

机器学习毕业项目

- yijigao 2018/08/20

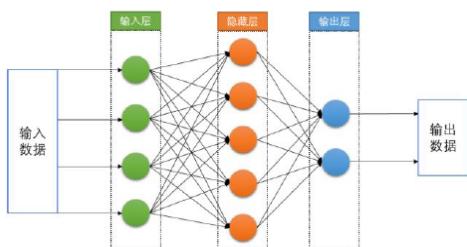
1. 问题的定义

1.1 项目概述

从图像中判断一张图是猫还是狗，似乎是一个非常容易的问题，恐怕三岁小孩就能做到。让人失望的是，人类拥有最先进的机器和计算设备，依然在图像识别方面表现非常吃力[1]。

所幸的是，随着近些年处理器、图形处理器的性能的飞速提升，带来了机器学习、深度学习领域的飞速发展，计算机已经拥有足够的图形处理性能，可以通过大量的训练标记数据，经过深度学习算法，开始逐步“学会”去识别图片。这也是本项目的目标，让计算机“学会”判断一张图上的动物是猫，还是狗。

深度神经网络作为深度学习技术的一种，由于相对于传统机器学习技术具有更高的准确度，目前已成为解决图像识别问题的最主要的工具。神经网络顾名思义是由神经元组成的网络，一个神经元根据输出值通过激活函数计算输出值。通常的激活函数由sigmoid函数， ReLu函数等。将多个神经元彼此连接起来就构成了神经网络。经典的神经网络由输入层和输出层组成，每一层都可以有多个神经元构成。如果在输入层和输出层之间加入更多的层（隐藏层）则构成了深度神经网络。



本项目数据来源于Kaggle[2]的一个竞赛项目。对于此类问题，目前主流的图像识别分类方法是采用深度学习卷积神经网络（Convolutional neural net,CNN）[3]。卷积神经网络通过卷积层和池化层来实现对图像特征的提取和筛选。目前流行的深度学习框架有TensorFlow、Caffe、Keras。

1.2 问题陈述

本项目需要解决的问题是从12500张包含猫或狗的图像中，对图片进行猫和狗分类区分。所以这是一个二分类问题。对于给定的图片，设计算法判断图片中的动物是猫还是狗。

- 输入：一张包含猫/狗的图片
- 输出：图片是猫还是狗

输出值是图片为狗的概率

1.3 评价指标

模型评估标准是交叉熵LogLoss函数，log loss 越小表示神经网络对于数据集有着较好的分类效果。

$$LogLoss = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- n 表示测试集的图像数量
- \hat{y}_i 表示图像是狗的概率
- y_i 1 代表图像是狗, 0 表示是猫
- $\log()$ 代表自然对数

2. 分析

2.1 数据的探索与可视化

数据集来源是Kaggle[3]，训练集是25000张包含猫或狗图像，每张图像都已经在文件名标注好了猫/狗。猫狗图片数量各占一半。而测试集则是12500张未标注猫狗属性的图像。



cat.4490.jpg



cat.4491.jpg



cat.4492.jpg



cat.4498.jpg



cat.4499.jpg



cat.4500.jpg



cat.4506.jpg



cat.4507.jpg



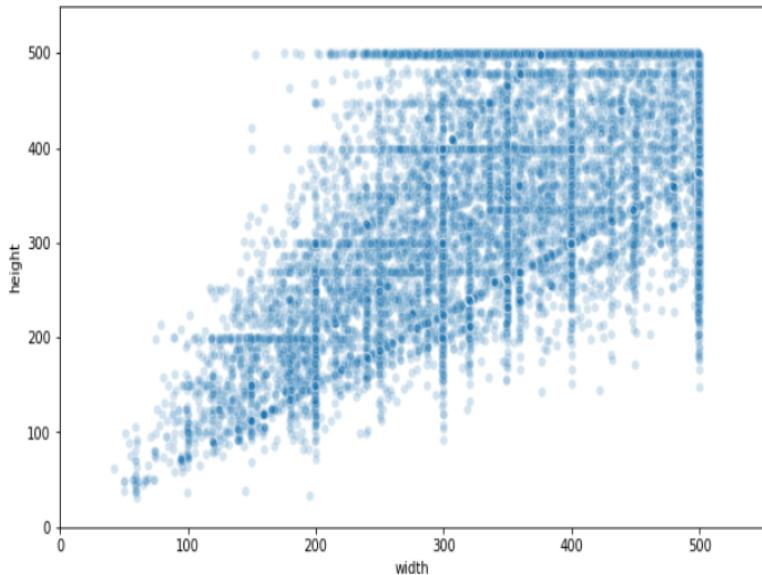
cat.4508.jpg

从训练集中的大部分图来看，照片多为日常拍摄，清晰度不错，人眼能很好的识别出猫和狗。当然也有部分图片是由人抱着动物拍的，而且图中的猫狗太小，很难分辨，这种图片属于“脏

数据", 如果这些数据会对结果造成大的影响, 则需要将其删除。



另外, 也需要注意到图片尺寸各不一样, 高度和宽度分布都从几十到500以上, 这对于数据训练来说是不利的, 为了使输入图像的尺寸一致, 项目中将用到Keras的图片生产器 (ImageDataGenerator) 对图像尺寸进行变换。



