采用cnn模型实现对猫狗图片的二分类(filter(4*4))

一. 数据加载

通过ImageDataGenerator进行图像的增强与预处理（将图片尺寸转化为50*50，设置分类模式为二分类），将数据归一化，并载入图片。

```
# 加载数据
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255)
train_set =
train_datagen.flow_from_directory('./dataset/training_set',target_size=
(50,50),batch_size=32,class_mode='binary')
```

```
Found 8000 images belonging to 2 classes.
```

二.建立CNN模型

1. 第一层卷积层共有32个filter，每一个filter的大小为4*4，且步长为1.激活函数采用relu
2. 第一层池化层采用最大池化，池化尺寸为2*2，步长为1
3. 第二层卷积层共有32个filter，每一个filter的大小为4*4，且步长为1.激活函数采用relu
4. 第二层池化层采用最大池化，池化尺寸为2*2，步长为1
5. 对处理结果进行展开
6. 建立mlp模型，采用128个神经元，激活函数为relu，输出一个型号，激活函数为sigmoid

```
# 建立cnn模型
from keras.models import Sequential
from keras.layers import Conv2D,MaxPool2D,Flatten,Dense

model = Sequential()
# CONV
model.add(Conv2D(32,(4,4),input_shape=(50,50,3),activation='relu'))
# MaxPool
model.add(MaxPool2D(pool_size=(2,2)))
# CONV
model.add(Conv2D(32,(4,4),activation='relu'))
# MaxPool
model.add(MaxPool2D(pool_size=(2,2)))
# flattening layer
model.add(Flatten())
# FC layer
model.add(Dense(units=128,activation='relu'))
model.add(Dense(units=1,activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 47, 47, 32)        1568
```

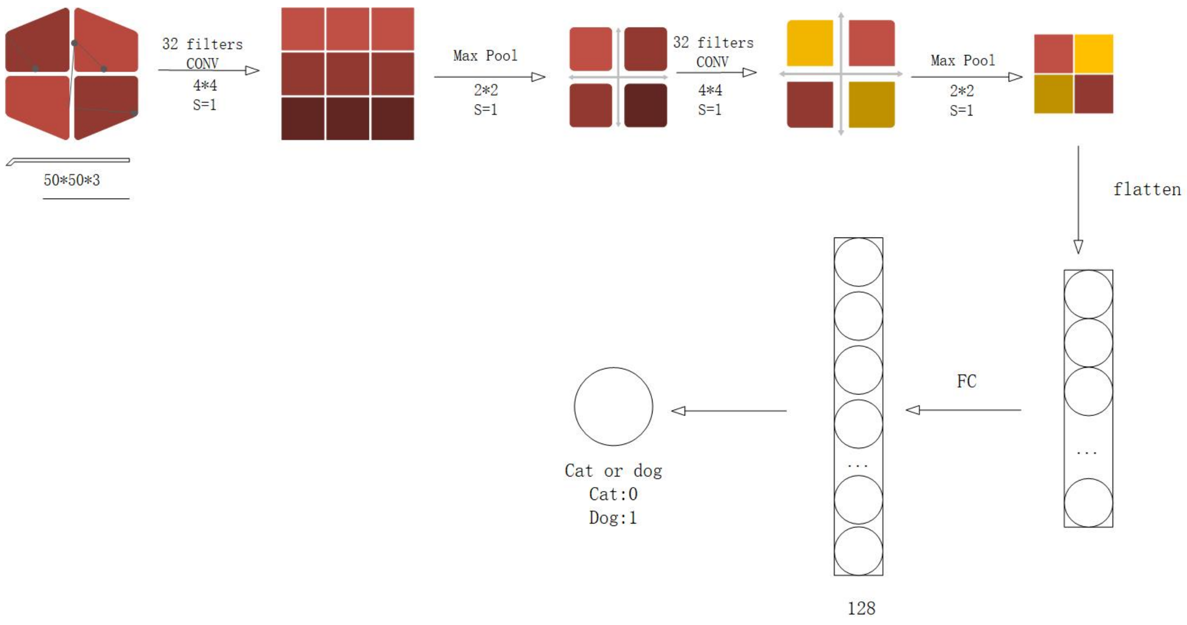
```
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 32)        16416
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               409728
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 427,841
Trainable params: 427,841
Non-trainable params: 0
_________________________________________________________________
```

三.模型求解参数配置与训练

1. 采用adam作为模型的优化器
2. 采用binary_crossentropy作为模型的损失函数
3. 采用metrics=['accuracy']提供模型训练时的准确率
4. 采用30次迭代训练模型

```
# 配置模型
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
# 训练模型
model.fit_generator(train_set,epochs=30)
```



