机器学习工程师纳米学位

毕业项目报告-猫狗大战

杨振群

2018/10/14

I. 问题的定义

问题陈述

猫狗大战是一个经典的kaggle竞赛项目，项目目标是训练一个模型在给定的图像中对猫狗进行分类，这是一个有监督的图像分类问题，典型的计算机视觉问题。计算机视觉是当今最热门的研究领域之一，20世纪70年代后期随着机器计算能力的增强而不断发展起来，计算机视觉是能够对图像中的客观对象构建明确而有意义的描述，这个研究领域已经衍生出一大批快速成长的、有实际作用的应用，例如人脸识别、图像检索、自动驾驶等。计算机视觉的关键技术包括图像分类、对象检测、目标追踪、语义分割、实例分割、关键点检测等，其中以图像分类这项技术最为基本，以imagenet数据集最为出名。Imagenet是一个庞大的标记图片数据集，约有1.4千万张图片，约2万多个分类，诸多著名卷积神经网络模型都使用这个数据集来进行测试。

项目使用的数据集是kaggle竞赛提供的数据（https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data），原始数据来自于微软的一个名为ASSIRA的项目（Animal Species Image Recognition for Restricting Access），ASIRRA用猫狗图片分类任务来作为captcha（Completely Automated Public Turing Test to Tell Computers and Humans Apart），图片识别任务对人类来说很简单但在当时对于机器来说很困难，微软和PEtfinder.com平台共同合作，三百多万张猫狗图片被数千个动物收容所的工作人员进行手工分类，kaggle竞赛所用的数据集正是这数据集中的一部分。训练集包括12500张被标记为猫的图片和12500张被标记为狗的图片，测试集包括12500张未标记图片。

Submissions are scored on the log loss:

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

where

- n is the number of images in the test set
- $\hat{y}_i$ is the predicted probability of the image being a dog
- y_i is 1 if the image is a dog, 0 if cat
- $log()$ is the natural (base e) logarithm

训练集的图片用来训练分类器模型，模型输出为给定一张图片预测出图像中是狗的概率。测试集的图片用来评价模型，模型评价指标是对数损失，即测试集样本正确标签的预测概率对数损失的平均值，数学表现形式如下所示。对于单个样本来说，对数损失是正确标签的预测概率的自然对数取负，正确分类标签给出的预测概率为100%的时候对数损失为零，预测概率越低对数损失越大。对数损失相对于正确率来说，是对模型分类能力更加准确的刻画，不仅仅需要模型分类正确，还需要模型对自己分类结果十分有把握。

项目希望得到的结果就是训练得到一个对数损失尽量小的分类器模型，项目预备作为对标的基准结果为该kaggle的leaderboard（https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/leaderboard），目前对数损失第1名得分为0.0033，第50名得分为0.04842。

方案设计

有监督图像分类问题的一般解决方案是通过深度神经网络来进行特征提取，再将提取出来的特征值用于训练分类器模型。

项目计划分成以下几个阶段来开展，使用aws的deep learning ami作为基础环境，使用jupyter notebook作为编程环境，使用keras作为编程基础，使用tableau作为可视化工具，使用github来作为代码文件管理系统（https://github.com/danielyang123321/DogCat_Capstone_Daniel），项目中所列示的文件都存储在github中。

1. 数据预处理

使用预训练模型对训练集的图片进行标签预测，对标签存疑的样本进行剔除，从而提高训练集的数据质量。将图像统一成同样大小的尺寸，使用预训练深度神经网络模型对训练子集和验证子集的归一化图片张量数据进行计算得到特征数据，将样本特征数据和样本的标签按30%的比例分割为训练子集和验证子集。

2.分类器模型搭建

搭建的分类器模型由归一化层、全连接层以及激活层组成，其中归一化层的作用是为了防止模型过拟合，对分类器模型进行编译，损失指标采用categorical_crossentropy, 优化程序采用rmsprop,额外观察的评价指标为 accuracy。

3.分类器模型训练

预备使用keras自带的fit函数以及训练子集的数据对分类器模型进行训练，使用ModelCheckpoint来追踪训练过程，batch size选择为整个训练集，epoch初步选择为100个回合，在五个回合没有提高之后停止，将损失指标最小的模型作为最终的分类器模型。

4.分类器模型评估以及调优

模型的评估将在验证子集上开展，评价指标为对数损失，验证集表现良好之后，在测试集上面进行开展。预备提高分类表现的手段包括：对训练子集的图片进行增强从而提高旋转不变性和平移不变形，从只使用一个神经网路提取的特征扩展到使用多个神经网络提取的特征。