2.2 算法与方法

2.2.1 卷积神经网络

卷积神经网络 (Convolutional Neural Network, CNN) 是一种前馈神经网络, 它由一个或多个卷积层和顶端的全连接层 (对应经典的神经网络) 组成, 同时也关联权重和池化层。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习, 卷积神经网络需要考量的参数更少。

以手写数字识别任务为例。输入层是 32×32 的手写数字图像, 输出层是识别结果卷积层和采样层对信号进行处理, 连接层实现与输出目标的映射。其中每一个卷积层都包含多个特征映射, 每个特征映射是一个由多元神经元构成的平面, 通过卷积滤波器提取输入的特征。采样层又称未池化层, 作用是基于局部卷积滤波器原理进行采样, 在减少数据量的同时保留有用信息。

2.2.2 迁移学习

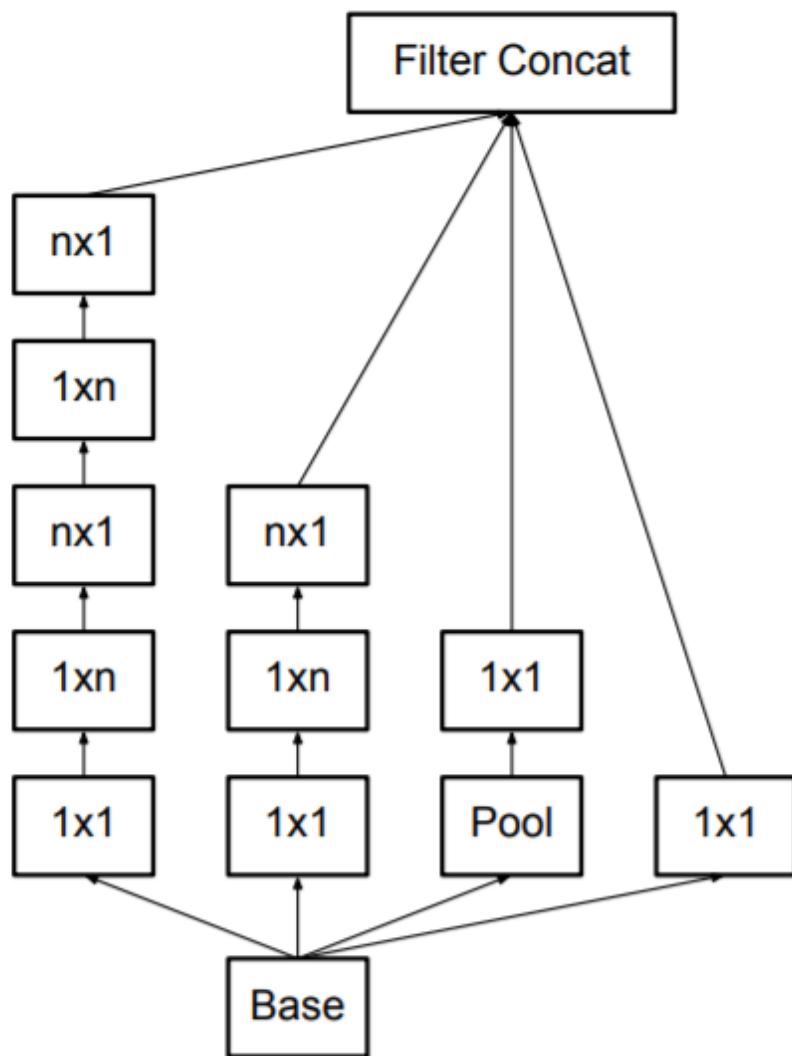
本项目采用迁移-融合学习的方法来构建训练神经网络。迁移学习 (Transfer Learning) 是指使用在一个任务中预训练好的模型。将此模型重新用于第二个相关任务的机器学习技术[4]。

在本项目中，我将使用多个已经在ImageNet上完成训练的网络模型进行迁移。ImageNet任务就是图像识别分类，因此在ImageNet数据上训练好的模型本身对于图像特征有很好的提取能力，可以说，使用预训练好的模型迁移至猫狗分类是可行的，并且相对于专门构建模型会有更好的训练效果。

