

# 华东理工大学

## 模式识别大作业

题目	<u>猫狗大战问题</u>
院系	<u>信息科学与工程学院</u>
专业	<u>电子信息工程</u>
组员	<u>蒋智鑫</u>
指导老师	<u>赵海清</u>

# 猫狗大战问题

组员：蒋智鑫

## 一、猫狗大战问题简介

“猫狗大战”是 kaggle 上的一个比赛：Dogs vs. Cats。其训练集有 25000 张图片，猫狗各 12500 张，以 class.num.jpg 格式命名。测试集共 12500 张图片，没有标定是猫还是狗，以 num.jpg 格式命名。要求建立一个模型，通过训练集进行训练，并输入测试集图片，输出测试集类别名称。

## 二、整体解决方案

实际上这是一个二分类问题。将猫类标签设置成 ‘0’，狗类标签设置为 ‘1’。对图片的大小进行拉伸、剪裁，将 RGB 三通道像素值输入卷积神经网络模型进行模型训练。但网络完成训练后，将测试集图片的像素值输入模型，并输出预测值。

### 2.1 数据集下载

首先，下载 kaggle 上数据集。由于该数据在境外，造成下载困难，可以通过 csdn 论坛 [https://download.csdn.net/download/qq\\_38210185/10227930](https://download.csdn.net/download/qq_38210185/10227930) 下载数据集。

将数据集中 train 文件将中 cat 和 dog 的图片分别分到两个文件夹下，并将两个文件夹均放在 train 文件夹下。将 train\cat 和 train\dog 的前一千张图片分别移动到 validation 文件夹下的 cat 和 dog 文件夹下，作为验证集使用。将测试集图片放在 test 文件夹下，并将 test 文件夹放在 test 文件夹。文件夹结构如图 1 所示：