5.可视化

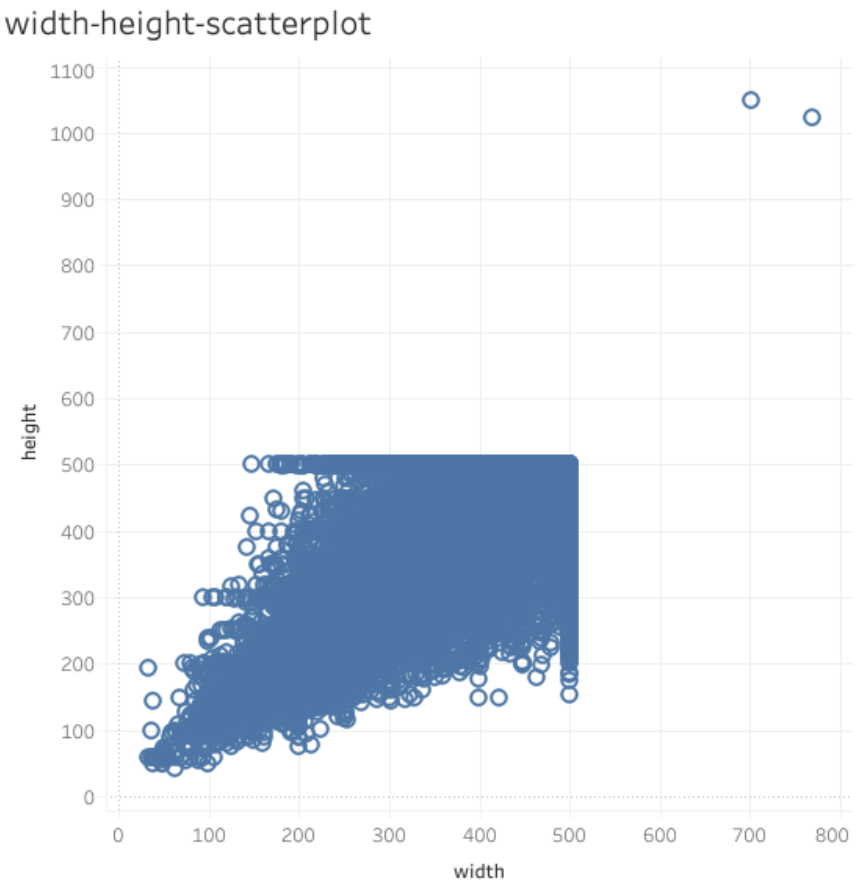
项目的可视化主要是模型训练过程中损失指标在训练集和验证集上的变化趋势。

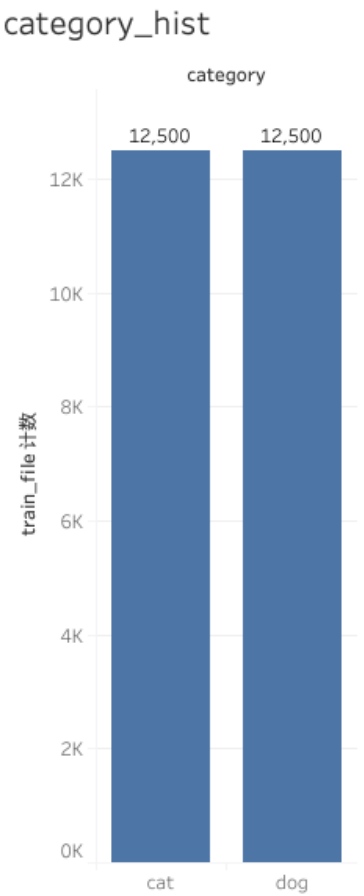
II. 分析

数据的探索

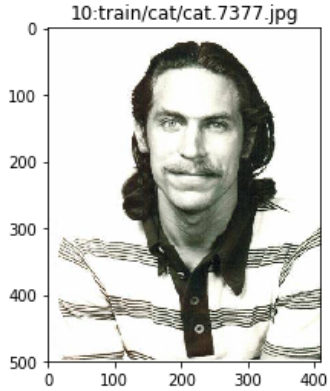
对训练集的图片进行了一些预览，可以发现猫狗在图片中的姿态以及拍摄角度非常多样，背拍、侧拍、俯拍的情况都存在，猫狗在图片的干扰也非常多样，铁丝网遮挡、衣物装饰、人和狗多主体、狗和狗多主体的情况都存在。

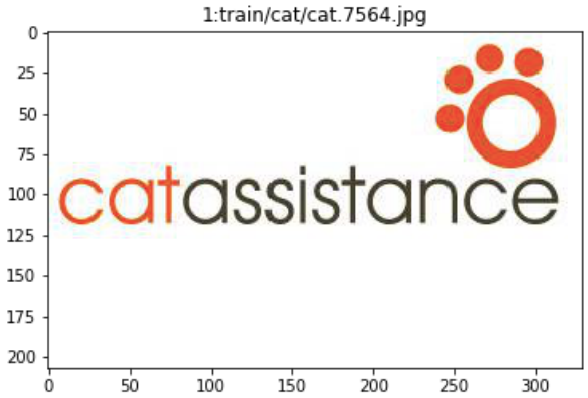
使用了jupyter notebook（DogCat_Capstone_01.Explore Dataset.ipynb）对训练数据集进行了图片维度提取，并观察了宽度、长度和深度的分布情况，图片深度一致为3，宽度和长度平均为360*400，存在分辨率较小和分辨率较高的两种极端情况。使用了tableau public对提取的训练集维度信息（train_dimension.csv）进行了可视化（https://public.tableau.com/profile/danielyang#!/vizhome/CatDogTableau/visualization），下面是样本类别分布直方图和样本长宽分布的散点图。