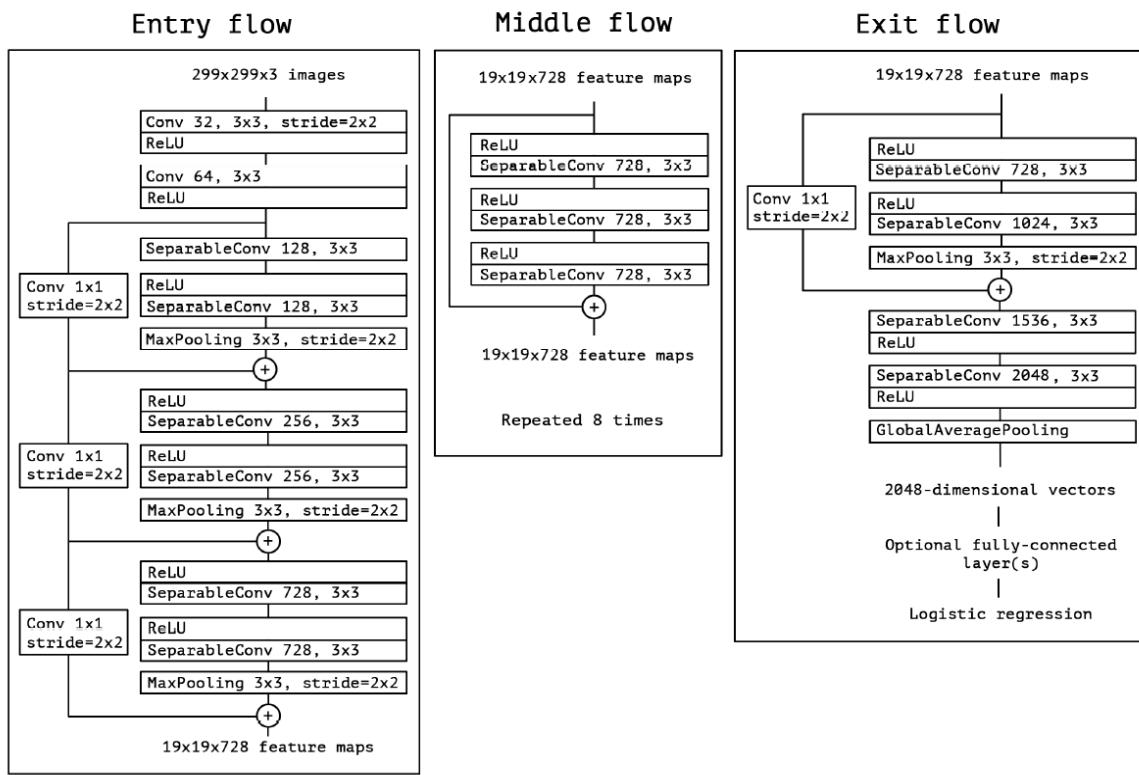
2.2.3 预训练网络

本项目预训练网络采用了Xception[5], InceptionV3[6], DenseNet201[7]。

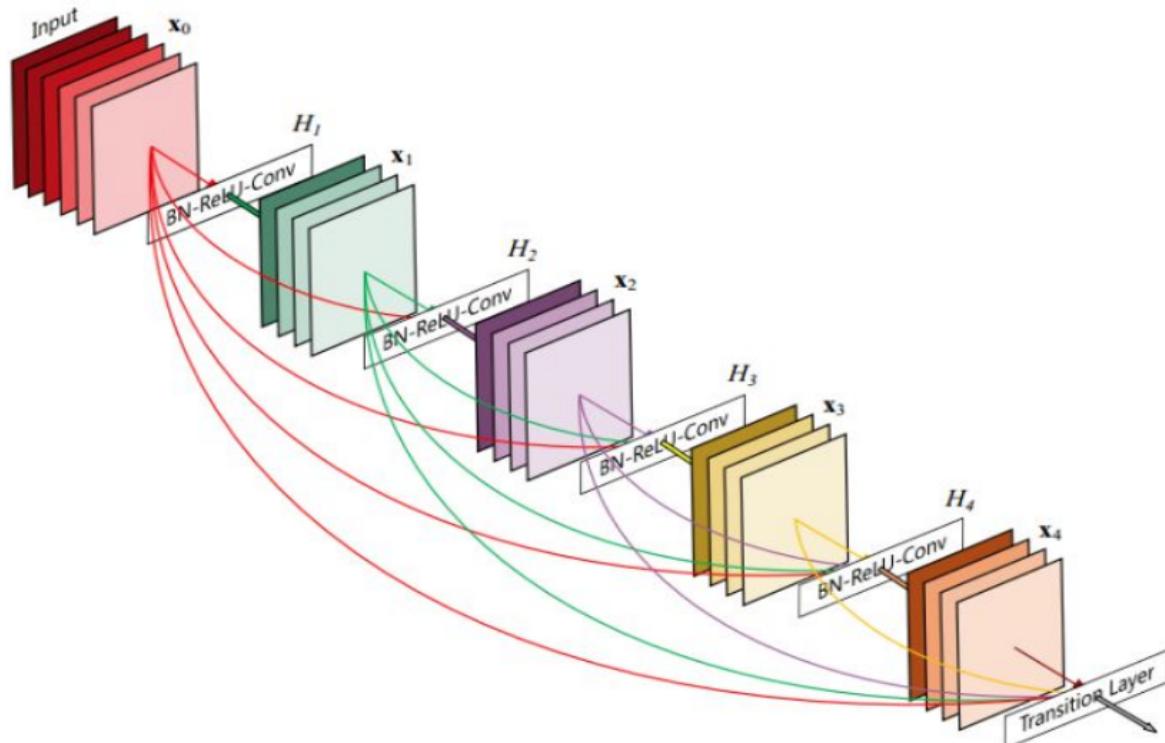
Inception是由谷歌发表的一项深度学习网络模型，Inception网络将 1×1 , 3×3 , 5×5 的conv和 3×3 的pooling, stack在一起，一方面增加了网络的width，另一方面增加了网络对尺度的适应性，经过几个版本的更新，Inception V3将 7×7 , 3×3 分解成两个一维的的卷积，这样既可以加速计算，又可以将1个conv拆分成2个conv，使网络深度进一步增加，增加了网络的非线性。