四. 计算模型准确率

对于训练集，模型预测准确率为1.0。
对于测试集，模型预测准确率为0.73。
在模型输出结果中

1. 猫为0
2. 狗为1

```
# 计算模型准确率
accuracy_train = model.evaluate_generator(train_set)
print(accuracy_train)
```

```
[9.22537874430418e-05, 1.0]
```

```
test_set = train_datagen.flow_from_directory('./dataset/test_set',target_size=
(50,50),batch_size=32,class_mode='binary')
accuracy_train = model.evaluate_generator(test_set)
print(accuracy_train)
```

```
Found 2000 images belonging to 2 classes.
[1.2192351818084717, 0.7379999756813049]
```

```
# 查看模型分类
train_set.class_indices
```

```
{'cats': 0, 'dogs': 1}
```

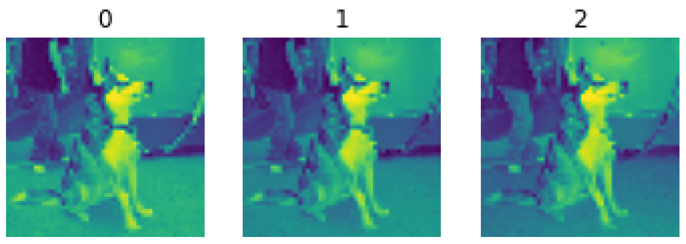
五. 模型中间层可视化

将多维图片分别绘制，并合成

加载图片

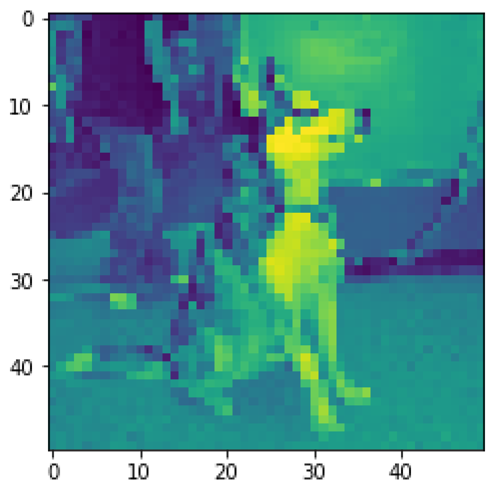
将图片加载，并对图片进行预处理

```
# 加载单张图片
from keras.preprocessing.image import load_img, img_to_array
%matplotlib inline
from matplotlib import pyplot as plt
pic_dog = 'dog.jpg'
pic_dog = load_img(pic_dog,target_size=(50,50))
pic_dog = img_to_array(pic_dog)
fig1 = pic_dog[:,:,0]
for i in range(3):
    fig = pic_dog[:,:,i]
    plt.subplot(1,3,i+1)
    plt.imshow(fig)
    plt.axis("off")
    plt.title(i)
    fig1 = fig1 + pic_dog[:,:,i]
plt.show()
fig1 = fig1 - pic_dog[:,:,0]
plt.imshow(fig1)
pic_dog = pic_dog/255
pic_dog = pic_dog.reshape(1,50,50,3)
result = model.predict_classes(pic_dog)
print(result)
```