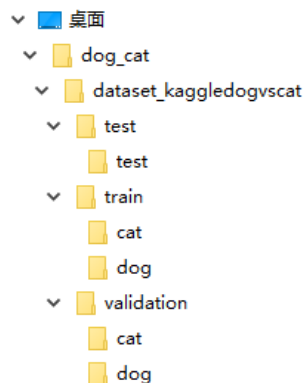


图 1 读入数据的存储文件夹方式

## 2.2 搭建卷积神经网络模型

通过 python 的 keras 模块搭建卷积神经网络。

设计卷积神经网络结构如图 2 所示：

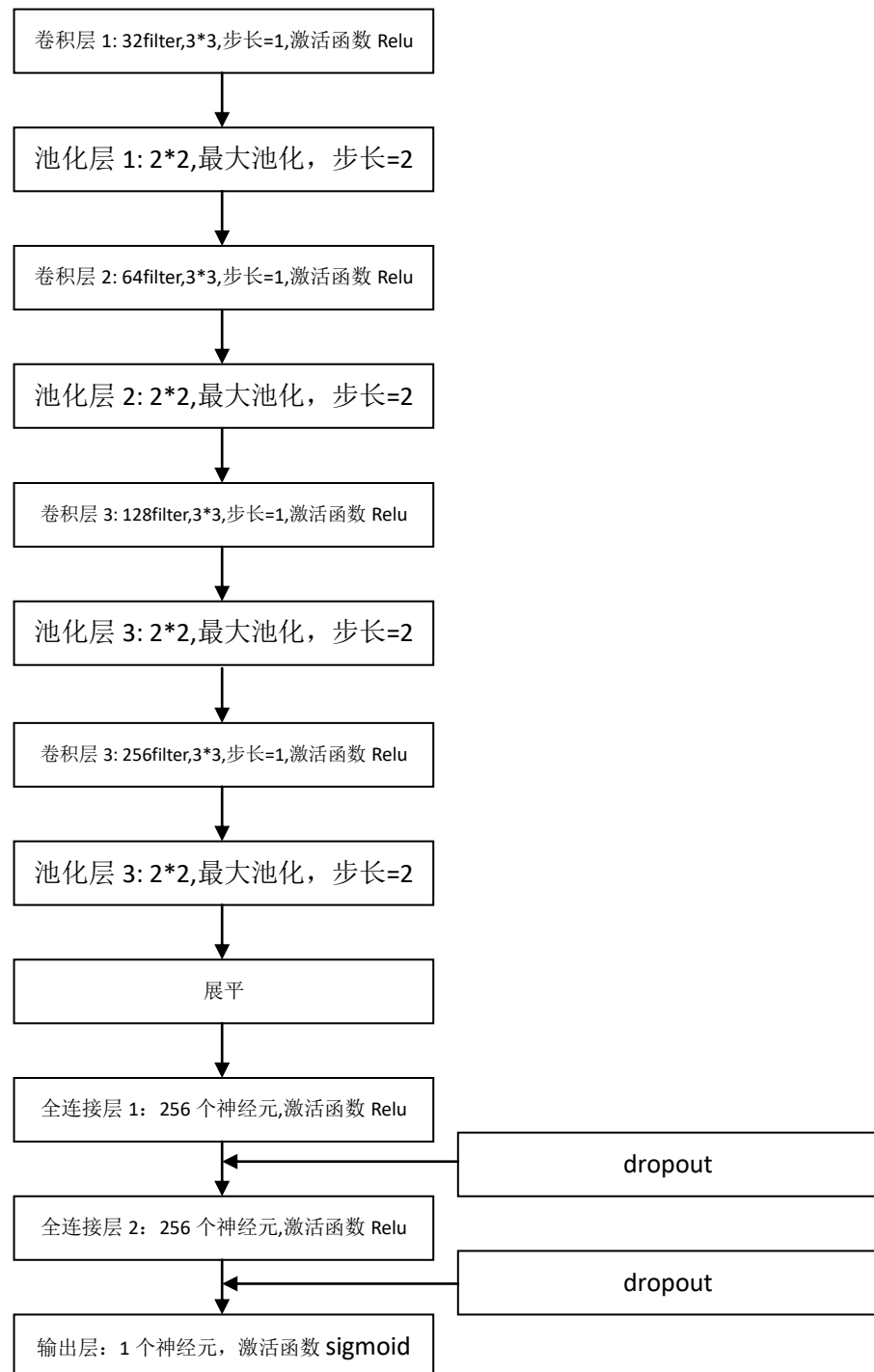


图 2 卷积神经网络模型结构

此模型各层神经元个数、参数个数如下图：

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 128, 32)	896
batch_normalization_1 (Batch Normalization)	(None, 128, 128, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	18496
batch_normalization_2 (Batch Normalization)	(None, 64, 64, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_3 (Conv2D)	(None, 32, 32, 128)	73856
batch_normalization_3 (Batch Normalization)	(None, 32, 32, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 128)	0
conv2d_4 (Conv2D)	(None, 16, 16, 256)	295168
batch_normalization_4 (Batch Normalization)	(None, 16, 16, 256)	1024
max_pooling2d_4 (MaxPooling2D)	(None, 8, 8, 256)	0
flatten_1 (Flatten)	(None, 16384)	0
dense_1 (Dense)	(None, 256)	4194560
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65792
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 2)	514
Total params: 4,651,202		
Trainable params: 4,650,242		
Non-trainable params: 960		

图 3 卷积神经网络神经元、参数个数

卷积神经网络模型程序如下：

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1), padding='same', input_shape=(128, 128, 3),
activation='relu', kernel_initializer='uniform', bias_initializer='zeros'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same', data_format='channels_last'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu',
kernel_initializer='uniform', bias_initializer='zeros'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same', data_format='channels_last'))
model.add(Conv2D(filters=128, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu',
kernel_initializer='uniform', bias_initializer='zeros'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same', data_format='channels_last'))
model.add(Conv2D(filters=256, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu',
kernel_initializer='uniform', bias_initializer='zeros'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same', data_format='channels_last'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='sigmoid'))
```

## 2.3 训练网络

首先，读入训练集。Keras 提供了非常方便的 `ImageDataGenerator()` 函数可以用来读取训练集的图片。以 100 张图片为一个 `batch`，输入模型中进行训练。同时，以同样的方式读取验证集，并设置每 50steps 验证一次。模型优化器选择随机梯度下降（SGD），学习率取 0.001。在训练集上总共训练 35 个 `epoch`。

程序如下：