Xception是一种轻量化模型，由Google 在2016年10月发表，Xception基于Inception V3, 其结构如下图所示，分为Entry flow, Middle flow, Exit flow，其中Entry flow 包含8层卷积，Middle flow 包含24层卷积，而Exit flow 包含4层卷积，共计36层。



DenseNet基于ResNet提出的skip-layer思想[8]，设计了一种全新的连接模式。为了最大化网络中的所有层之间的信息流，作者将网络中的所有层两两都进行了连接，使得网络中每一层都接受它前面所有层的特征作为输入。



2.3 基准测试

项目要求是达到kaggle top10%。目前Kaggle该项目Leaderboard一共1314参赛选手，第131名成绩是0.06127。因此基准得分必须小于0.06127。

本项目基础模型采用Xception, Xception的优势是，在给定的硬件资源下，可以尽可能的增加网络效率和训练性能。

使用Xception进行迁移学习得到的kaggle得分为0.1135, 不能满足要求，而开放97层以上权重后，得分0.04664。小于项目要求的0.06127。本项目将使用这个结果作为基准，改进模型，希望得到更高的得分

3. 方法

3.1 数据预处理

之前提到，我们将使用迁移学习（Transfer Learning）的方法来训练、预测。使用Xception、DenseNet201、InceptionV3神经神经网络结构。而使用Keras的ImageDataGenerator需要将不同种类的图片分在不同的文件夹中，并且，Xception默认的图片尺寸299x299，而原始数据图片尺寸大小不一，因此需要对图片进行缩放或裁剪。

解压原始数据集 train.zip , test.zip

异常值处理

该部分代码见 Xception 异常值清理.ipynb

前面已经指出，数据集存在异常值，有部分图片既非狗，也非猫。但是训练集有25000张照片，人工处理太费时费力。参考使用预训练模型自动检测并剔除异常数据的方法[9] 预训练模型选用的是Xception，猫狗种类下载参考文献种类列表

```
model_err = Xception(weights='imagenet')

def is_err_img(img_path):
    img = image.load_img(img_path, target_size=(299, 299))
    x = image.img_to_array(img)

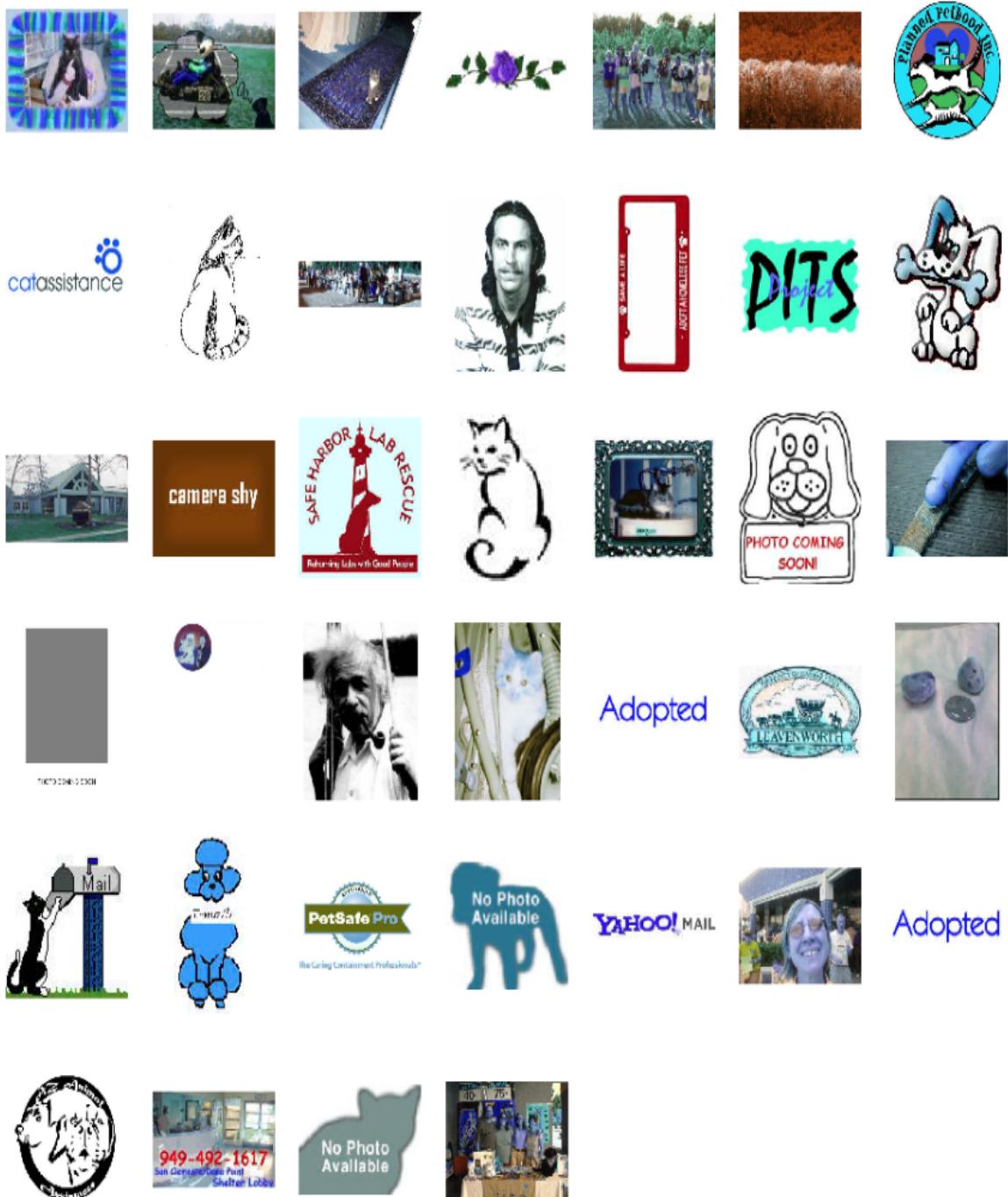
    x = preprocess_input(x)
    preds = np.array(decode_predictions(model_err.predict(x), top=40)[0])

    is_err = True
    for pred in preds[:, 0]:
        if pred in dogs or pred in cats:
            is_err = False
    return is_err

err_list = []

for img in os.listdir('train/'):
    if is_err_img(f'train/{img}'):
        err_list.append(img)
```