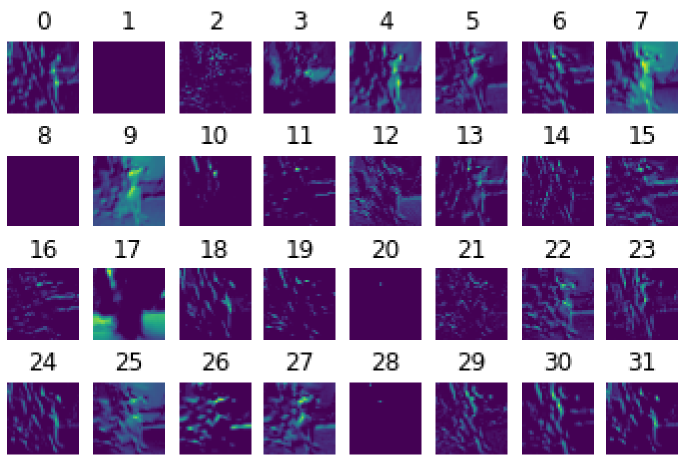
这里模型成功对图片进行了分类。

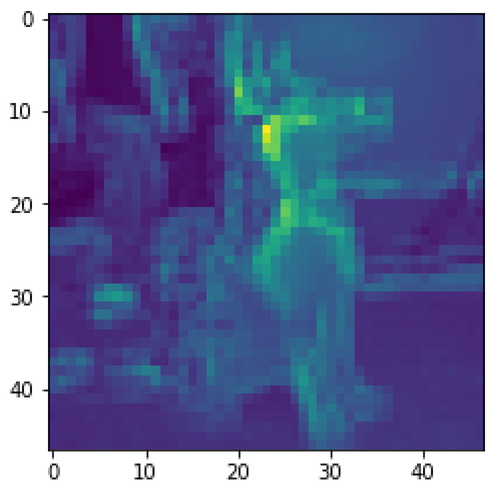
```
[[1]]
```



可视化第一层卷积层

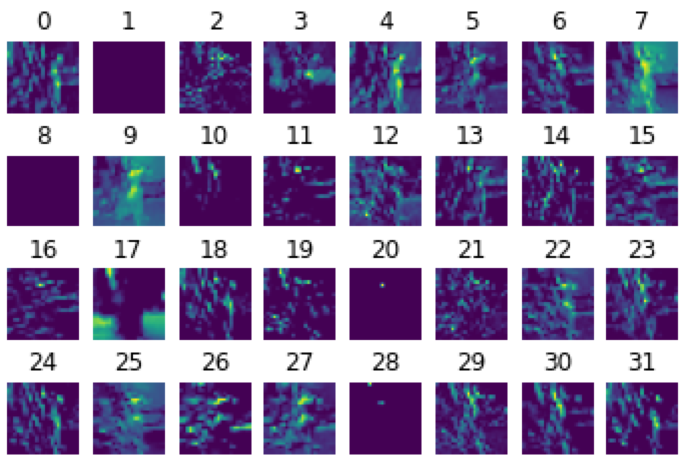
```
# 可视化第一层卷积层
from keras.models import Model
layer_model = Model(inputs = model.input,outputs = model.get_layer(index =
0).output)
result = layer_model.predict(pic_dog)
print(result.shape)
result = result.reshape(47,47,32)
print(result.shape)
fig1 = result[:,:,0]
for i in range(32):
    fig = result[:,:,i]
    plt.subplot(4,8,(i+1))
    plt.axis('off')
    plt.imshow(fig)
    plt.title(i)
    fig1 = fig1 + result[:,:,i]
plt.show()
fig1 = fig1 - result[:,:,0]
plt.imshow(fig1)
```

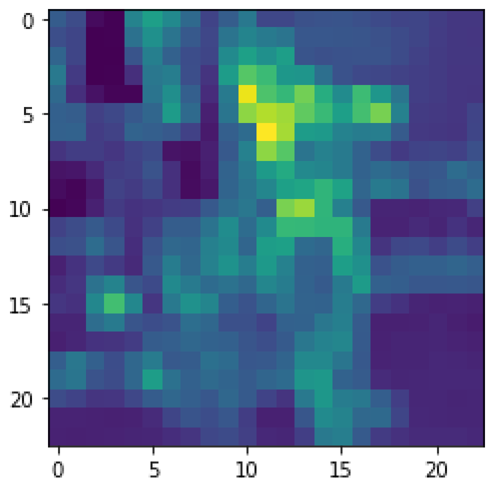




可视化第一层池化层

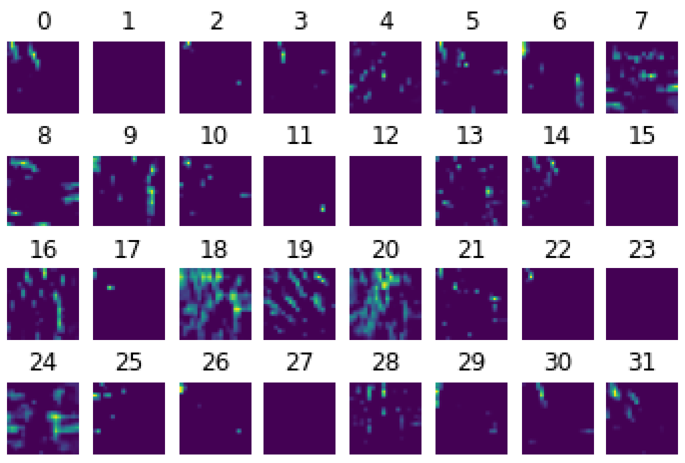
```
# 可视化第一层池化层
layer_model = Model(inputs = model.input,outputs = model.get_layer(index =
1).output)
result = layer_model.predict(pic_dog)
print(result.shape)
result = result.reshape(23,23,32)
print(result.shape)
from matplotlib import pyplot as plt
fig1 = result[:,:,0]
for i in range(32):
    fig = result[:,:,i]
    plt.subplot(4,8,(i+1))
    plt.axis('off')
    plt.imshow(fig)
    plt.title(i)
    fig1 = fig1 + result[:,:,i]
plt.show()
fig1 = fig1 - result[:,:,0]
plt.imshow(fig1)
```

