遥感图像分类

1.1 LeNet-5

@tm9161

L1

LeNet-5

1. 基于MNIST的遥感数据集
2. LeNet-5搭建、训练和保存
3. LeNet-5调用和预测

Data | Tasks (1) | Code (7) | Discussion | Activity | Metadata

Usability 4.1 | Tags computer science

Data Explorer

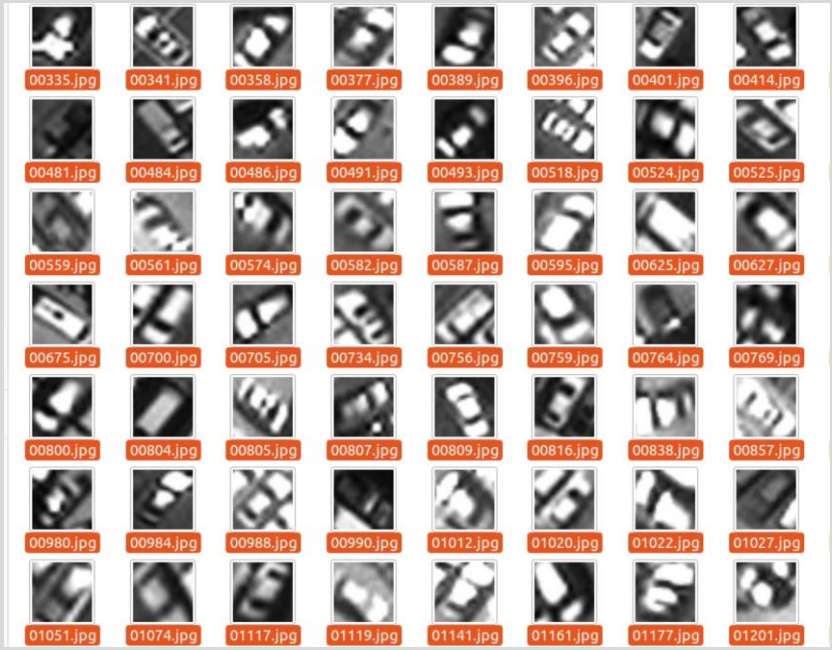
40.57 MB

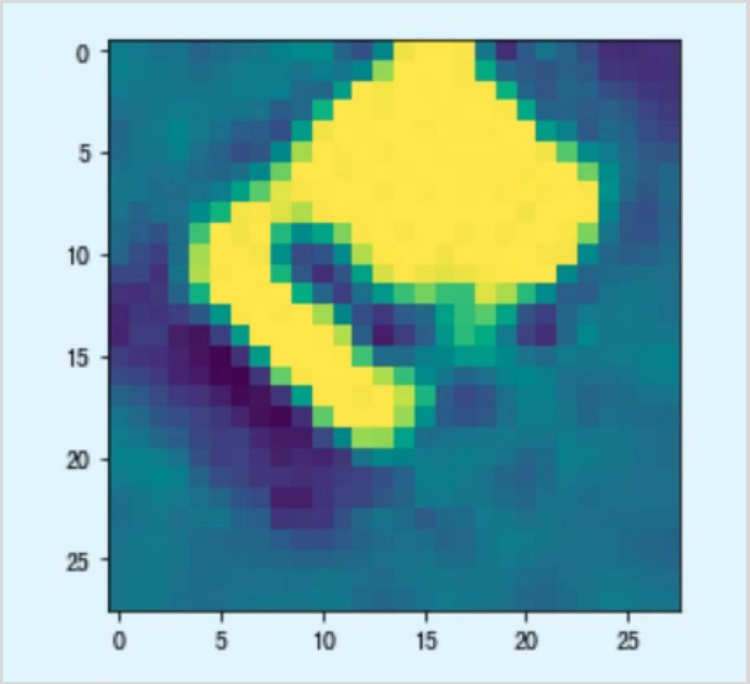
- overhead
  - testing
    - car
      - 08531.jpg
      - 08533.jpg
      - 08540.jpg
      - 08560.jpg
      - 08570.jpg
      - 08577.jpg
      - 08582.jpg