使用预训练模型， top=40， 一共抓出39张异常图片， 看起来缺失都是非猫非狗图片。说明预训练模型处理很成功。39张异常图将在训练时删除



按猫狗对文件进行分类

```
train_files = os.listdir('train/')
train_cat = [x for x in train_files if 'cat' in x]
train_dog = [x for x in train_files if 'dog' in x]
```

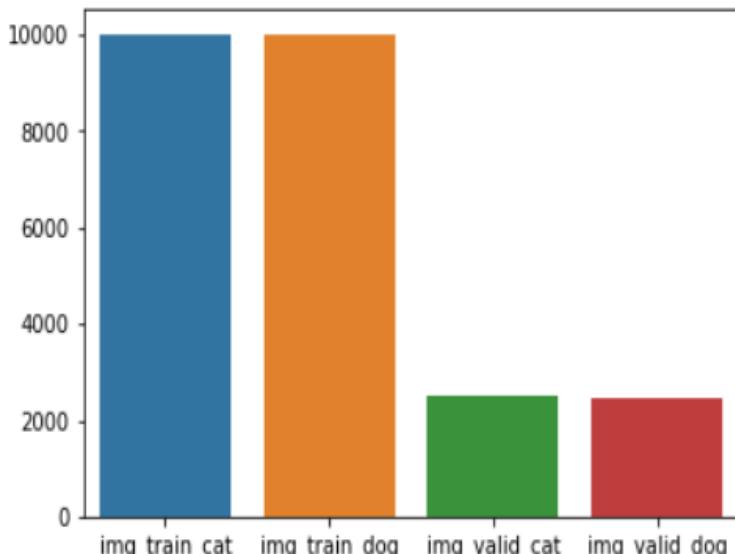
分类完成后，各个文件夹图片数量

```
>>> count = f'猫: {len(train_cat)}}, 狗: {len(train_dog)}}, 测试集: {len(os.listdir("test,
>>> count
'猫: 12500, 狗: 12500, 测试集: 12500'
```

按4：1 拆分训练集和验证集

```
>>> from sklearn.model_selection import train_test_split  
>>> img_train, img_valid = train_test_split(train_files, test_size=0.2, random_state = 42)  
>>> len(img_train), len(img_valid)
```

20000, 5000



创建符号连接

创建符号连接的好处是，不用手动再去复制一遍图片，避免不必要的麻烦，节省空间和时间

```
def remove_and_mkdir(dirname):  
    if os.path.exists(dirname):  
        shutil.rmtree(dirname)  
    os.mkdir(dirname)  
    if dirname == 'img_test':  
        os.mkdir(f'{dirname}/test')  
    else:  
        os.mkdir(f'{dirname}/cat')  
        os.mkdir(f'{dirname}/dog')  
  
remove_and_mkdir('img_train')  
remove_and_mkdir('img_valid')  
remove_and_mkdir('img_test')  
  
img_test_files = os.listdir("test/")  
  
for filename in img_train_cat:  
    os.symlink('../..../train/'+filename, 'img_train/cat/'+filename)  
  
for filename in img_train_dog:
```

```
os.symlink('../train/'+filename, 'img_train/dog/'+filename)

for filename in img_valid_cat:
    os.symlink('../train/'+filename, 'img_valid/cat/'+filename)

for filename in img_valid_dog:
    os.symlink('../train/'+filename, 'img_valid/dog/'+filename)

for filename in img_test_files:
    os.symlink('../test/'+filename, 'img_test/test/'+filename)
```