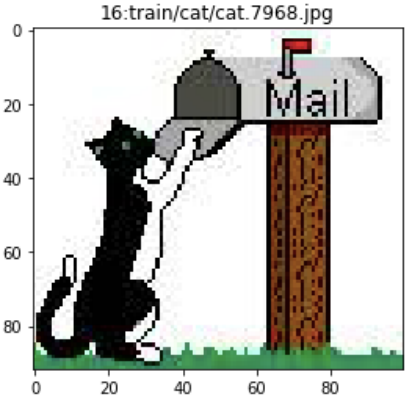


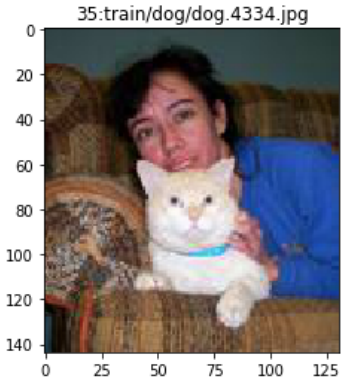


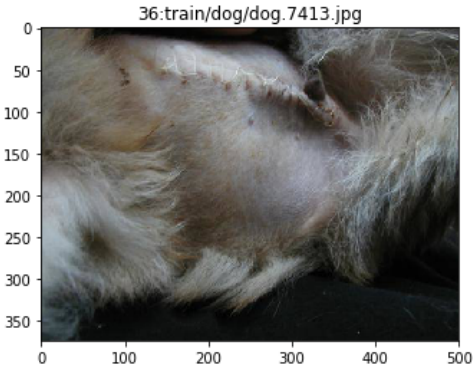
使用了jupyter notebook（DogCat_Capstone_01.Explore Dataset.ipynb），通过预训练模型naslarge对训练数据集上进行了Top30的标签预测，其中识别出来了标签存疑的三种情况：”既不是猫也不是狗”的样本有54个，“预测为猫标记为狗”的样本有14个，“预测为狗标记为猫”的样本有483个。手工进行了一一复核，预备剔除的样本共计110个（Samples2Exclude.xlsx）。

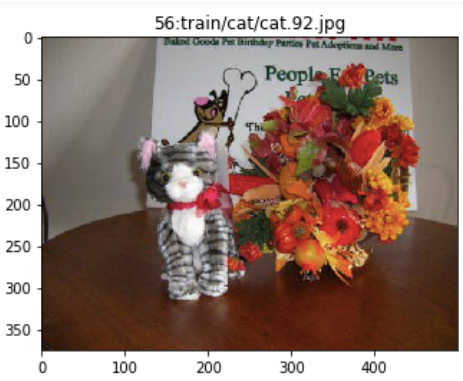












算法和技术

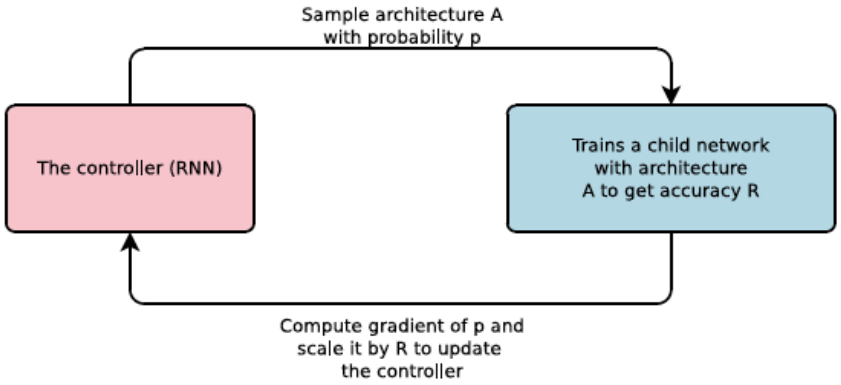
项目使用到的技术是用于提取特征的预训练卷积神经网络和用于对提取的特征数据进行分类的全联接神经网络，整个方案相当于迁移学习（参考1）。预备使用的预训练卷积神经网络将是在图像分类领域表现良好的模型，与此项目的领域相似性是迁移学习的基础，这种预训练网络的前端神经元提取的特征对于图像分类问题来说是比较通用的。

keras中提供支持了很多预训练模型（https://keras.io/applications/），其中包括2014年的VGGnet（参考2），2015年的resnet（参考3），2015年的inception（参考4），2016年的xception（参考5），2016年的inception-resnet（参考6）,2016年的densenet（参考7），2017年的nasnet（参考8），2017年的mobilenet（参考9）。