08531.jpg (586 B)

https://www.kaggle.com/datamunge/overheadmnist/version/1

基于MNIST的遥感图像数据集





包含76690张遥感图像（共10类）：68161张用于训练；8529张用于测试。

28x28像素的灰度图（有像素缺少）。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14
2	3	112	123	123	133	159	167	170	137	124	105	109	85	97	90
3	6	155	159	161	160	160	160	161	164	173	194	178	175	172	164
4	9	121	94	91	57	98	126	61	78	93	169	121	104	110	88
5	8	169	129	141	164	230	255	245	248	255	255	255	255	255	255
6	7	72	66	58	61	64	57	71	60	50	67	66	61	69	79
7	8	183	174	173	186	174	174	171	170	172	170	167	165	168	171
8	9	124	147	173	175	154	155	147	154	182	181	111	107	110	145
9	7	93	90	35	250	255	254	89	108	133	149	118	216	213	253
10	0	106	101	98	106	100	99	69	34	23	24	52	141	231	233
11	6	126	136	157	155	137	130	161	175	181	167	163	164	158	191
12	5	188	172	182	164	176	167	140	173	191	196	159	161	204	204
13	2	255	155	121	255	252	251	255	255	255	251	255	255	218	255
14	3	132	118	110	112	113	108	108	113	91	130	146	122	132	102
15	2	176	186	182	180	220	219	238	238	241	203	159	159	164	141
16	7	72	85	76	73	97	68	87	98	81	71	88	66	64	88
17	6	150	152	143	156	174	144	130	133	194	210	212	207	203	205
18	8	135	130	137	163	155	134	133	135	180	248	239	239	255	255
19	2	51	43	50	48	53	66	66	76	82	87	94	99	101	113
20	8	71	72	75	72	72	72	71	77	75	74	72	70	69	68
21	2	37	3	232	252	255	251	255	250	136	134	168	155	158	169
22	0	57	57	66	65	70	62	72	54	58	79	96	119	128	129
23	7	48	43	40	18	45	43	45	48	35	38	37	44	47	43

载入数据

1.读取CSV文件

2.转换成数组

3.读取图片

4.读取标签

```
train =pd.read_csv('sat/train.csv')
test = pd.read_csv('sat/test.csv')

#从csv文件中载入数据
```

```
train = np.array(train)
test = np.array(test)
```

```
train_images=train[:,1:]
test_images=test[:,1:]

#提取图片信息  28*28
```

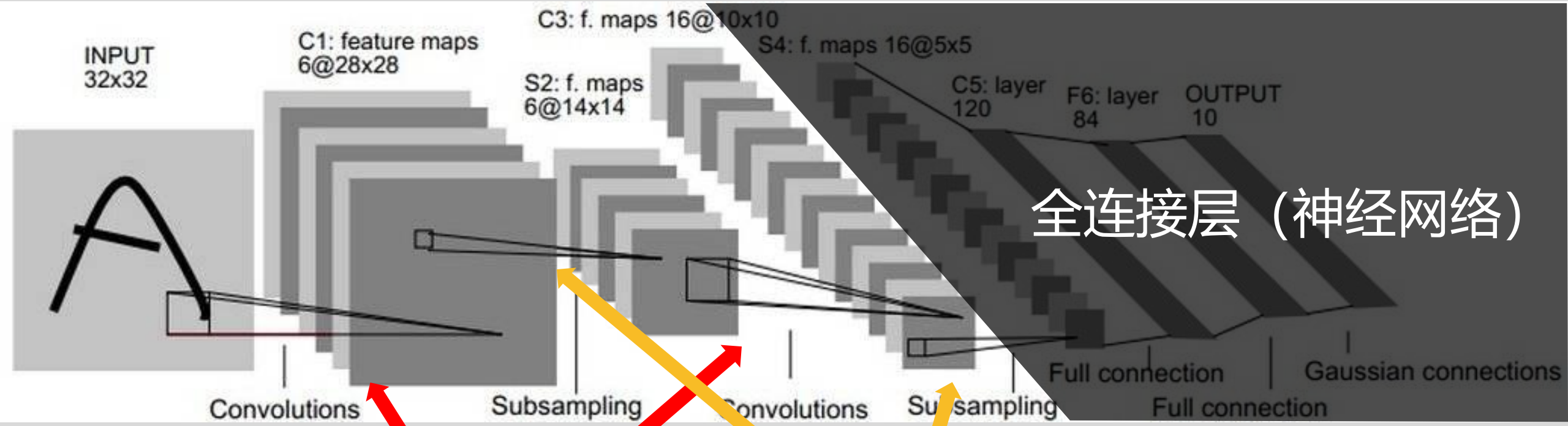
```
train_labels=train[:,:1]
test_labels=test[:,:1]

#提取标签信息  1
```