使用ImageDataGenerator预处理

- 像素缩放到0和1
- 照片统一尺寸 299*299

```
from keras.preprocessing.image import ImageDataGenerator

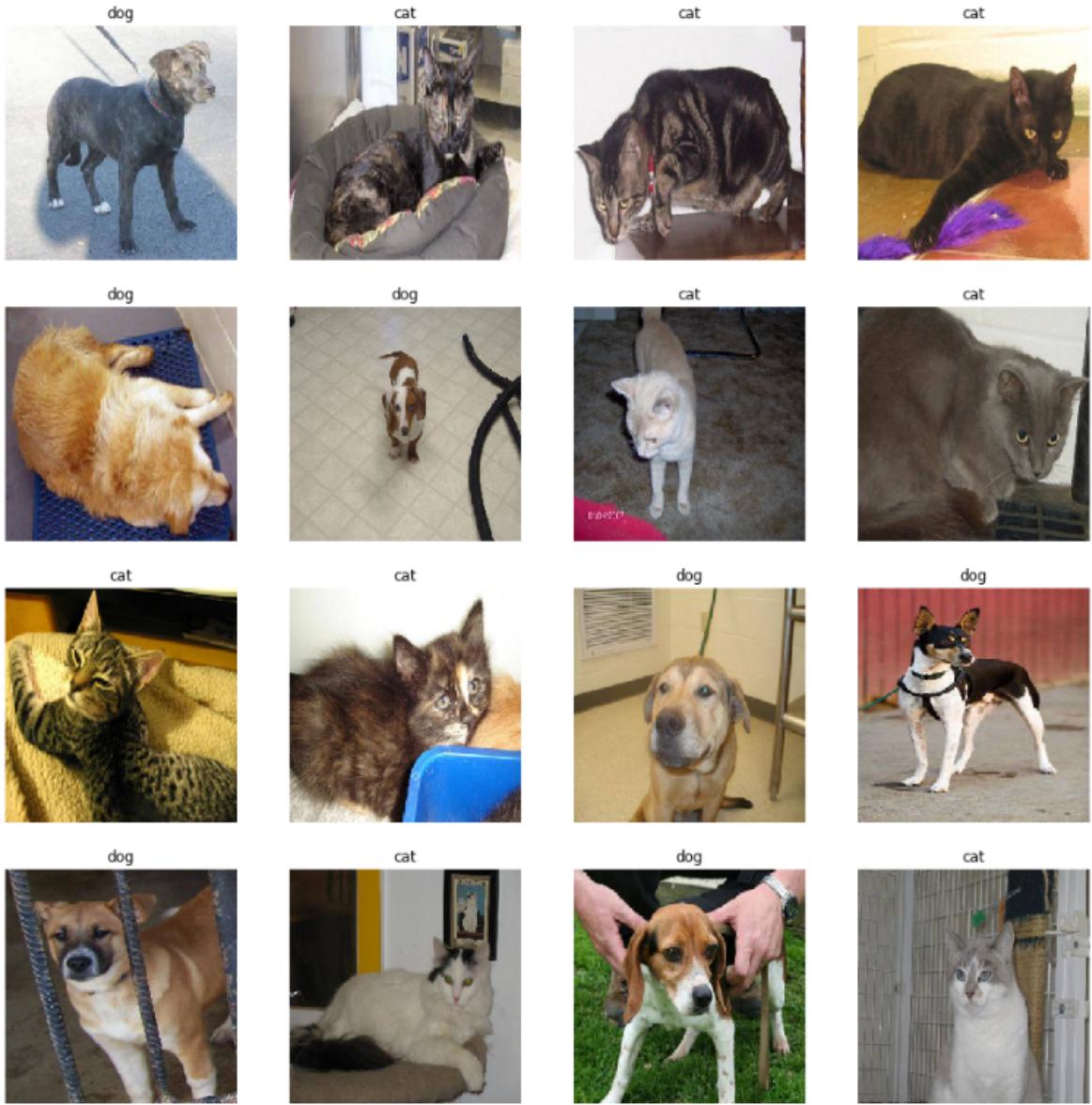
target_image_size = (299, 299)

train_datagen = ImageDataGenerator(rescale=1.0/255)

train_generator = train_datagen.flow_from_directory(
    'img_train',
    target_size=target_image_size, # resize
    batch_size=16,
    class_mode='binary')

validation_datagen = ImageDataGenerator(rescale=1.0/255)
validation_generator = validation_datagen.flow_from_directory(
    'img_valid',
    target_size=target_image_size, # resize
    batch_size=16,
    class_mode='binary')
```

统一尺寸后的图片如下图所示



3.2 实施

该部分代码详见 `base_model_xception.ipynb`

该部分包含以下步骤

- 构建Xception神经网络
- 训练数据，按照Val_loss调整参数
- 保存最佳模型
- 使用最佳模型预测测

初次使用`keras.application.Xception`预训练模型，固定所有ImageNet权重，只允许分类器被训练。损失函数使用交叉熵cross-entropy, 优化器使用adadelta, 使用`dropout=0.5`防止模型过拟合。经过5代训练后，得分为0.11315，显然达不到项目要求。而开放97层以上权重后，得分达到0.04664，`val_loss=0.0360, val_acc=0.9920`。低于目标的0.0617。我们将以此结果作为基准。

3.3 改进

该部分详细代码见 [迁移-融合学习.ipynb](#)， 方法参考了[10]

单独使用开放权重后的Xception就已经能到达项目要求，以上述得分为基准，需要获得比基准更好的得分。参考mentor-杨培文的经验[10]，综合多个不同的模型，将各个模型的网络输出的特征向量保存下来，综合三个模型的训练结果，可以获得更高的准确率，从而提高得分。