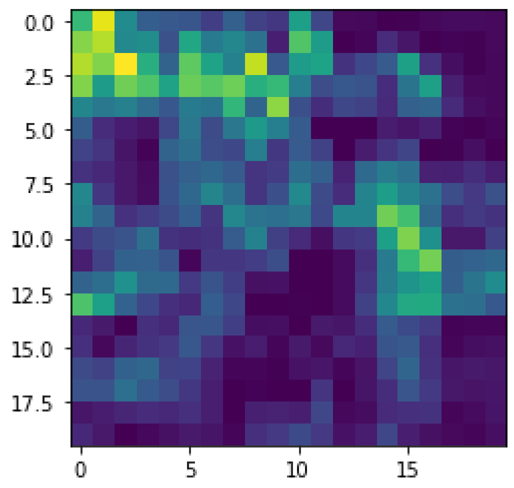




可视化第二层卷积层

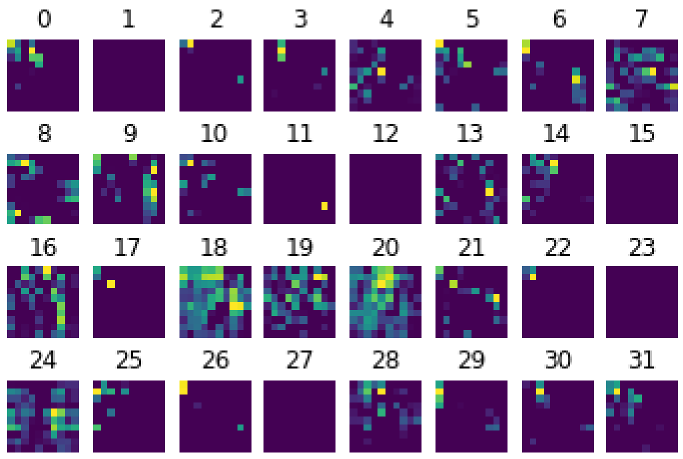
```
# 可视化第二层卷积层
layer_model = Model(inputs = model.input,outputs = model.get_layer(index =
2).output)
result = layer_model.predict(pic_dog)
print(result.shape)
result = result.reshape(20,20,32)
print(result.shape)
from matplotlib import pyplot as plt
fig1 = result[:,:,0]
for i in range(32):
    fig = result[:,:,i]
    plt.subplot(4,8,(i+1))
    plt.axis('off')
    plt.imshow(fig)
    plt.title(i)
    fig1 = fig1 + result[:,:,i]
plt.show()
fig1 = fig1 - result[:,:,0]
plt.imshow(fig1)
```

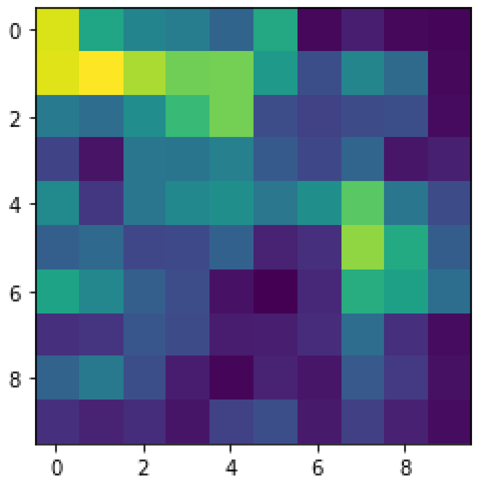




可视化第二层池化层

```
# 可视化第二层池化层
layer_model = Model(inputs = model.input,outputs = model.get_layer(index =
3).output)
result = layer_model.predict(pic_dog)
print(result.shape)
result = result.reshape(10,10,32)
print(result.shape)
from matplotlib import pyplot as plt
fig1 = result[:,:,0]
for i in range(32):
    fig = result[:,:,i]
    plt.subplot(4,8,(i+1))
    plt.axis('off')
    plt.imshow(fig)
    plt.title(i)
    fig1 = fig1 + result[:,:,i]
plt.show()
fig1 = fig1 - result[:,:,0]
plt.imshow(fig1)
```


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31


0
2
4
6
8
0
2
4
6
8