载入数据

5.维度改变

```
In [7]: train_images.shape

Out[7]: (68161, 784)

In [8]: train_labels.shape

Out[8]: (68161, 1)

In [6]: train_labels = train_labels.reshape(68161)
        test_labels = test_labels.reshape(8529)

        train_images = train_images.reshape(68161,28,28)
        test_images = test_images.reshape(8529,28,28)

        #初始化维度

In [10]: train_images.shape

Out[10]: (68161, 784)

In [11]: train_labels.shape

Out[11]: (68161, 1)
```

网络结构



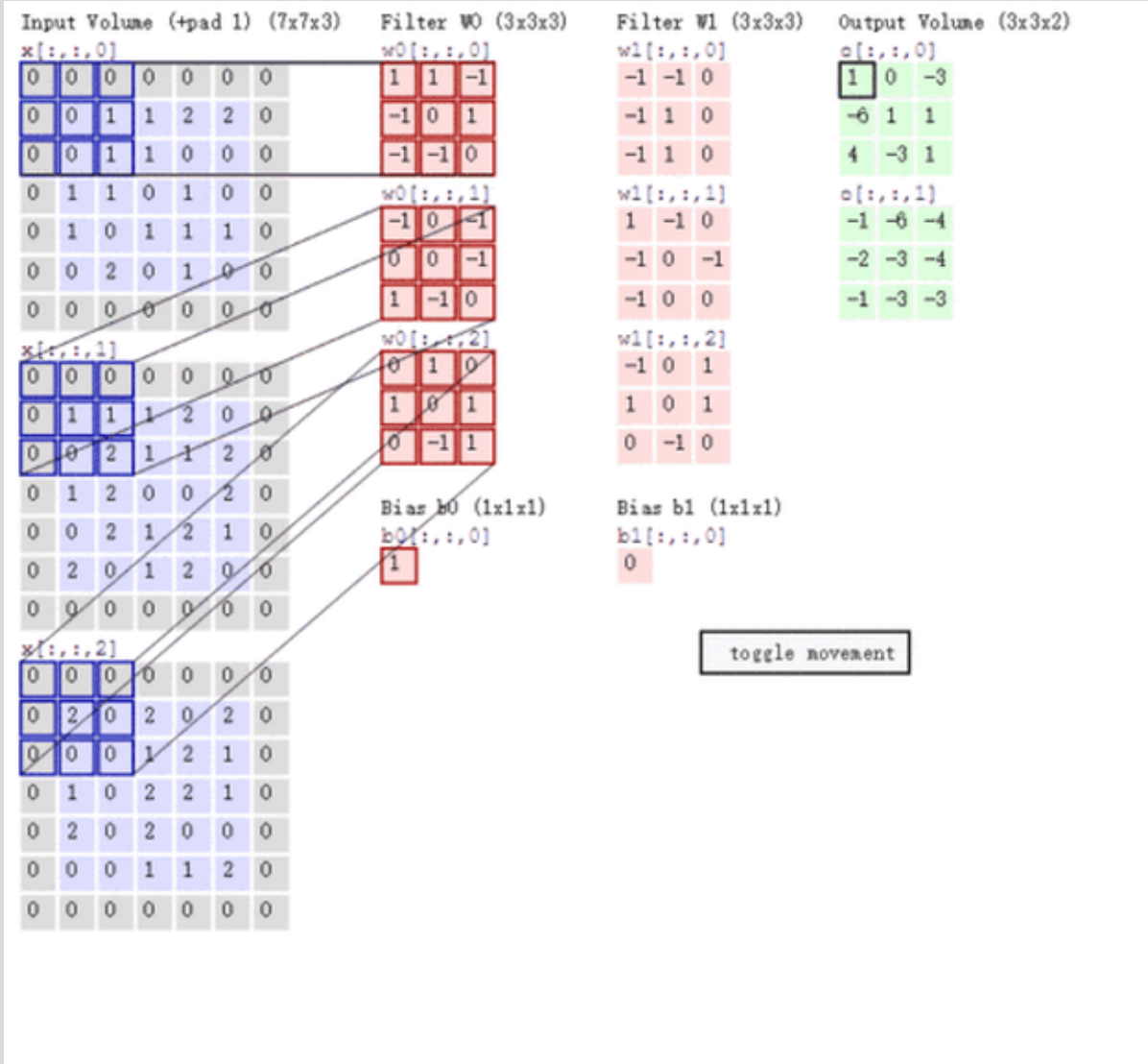
卷积层　　下采样层（池化层）

训练参数

卷积层：卷积核中的参数+偏置项；
（3层*3x3大小 +1*偏置）*2个

池化层：无参数需要训练。

*卷积核维度&卷积核个数区分



步长 Stride & 加边 Padding

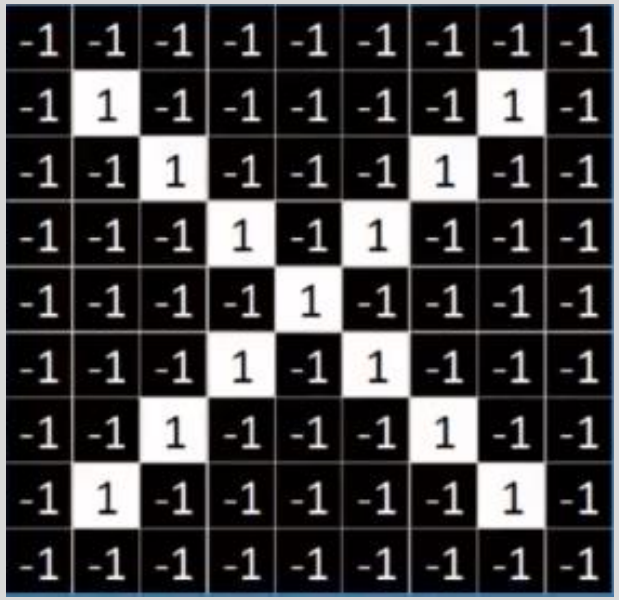
卷积后尺寸=
（输入-卷积核+加边像素数）/步长 +1
(6-3+0)/1 +1 =4