因此，借鉴上述参考资料的已有经验，我选择使用融合Xception, Densenet201, InceptionV3这三个模型，分别预训练，导出特征向量。各个子模型的权重采用ImageNet权重，去掉了各个模型的分类器（Top层），并将GlobalAveragePooling2D的输出合并传递给分类器（output layer）。训练时，固定除了分类器以外的权重。

```
def write_feature_data(MODEL, image_shape, train_data, test_data, batch_size, preprocess_input_tensor = Input((image_shape[0], image_shape[1], 3)))
x = input_tensor
if preprocess_input:
    x = Lambda(preprocess_input)(x)

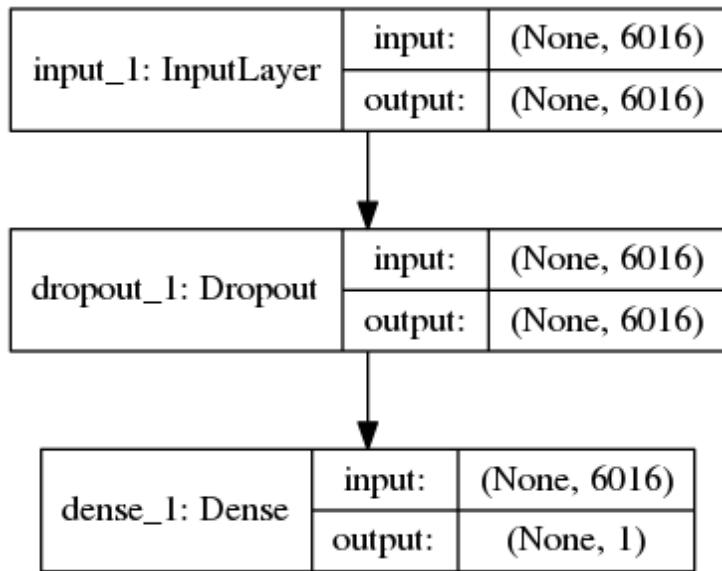
base_model = MODEL(input_tensor=x, weights='imagenet', include_top=False)
base_model.save_weights(f'{base_model.name}-imagenet.h5')

model = Model(base_model.input, GlobalAveragePooling2D()(base_model.output))

gen = ImageDataGenerator()
train_generator = gen.flow_from_directory(train_data, image_shape, shuffle=False,
                                         batch_size=batch_size)
test_generator = gen.flow_from_directory(test_data, image_shape, shuffle=False,
                                         batch_size=batch_size, class_mode=None)
train_feature = model.predict_generator(train_generator, train_generator.samples,
                                         verbose=0)
test_feature = model.predict_generator(test_generator, test_generator.samples, verbose=0)
with h5py.File(f"feature_{base_model.name}.h5") as h:
    h.create_dataset("train", data=train_feature)
    h.create_dataset("test", data=test_feature)
    h.create_dataset("label", data=train_generator.classes)

write_feature_data(Xception, (299, 299), train_data, test_data, batch_size=1, preprocess_input=True)
write_feature_data(DenseNet201, (224, 224), train_data, test_data, batch_size=1, preprocess_input=True)
write_feature_data(InceptionV3, (299, 299), train_data, test_data, batch_size=1, preprocess_input=True)
```

依次得到3个特征向量文件 feature_densenet201.h5, feature_inception_v3.h5, feature_xception.h5，然后构建模型，融合模型分类器如下图所示。其中输入特征为合并了的特征向量，模型采用 adadelta 优化器训练，保留20%的数据作为验证集，在训练过程保证存在验证集上损失函数最小的模型权重。经过20代训练，val_loss最小能达到0.0156，val_acc达0.9956