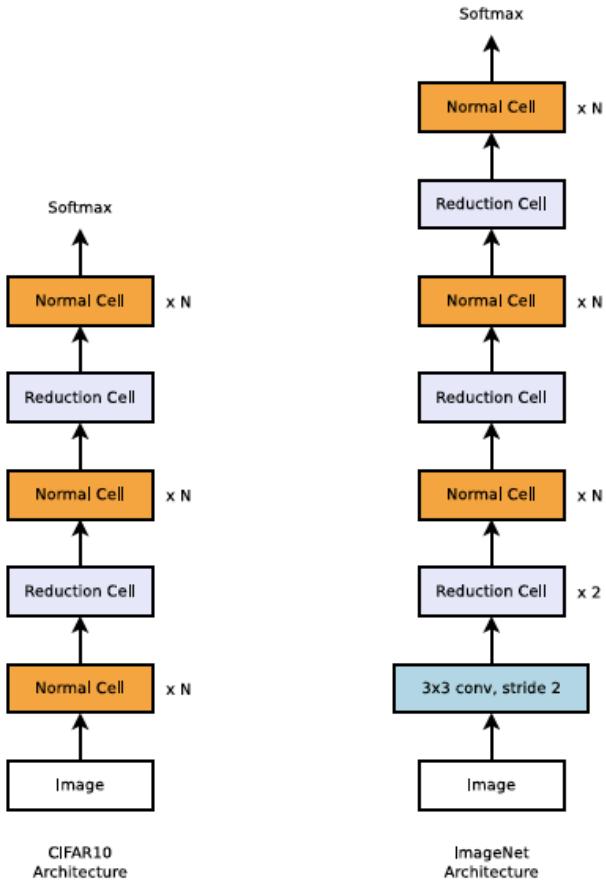
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
InceptionV3	92 MB	0.779	0.937	23,851,784	159
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
ResNet50	99 MB	0.749	0.921	25,636,712	168
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
Xception	88 MB	0.790	0.945	22,910,480	126

从上图中的准确率可以看出，表现最好的预训练模型是NASNetLarge，在imagenet数据集上面的第一预测准确率为82.5%，前五预测准确率为96%，目前图像分类领域的最佳模型。这个网络延续了NAS-Neural Architecture Search With Reinforcement Learning这篇论文（参考10）的核心机制，通过强化学习自动产生网络结构（inception网络结构设计中也有通过训练来寻找最佳卷积核的思想）；借鉴了Inception的模块堆叠结构减少了网络结构优化的搜索空间，大型网络直接由大量的同构模块堆叠而成，学习效率得到了提高；在CIFAR-10小型数据集上进行了神经网络架构搜索，并将搜索得到的最好的架构迁移到ImageNet图像分类和COCO物体检测上。

NASnet利用RNN控制器去预测一个子网络结构，接着训练这个子网络直到收敛，去验证集上测试得到一个准确率R，将这个R作为回报信号反馈给RNN控制器去更新RNN控制器的参数，从而产生更好的子网络结构，如下图所示。

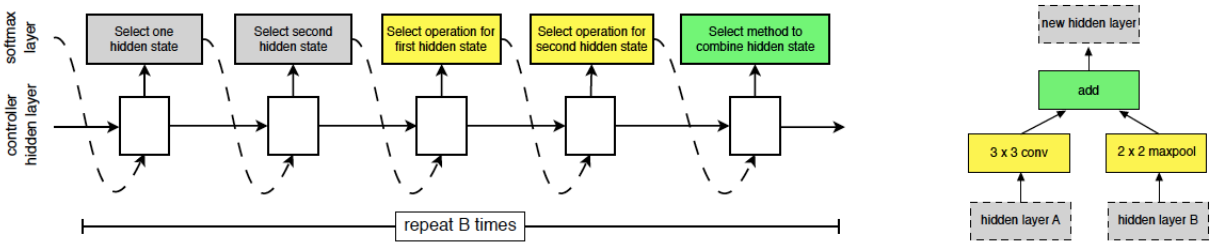


子网络结构的骨架是人为预先设计好的，借鉴了inception的模块重复堆叠的设计思想，由两种基本单元Normal Cell和Reduction Cell组成，设计好这两种基本单元的结构是rnn控制器的学习目标。其中Normal Cell是不改变输入feature map的大小的卷积单元，Reduction Cell是将输入feature map的长宽各减少为原来的一半的卷积单元。