Tensorflow 默认：
Padding= ‘valid ‘（丢弃），strides=1

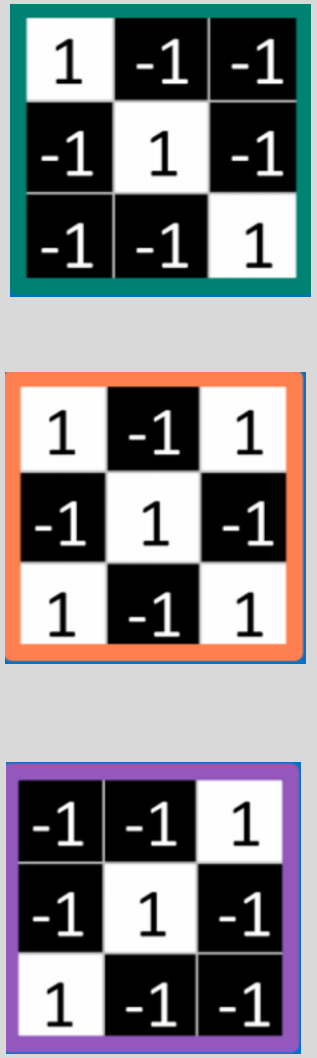
*长宽的改变

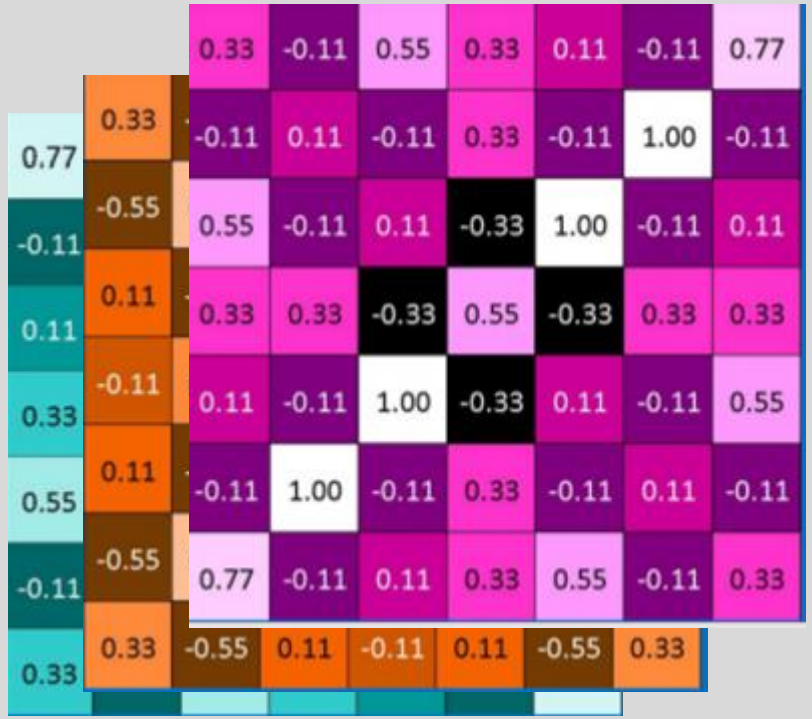
https://ezyang.github.io/convolution-visualizer/index.html