```
train_file_path = 'C:/Users/Lenovo/Desktop/dog_cat/dataset_kaggledogvscat/train'
validation_file_path = 'C:/Users/Lenovo/Desktop/dog_cat/dataset_kaggledogvscat/validation'
idg = ImageDataGenerator()
train_generator = idg.flow_from_directory(train_file_path, (128, 128), shuffle=True, batch_size=100,
class_mode='categorical')
validation_generator = idg.flow_from_directory(validation_file_path, (128, 128), shuffle=True, batch_size=100,
class_mode='categorical')
sgd = SGD(lr=0.001)
model.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
```

```

model.summary()

history=model.fit_generator(train_generator,epochs=35,steps_per_epoch=23000//100,validation_data=validation_generator,
validation_steps=10)

```

部分训练结果如下图所示：

```

197/230 [=====>.....] - ETA: 6s - loss: 0.3786 - acc: 0.8338
198/230 [=====>.....] - ETA: 6s - loss: 0.3791 - acc: 0.8332
199/230 [=====>.....] - ETA: 6s - loss: 0.3791 - acc: 0.8331
200/230 [=====>.....] - ETA: 6s - loss: 0.3787 - acc: 0.8332
201/230 [=====>.....] - ETA: 5s - loss: 0.3784 - acc: 0.8334
202/230 [=====>.....] - ETA: 5s - loss: 0.3784 - acc: 0.8332
203/230 [=====>.....] - ETA: 5s - loss: 0.3780 - acc: 0.8336
204/230 [=====>.....] - ETA: 5s - loss: 0.3777 - acc: 0.8339
205/230 [=====>.....] - ETA: 5s - loss: 0.3774 - acc: 0.8341
206/230 [=====>.....] - ETA: 4s - loss: 0.3775 - acc: 0.8340
207/230 [=====>.....] - ETA: 4s - loss: 0.3780 - acc: 0.8336
208/230 [=====>.....] - ETA: 4s - loss: 0.3783 - acc: 0.8333
209/230 [=====>.....] - ETA: 4s - loss: 0.3781 - acc: 0.8335
210/230 [=====>.....] - ETA: 4s - loss: 0.3778 - acc: 0.8338
211/230 [=====>.....] - ETA: 3s - loss: 0.3777 - acc: 0.8336
212/230 [=====>.....] - ETA: 3s - loss: 0.3777 - acc: 0.8336
213/230 [=====>.....] - ETA: 3s - loss: 0.3776 - acc: 0.8338
214/230 [=====>.....] - ETA: 3s - loss: 0.3777 - acc: 0.8338
215/230 [=====>.....] - ETA: 3s - loss: 0.3778 - acc: 0.8337
216/230 [=====>.....] - ETA: 2s - loss: 0.3775 - acc: 0.8338
217/230 [=====>.....] - ETA: 2s - loss: 0.3777 - acc: 0.8338
218/230 [=====>.....] - ETA: 2s - loss: 0.3779 - acc: 0.8334
219/230 [=====>.....] - ETA: 2s - loss: 0.3780 - acc: 0.8333
220/230 [=====>.....] - ETA: 2s - loss: 0.3780 - acc: 0.8333
221/230 [=====>.....] - ETA: 1s - loss: 0.3783 - acc: 0.8331
222/230 [=====>.....] - ETA: 1s - loss: 0.3785 - acc: 0.8332
223/230 [=====>.....] - ETA: 1s - loss: 0.3791 - acc: 0.8330
224/230 [=====>.....] - ETA: 1s - loss: 0.3789 - acc: 0.8329
225/230 [=====>.....] - ETA: 1s - loss: 0.3787 - acc: 0.8332
226/230 [=====>.....] - ETA: 0s - loss: 0.3788 - acc: 0.8331
227/230 [=====>.....] - ETA: 0s - loss: 0.3788 - acc: 0.8331
228/230 [=====>.....] - ETA: 0s - loss: 0.3792 - acc: 0.8329
229/230 [=====>.....] - ETA: 0s - loss: 0.3789 - acc: 0.8329
230/230 [=====] - 48s 210ms/step - loss: 0.3792 - acc: 0.8327 - val_loss: 0.3798 - val_acc: 0.8290

```

图 4 部分训练结果

每个 epoch 损失如图 5 所示：

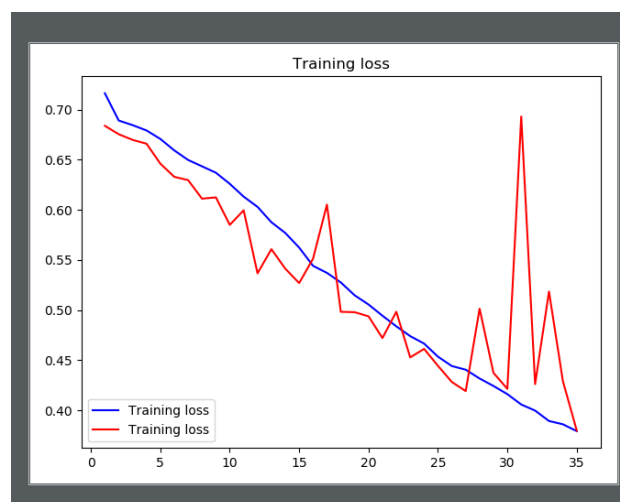


图 5 loss 曲线（蓝色为训练集、红色为验证集）

每个 epoch 准确率如图 6 所示：

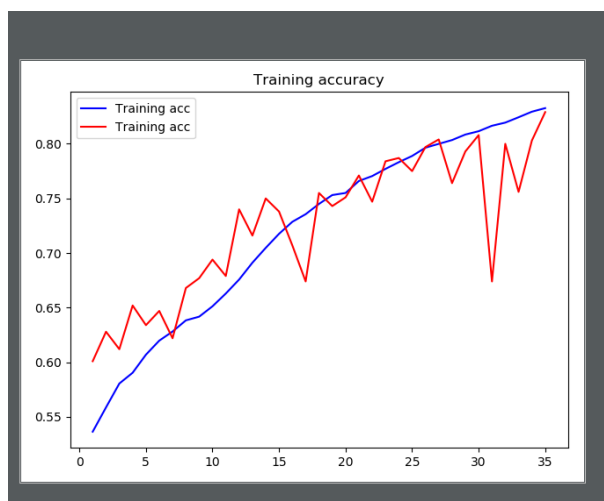


图 6 accuracy 曲线（蓝色为训练集、红色为验证集）

读取测试集数据，程序如下：