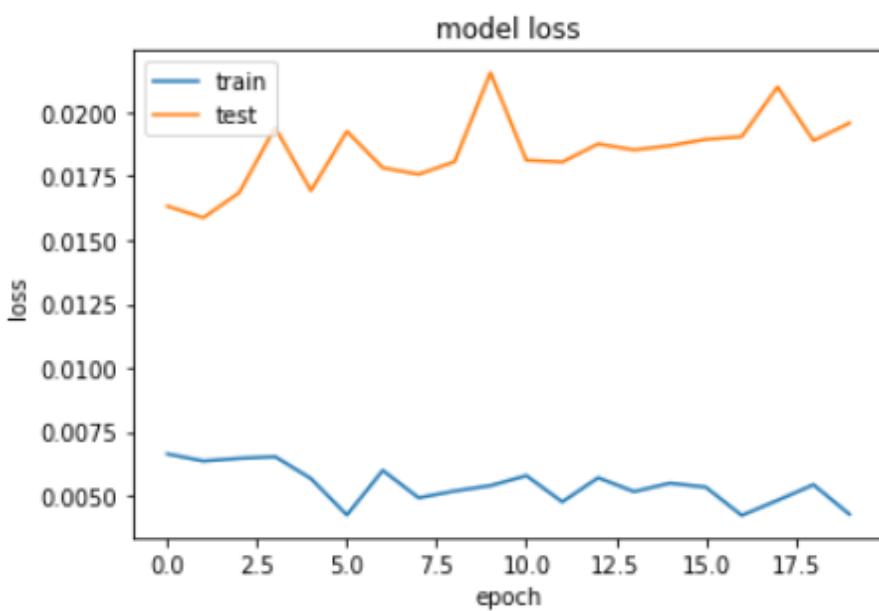
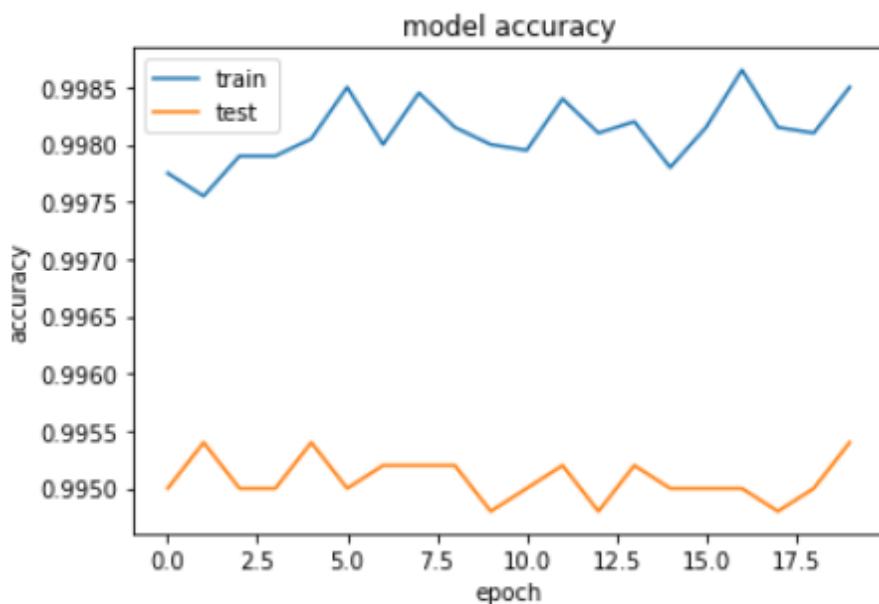


在测试集上预测并提交kaggle进行提交，最终得到kaggle得分为 0.03834。为避免预测数据为0或1带来的log函数溢出问题，采用clip方式，使预测结果最小值为0.005，最大值为0.995。

4. 结果

4.1 模型评估与验证

融合模型训练过程acc和loss数据如下。使用融合模型很快达到收敛，val_acc稳定在0.995以上，val_loss在0.012左右徘徊，说明模型准确性非常好。



如上所述，本项目最终选用基于Xception、DenseNet201、InceptionV3三个基础模型建立的融合模型，相比基准测试使用的单一Xception模型，得分得到明显提升。该模型最终在测试集上的得分是 0.03834，满足了项目要求。

4.2 改进

最初的融合模型没有去除异常值，考虑将异常值可能造成的影响，剔除了39张异常图片，使用同样的模型参数进行训练。异常值查找按以上方法剔除。使用去除异常值后的训练值训练模型后，得分提升至了 0.03796。

4.3 预测结果查看

随机取图片验证预测效果，如下图，准确率非常高，达99%以上。

NO.0 99.94% Dog



NO.1 99.95% Dog



NO.2 100.00% Cat



NO.3 100.00% Cat



NO.4 99.92% Dog



NO.5 100.00% Cat



NO.6 100.00% Dog



NO.7 100.00% Cat



NO.8 100.00% Cat



NO.9 100.00% Cat



NO.10 100.00% Cat



NO.11 100.00% Cat



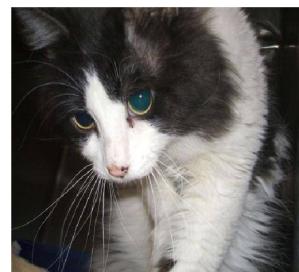
NO.12 100.00% Dog



NO.13 100.00% Cat



NO.14 100.00% Cat



NO.15 99.94% Dog



5. 结论与思考

本项目开始选用的经过Imagenet训练过的模型去预测测试集，经过开放权重后，Xception模型就已经达到了项目要求。让我惊讶的是，综合多个预训练模型的训练结果，可以明显提高预测得分。并且，保存训练好的模型的特征向量，然后在融合模型中载入，无需重复训练，很快就能得到训练结果，这样既提高了训练得分，还能节省大量训练时间。因此，使用迁移-融合的方法能够很好地应用于图像分类问题。

6. 参考文献

[1] : 李飞飞：如何教计算机理解图片.

http://open.163.com/movie/2015/3/Q/R/MAKN9A24M_MAKN9QAQR.html

[2] : <https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition>

[3] : Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[4] : https://en.wikipedia.org/wiki/Transfer_learning

[5] : Chollet, François. "Xception: Deep learning with depthwise separable convolutions." arXiv preprint (2017): 1610-02357.

[6] : Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.

[7] : Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks." In CVPR, vol. 1, no. 2, p. 3. 2017.

[8] : He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

[9] : <https://zhuanlan.zhihu.com/p/34068451>

[10] : https://github.com/ypwhs/dogs_vs_cats