维度改变 reshape

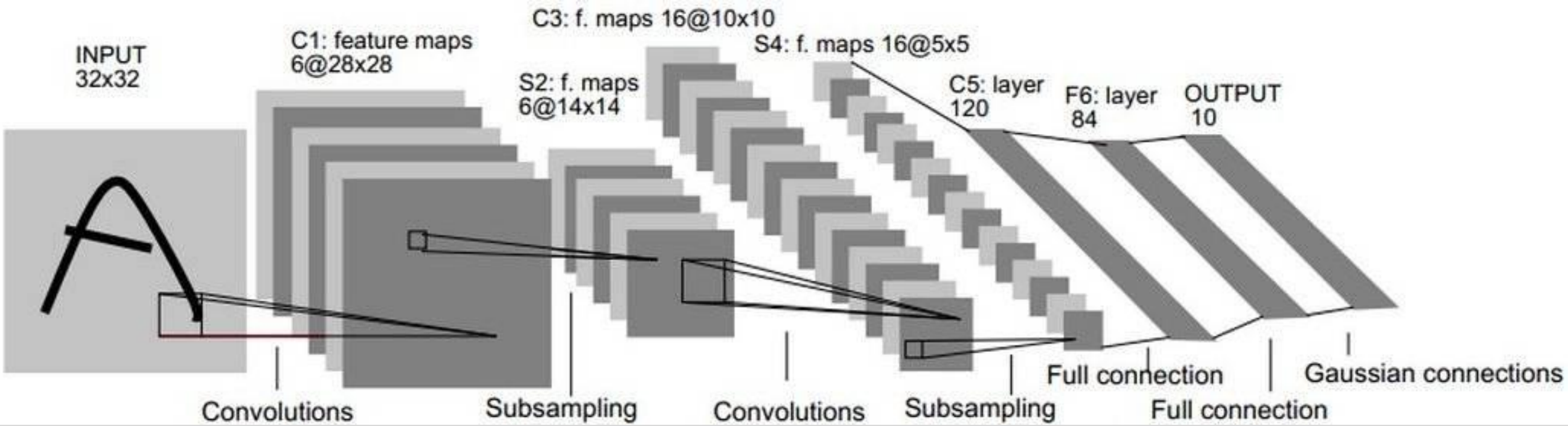


9*9*1





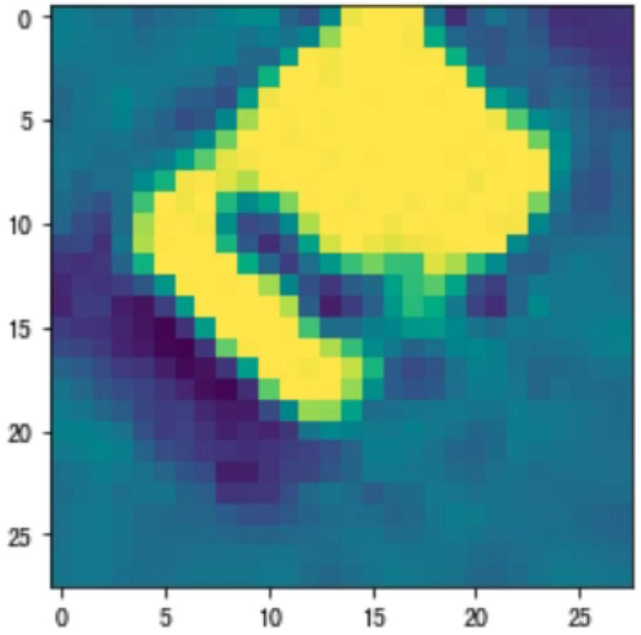
7*7*3



	输入	卷积、池化、神经元	输出	训练参数
输入层*			32*32	0
卷积层1	32*32	6个 5*5卷积核 步长为1	6*28*28（32-5+0）/1+1	1*（5*5）*6 +6=156
池化层1	6*28*28	2*2 步长为2	6*14*14	0
卷积层2	6*14*14	16个 5*5卷积核 步长为1	16*10*10 （14-5+0）/1+1	6*（5*5）*16 +16=2416
池化层2	16*10*10	2*2 步长为2	16*5*5	0
全连接层1	16*5*5	120个 5*5卷积核 步长为1	120*1*1（5-5+0）/1+1	16*（5*5）*120+120=48120
全连接层2	120		84	120*84+84=10164
输出层	84		10	84*10+10=850

图片读取&预处理

```
In [3]: img = cv2.imread('68187.jpg',0)
        #读取图片

In [4]: plt.imshow(img)

Out[4]: <matplotlib.image.AxesImage at 0x7ff6143844f0>
```



1.图片读取：cv2.imread

2.图片大小调整：cv2.resize

3.图片维度调整：reshape

```
train_images.shape
```

```
(60000, 28, 28)
```

```
train_images = train_images.reshape(60000, 28, 28, 1)
test_images = test_images.reshape(10000, 28, 28, 1)
```

```
train_images.shape
```

```
(60000, 28, 28, 1)
```

4.归一化：/255

模型预测

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

In [9]: predict

Out[9]: array([[9.8759693e-01, 2.6929181e-06, 2.4914041e-08, 2.4818671e-11,
                2.6326370e-06, 6.4962874e-10, 9.4881425e-06, 7.1182288e-04,
                2.6242146e-06, 1.1673761e-02]], dtype=float32)

In [10]: label = ['car','harbor','helicopter','oil_gas_field','parking_lot','plane','

In [12]: label[np.argmax(predict)]

Out[12]: 'car'
```

1

数据采集

In [1]:
```
import tensorflow as tf

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
```

In [2]:
```
train =pd.read_csv('sat/train.csv')
test = pd.read_csv('sat/test.csv')

#从csv文件中载入数据
```

In [3]:
```
train = np.array(train)
test = np.array(test)
```

In [4]:
```
train_images=train[:,1:]
test_images=test[:,1:]

#提取图片信息 28*28
```

In [5]:
```
train_labels=train[:,:1]
test_labels=test[:,:1]