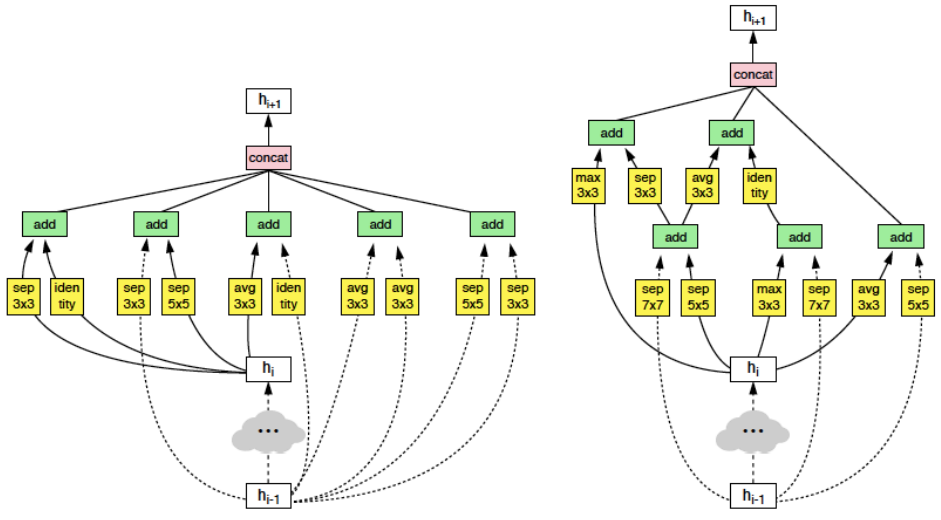


卷积单元的构建方法是选择两个hidden state以及对应的operation并选择一种方法生成一个新的hidden state，这种过程会被重复数次。其中operation也是人为预先设计的，包括如下：

- identity运算，是将特征不进行任何处理直接输出；
- 常规卷积运算共设计（3*3 convolution，1*7 then 7*1，1*3 then 3*1），用于提取特征，其中1*7 then 7*1, 1*3 then 3*1是先提取行特征再提取列特征，与直接进行7*7或者3*3相比，这种卷积预算的计算量相对少一些；
- Depthwise-seperable Convolution的运算共设计三种（3*3，5*5，7*7），这种运算中一个通道只被一个卷积核卷积，各个通道卷积计算完成之后才会在深度方向上进行加权组合，与常规卷积预算相比，参数数量和运算量较低，xception比inception的进步之处也在于利用了这种卷积预算；
- Pooling的运算共设计了四种（3*3 average，3*3 max，5*5 max，7*7 max），这种预算是对特征进行长宽的降维操作；
- 1*1 卷积预算，可以用来对特征进行深度上的维度调整。



nasnet学习得到的最好卷积单元如下所示，可以发现其发展出来了跟resnet相似的跳跃结构，ResNet就是通过这种残差结构（新层的输出是前层的输出加上这一层计算出来的残差值）来达到缓解随着网络加深而出现的梯度消失问题。



Normal Cell *Reduction Cell*

预备将采用这个预训练模型来提取特征数据，对于每张图片将可以提取出来4032个特征值用于进一步分类。在keras中，这个预训练模型需要设置的参数如下：

- include_top：预备设置为false，不保留顶层的全连接网络。
- weights：预备设置成'imagenet'，代表加载预训练权重。
- pooling：预备设置为max，代表了采用全局最大值池化方式。
- input_shape：预备采用模型默认的（331，331，3）.

基准结果

预备使用kaggle这个项目竞赛的leaderboard最佳分数作为基准阈值（https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/leaderboard）。目前猫狗大战在kaggle leaderboard上面的对数损失第1名得分为0.0033，第50名得分为0.04842，我们将以进入前50名作为这个项目的基准阈值，即对数损失小于0.04842。

III. 方法

数据预处理

使用了jupyter notebook（DogCat_Capstone_02.Preprocess Dataset），利用在数据探索阶段的成果剔除了110个样本，最终选取的样本共计24890个。

翻阅了nasnet的论文（参考8），论文中没有对图像进行了类似resnet中减去平均值那样的额外处理；翻阅了nasnet在keras中的实现（https://github.com/keras-team/keras-applications/blob/master/keras_applications/nasnet.py），其中提供了preprocess_input的函数对rgb值表示的图像数据进行预处理，这种预处理是将图像数据归一化到[-1,1]。使用了jupyter notebook（DogCat_Capstone_02.Preprocess Dataset），将图片按照统一的大小转换成张量数据（331，331，3），并使用了tensorflow中的preprocess_input将其归一化到[-1,1]。

```
#convert image to input data for model
def getdatafromimage(img_path):
    img = image.load_img(img_path, target_size=(331, 331))
    x = image.img_to_array(img)
    tensor=np.expand_dims(x, axis=0)
    imagedata=preprocess_input(tensor) #normalize image data using tf's preprocess_input to [-1,1]
    return imagedata
```

使用了jupyter notebook（DogCat_Capstone_02.Preprocess Dataset），使用预先训练模型nasnetlarge对训练集和测试的图片数据进行特征提取，预训练模型去除掉顶端神经元，网络权重采用imagenet预训练参数，pooling方式选择为max，输入形状采用默认的331x331x3。