```
def get_test(batch_size):
    test = np.zeros([12500, 128, 128, 3])
    for i in range(12500//batch_size):
        test[i*batch_size:(i+1)*batch_size, :, :, :] = test_generator.next()
    return test

test_file_path = 'C:/Users/Lenovo/Desktop/dog_cat/dataset_kaggledogvscat/test'
test_generator = idg.flow_from_directory(test_file_path, (128, 128), shuffle=False, batch_size=100, class_mode=None)
test = get_test(100)
```

将测试集数据输入模型得到预测类别并输出。

```
pre = model.predict_classes(test, batch_size=100, verbose=1)
cla = []
output_path = 'C:/Users/Lenovo/Desktop/dog_cat/dataset_kaggledogvscat/class.csv'
for i in range(len(pre)):
    if pre[i] == 0:
        cla.append('cat')
    else:
        cla.append('dog')
df = DataFrame({'class': cla})
df.to_csv(output_path)
```

### 三、实验结论

本实验通过该卷积神经网络模型对此问题进行了处理，经过 35 次迭代，模型对验证集取得了 82.9% 的准确率。此时，模型的 loss 仍在下降，对验证集准确率仍在提高。如果继续训练，模型准确率可能可以继续提高。但在实验中也遇到了很多问题。模型的最后训练结果并不十分理想，训练集的结果较之训练集和验证集更加存在问题，模型仍需要继续优化。这将作为后续工作而继续进行。