#提取标签信息 1
```

In [7]:
```
train_images.shape
```

Out[7]: (68161, 784)

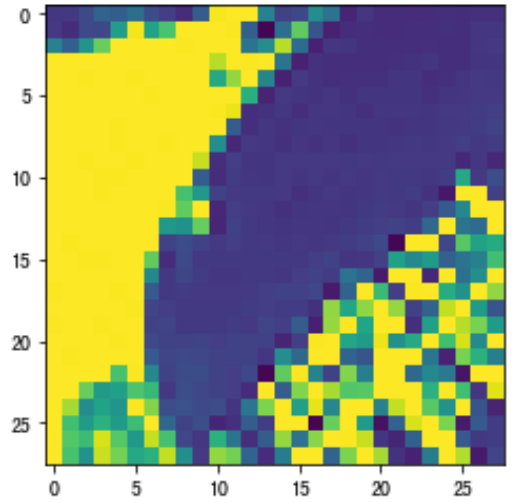
In [8]:
```
train_labels.shape
```

Out[8]: (68161, 1)

In [13]:
```
train_labels = train_labels.reshape(68161)
test_labels = test_labels.reshape(8529)

train_images = train_images.reshape(68161,28,28)
test_images = test_images.reshape(8529,28,28)

#初始化维度
```

```
plt.imshow(train_images[100])
```

```
<matplotlib.image.AxesImage at 0x7f85ba6494f0>
```




```
train_images = train_images.reshape(68161,28,28,1)
test_images = test_images.reshape(8529,28,28,1)
#增加维度，用于卷积操作
```

```
train_images = train_images / 255
test_images = test_images/ 255
```

```
train_labels = np.array(pd.get_dummies(train_labels))
test_labels = np.array(pd.get_dummies(test_labels))
```

2

建立模型

In [10]:
```
model = tf.keras.Sequential()
```

In [11]:
```
model.add(tf.keras.layers.Conv2D(filters = 6,kernel_size = (5,5),input_shape=(28,28,1),paddi
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'))
```

In [12]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 5, 5, 16)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1, 1, 120)         48120
_________________________________________________________________
flatten (Flatten)            (None, 120)               0
_________________________________________________________________
dense (Dense)                (None, 84)                10164
_________________________________________________________________
dense_1 (Dense)              (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

3

模型训练

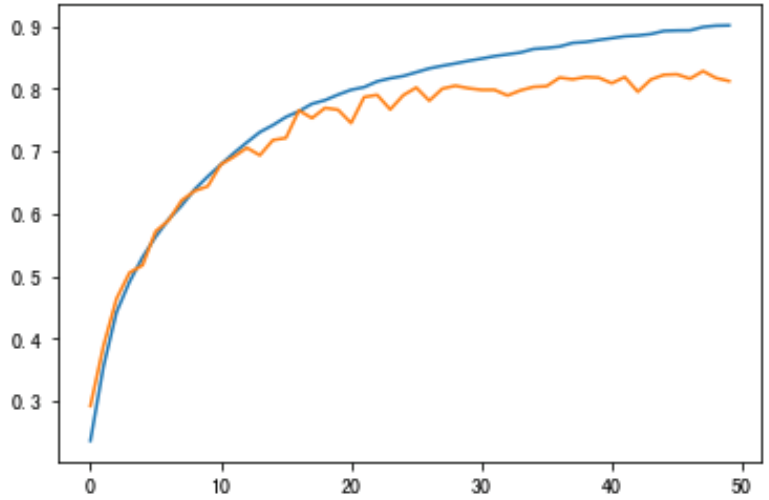
```
In [13]: model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['

In [14]: history = model.fit(train_images,train_labels,epochs = 50,validation_data=
Epoch 1/50
2131/2131 [==============================] - 8s 2ms/step - loss: 2.1519 -
0.2923
Epoch 2/50
2131/2131 [==============================] - 5s 2ms/step - loss: 1.7594 -
0.3882
Epoch 3/50
2131/2131 [==============================] - 5s 2ms/step - loss: 1.5339 -
0.4638
Epoch 4/50
2131/2131 [==============================] - 5s 2ms/step - loss: 1.3987 -
0.5056
Epoch 5/50
2131/2131 [==============================] - 5s 2ms/step - loss: 1.2922 -
0.5175
Epoch 6/50
2131/2131 [==============================] - 5s 3ms/step - loss: 1.2019 -
0.5716
Epoch 7/50