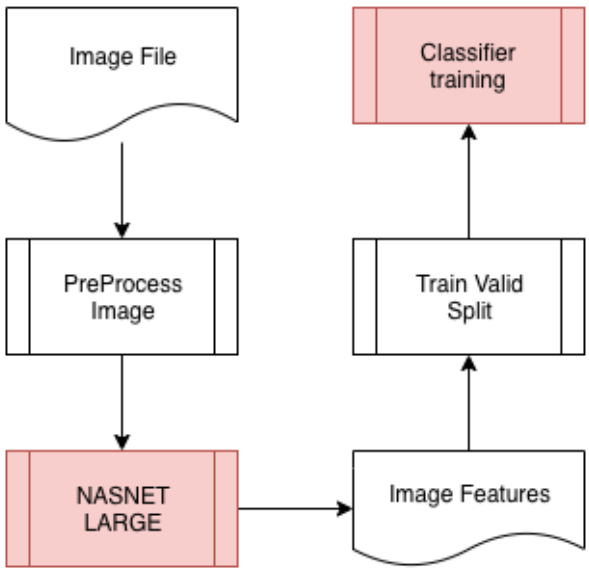
```
#extract features
pretrained_model=NASNetLarge(include_top=False, weights='imagenet',pooling='max',input_shape = (331, 331, 3))

def extract_features(train_file):
    imagedata=getdatafromimage(train_file)
    features=pretrained_model.predict(imagedata)
    return features.tolist()[0]
```

每个样本可以提取出来4032个特征值，提取出来的特征数据以及标签数据存储在train_features_final.csv和train_targets_final.csv, test_features.csv，这些特征数据文件较大。

执行过程

在预处理过程中已经使用了nasnetlarge这个预训练模型作为backbone，预训练模型的所有重要参数已经在预处理过程中详细列示出来。针对提取出来了训练集和测试集图片的特征数据，以下执行过程只是对特征数据进行分类器训练，其中分类器模型采用的是神经网络。



使用了jupyter notebook（DogCat_Capstone_03.Model Training），将预处理过程中的数据文件（train_features_final.csv和train_targets_final.csv）进行加载，利用scikitlearn对于提取出来的特征数据train_features_final和标签数据进train_targets_final行了训练集和验证集的分割，分割比例设置为30%，分割完之后生成了训练子集的特征数据和标签数据（X_train，y_train）以及验证子集的特征数据和标签数据（X_valid，y_valid）。

```
#read features and label data from csv file
import pandas as pd
train_features_final_df = pd.DataFrame(pd.read_csv("train_features_final.csv")).drop('Unnamed: 0', axis=1)
train_targets_final_df = pd.DataFrame(pd.read_csv("train_targets_final.csv")).drop('Unnamed: 0', axis=1)

#Split training dataset into training subset and testing subset
from sklearn.model_selection import train_test_split
X_train,X_valid,y_train,y_valid= train_test_split(train_features_final_df,train_targets_final_df,test_size=0.3,r
```

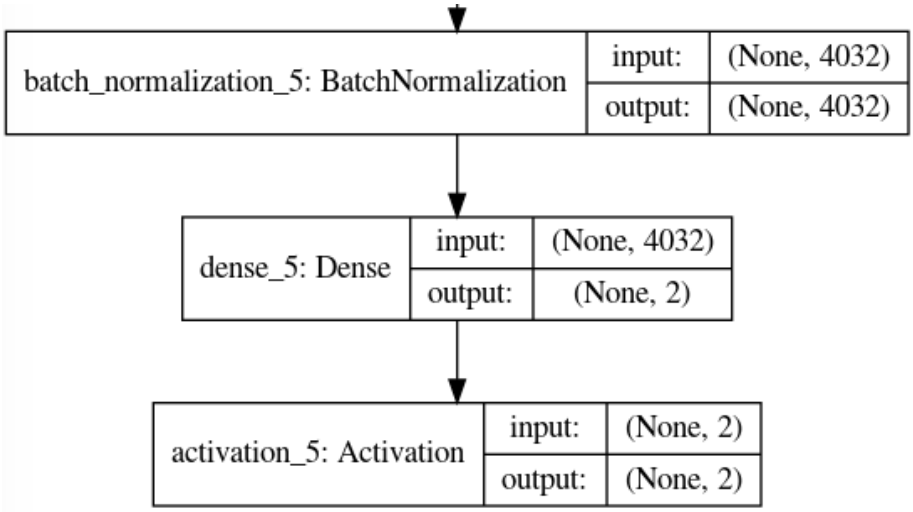
使用了jupyter notebook（DogCat_Capstone_03.Model Training），使用特征数据的维度-4032作为输入，以标签数据的维度-2作为输出，搭建了一个正则化层（防止过拟合）、一个全连接层、一个激活层，参数共计24,194个，其中可训练参数共计16,130个。对分类器模型进行编译，损失指标采用categorical_crossentropy, 优化程序采用rmsprop,额外观察的评价指标为accuracy。

```
#Model Building & Compling
from keras.models import Sequential
from keras.layers import Dense, Activation, BatchNormalization
from keras.utils.vis_utils import plot_model

DogCat_model = Sequential()
DogCat_model.add(BatchNormalization(input_shape=X_train.shape[1:]))
DogCat_model.add(Dense(2))
DogCat_model.add(Activation('softmax'))
DogCat_model.summary()

DogCat_model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
plot_model(DogCat_model,to_file='model_v0.png', show_shapes=True, show_layer_names=True)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_10 (Batc (None, 4032)              16128
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 8066
_________________________________________________________________
activation_8 (Activation)    (None, 2)                 0
=================================================================
Total params: 24,194
Trainable params: 16,130
Non-trainable params: 8,064
_________________________________________________________________
```



