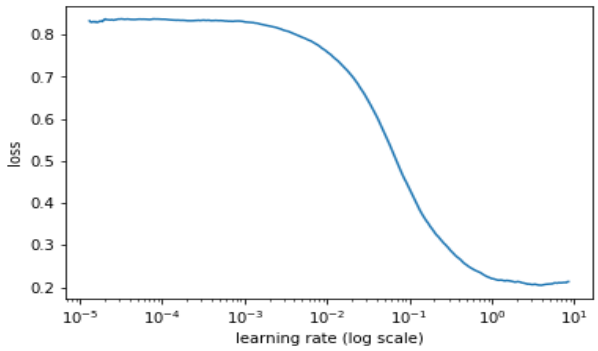
```
data = get_data(image_size)
```

```
# This goes through and resizes all of our images for efficiencies sake?
data = data.resize(int(image_size*1.3), 'tmp')
```

```
# First set up our model
learn = ConvLearner.pretrained(f_model, data, metrics=metrics)
```

```
learning_rate_finder = learn.lr_find()
learn.sched.plot()
```

epoch	trn_loss	val_loss	f2
0	0.220537	0.422463	0.789931



- 第一次训练时将图片大小调整为较小的64x64，在dogs vs. cats的任务中我们没有从尺寸如此小的图片开始训练，这是因为我们读取了在ImageNet预训练好的模型权重，ImageNet中的数据与猫狗数据的分布很相似，因此我们不需要对这些权重做太多更新，ImageNet训练时的图片尺寸大多为224×224或299×299，若使用64x64的尺寸去训练就不能很好地利用这些权重了。而卫星图片与ImageNet中的图片分布完全不同，我们不需要去考虑去这些权重的“保护”。
- 最后需要注意，在多标签分类任务中不应在最后采用softmax激活函数，因为如前述，softmax倾向于突出概率向量中的某一个值，而对于一个具有多个标签的数据来说，在其对应的多个标签维度上都应有较高的输出概率，因此可以采用sigmoid函数进行激活。

谢 谢！