In [15]: plt.plot(history.epoch,history.history.get('acc'))
```

4

模型测试

In [15]:
```
plt.plot(history.epoch,history.history.get('acc'))
plt.plot(history.epoch,history.history.get('val_acc'))
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x7fbf6044f2e0>]
```



In [16]:
```
model.evaluate(test_images,test_labels)
```

```
267/267 [==============================] - 0s 1ms/step - loss: 0.6008 - acc: 0.8123
```

Out[16]:
```
[0.6007986068725586, 0.8122750944137571]
```

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

In [9]:
```
predict
```

Out[9]:
```
array([[9.8759693e-01, 2.6929181e-06, 2.4914041e-08, 2.4818671e-11,
        2.6326370e-06, 6.4962874e-10, 9.4881425e-06, 7.1182288e-04,
        2.6242146e-06, 1.1673761e-02]], dtype=float32)
```

In [10]:
```
label = ['car','harbor','helicopter','oil_gas_field','parking_lot','plane','runway_mark
```

In [12]:
```
label[np.argmax(predict)]
```

Out[12]:
```
'car'
```

总结:

In [2]:
```
train =pd.read_csv('sat/train.csv')
test = pd.read_csv('sat/test.csv')

#从csv文件中载入数据
```

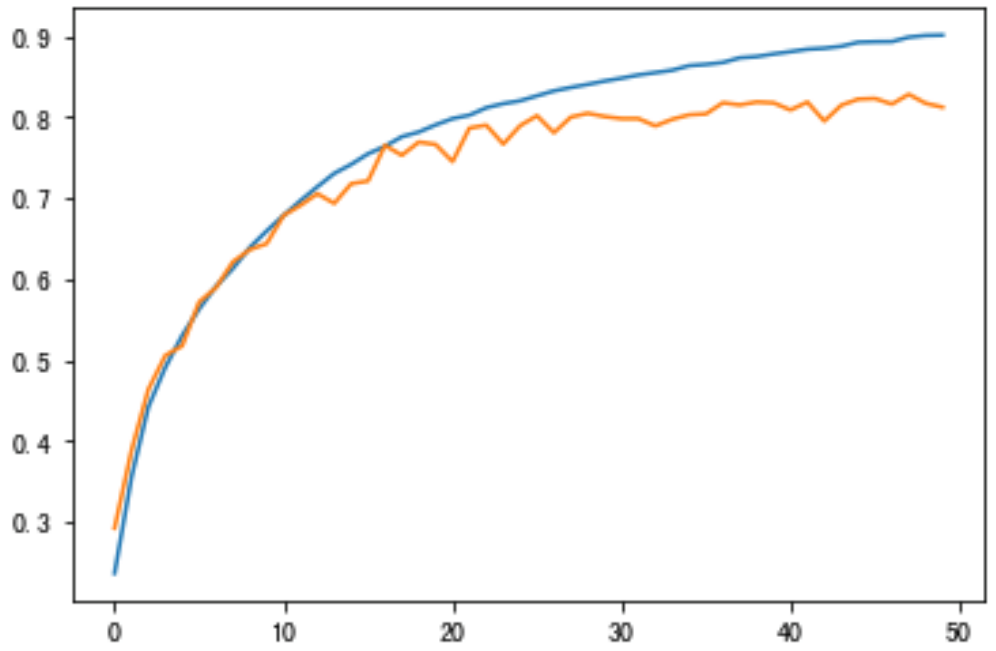
In [3]:
```
train = np.array(train)
test = np.array(test)
```

In [4]:
```
train_images=train[:,1:]
test_images=test[:,1:]

#提取图片信息 28*28
```

In [5]:
```
train_labels=train[:,:1]
test_labels=test[:,:1]

#提取标签信息 1
```

```
plt.plot(history.epoch,history.history.get('acc'))
plt.plot(history.epoch,history.history.get('val_acc'))
```

```
[<matplotlib.lines.Line2D at 0x7fbf6044f2e0>]
```



参考资料：

1. Gradient-Based Learning Applied to Document Recognition
http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

2. Overhead-MNIST
https://www.kaggle.com/datamunge/overheadmnist

3.手写数字识别 1.4 LeNet-5
https://www.bilibili.com/video/BV1Z54y187WW