使用了jupyter notebook（DogCat_Capstone_03.Model Training），使用keras自带的fit函数以及训练子集的数据对分类器模型进行训练，使用ModelCheckpoint来追踪训练过程，因为环境内存资源充足所以将batch size设置为训练子集所有样本，epoch设置为100个回合，earlstopping设置为5个回合没有提高即可停止，将损失指标最小的模型作为最终的分类器模型。

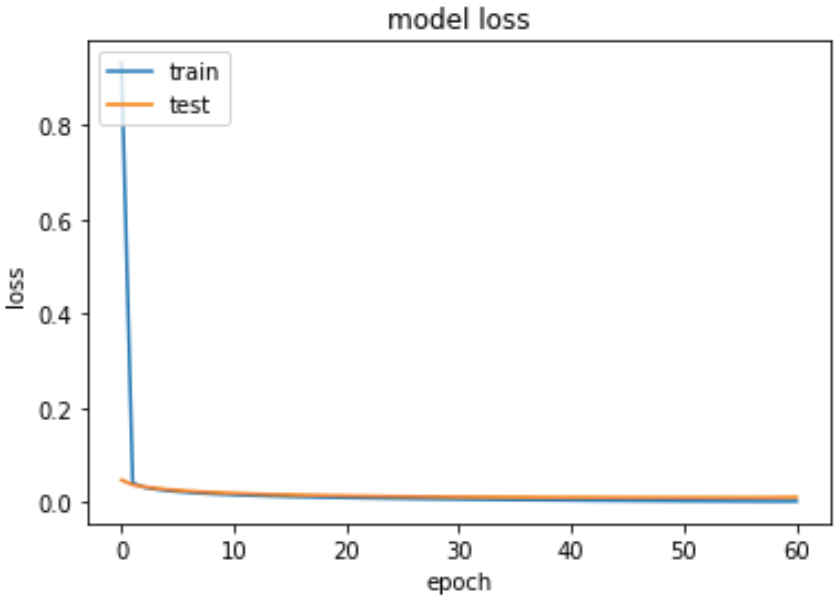
```
#Model Training

from keras.callbacks import ModelCheckpoint
checkpointer = ModelCheckpoint(filepath='model.v0.weights.best.hdf5',
                               verbose=1, save_best_only=True)

from keras.callbacks import EarlyStopping
earlystopping = EarlyStopping(monitor='val_loss', patience=5, verbose=2)

history=DogCat_model.fit(X_train, y_train,
          validation_data=(X_valid, y_valid),
          epochs=100, batch_size=X_train.shape[0], callbacks=[checkpointer,earlystopping], verbose=1)
```

训练在61个回合之后停止，在第56个回合时得到了验证集最佳损失指标，验证集最佳损失指标是0.00982，训练集损失指标是0.0020，训练集和验证集之间的损失指标相差不大，模型不存在过拟合的情况，应该具备较好的泛化能力。



使用了jupyter notebook（DogCat_Capstone_03.Model Training），将具有最小损失指标的分类模型应用在预处理过程中提取出来的测试特征数据（test_features.csv）上，并将提取出来的预测结果数据（submission_v0.csv）提交至kaggle上，kaggle给出的损失指标为0.06117。

Name	Submitted	Wait time	Execution time	Score
submission_v0.csv	a few seconds ago	0 seconds	0 seconds	0.06117

完善过程

在执行过程中，观察了预测结果数据，发现模型的输出比较武断，给出的概率大部分都是1或者0，这种预测概率在预测错误时候带来的对数损失会比较大，因此考虑在分类模型之后增加一个预测概率修正函数predict_adjust，将非常武断的预测概率按照0.005进行修剪。

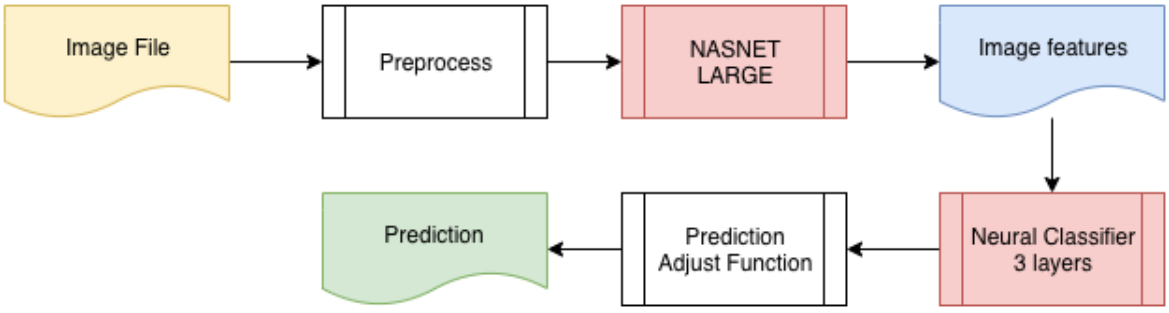
```
#function to reduce absolute prediction
def predict_adjust(x):
    epsilon=0.005
    if x>1-epsilon:
        return 1-epsilon
    if x<epsilon:
        return epsilon
    return x
```

使用了jupyter notebook（DogCat_Capstone_03.Model Training），将提取出来的调整预测结果数据提交至kaggle上，kaggle给出的损失指标为0.03940，这样的对数损失在kaggle public leaderboard中可以排到前15名，达到了项目的基准阈值（0.04842）。

Name	Submitted	Wait time	Execution time	Score
submission_v1.csv	a few seconds ago	0 seconds	1 seconds	0.03940

IV. 结果

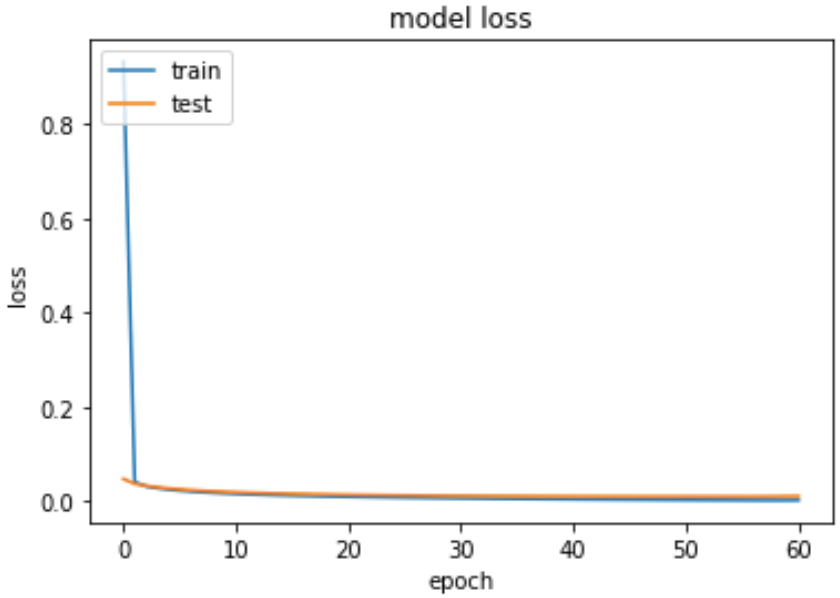
改善后的模型是由预训练的naslarge卷积神经网络、三层神经网络分类器、预测概率调整函数组成，整个模型的组成非常简洁，其中预训练的naslarge卷积神经网络是核心部件，承担了图像特征数据提取的重任。模型的输出跟期待的结果一致，验证集最佳损失指标是0.00982，训练集损失指标是0.0020，训练集和验证集之间的损失指标相差不大，模型不存在过拟合的情况，具备较好的泛化能力，模型在测试集上的损失指标达到了0.03940，达到了项目的基准阈值。可以推断，该模型可以很稳定地解决猫狗图像分类问题。



V. 项目结论

结果可视化

训练在61个回合之后停止，在第56个回合时得到了验证集最佳损失指标，验证集最佳损失指标是0.00982，训练集损失指标是0.0020，训练集和验证集之间的损失指标相差不大，模型不存在过拟合的情况，具备较好的泛化能力。



对项目的思考

项目比预期的要进行得顺利很多，阅读了这个项目的参考论文，发现前辈们在探索解决方案的时候需要尝试很多不同的优化方法，而在这个项目中我几乎是一次尝试即可达到目的。我认为这是得益于预训练卷积神经网络的进化，这次做项目的时候keras已经开始支持naslarge这个预训练神经网络模型，这个cnn在提取图像特征上的能力相对于之前的模型来说得到了提高，高质量的特征数据为分类模型的训练打下了很好的基础。这个项目给我最大的感悟是，特征提取的质量对于图像分类任务来说是极其关键的。

可以作出的改进

预备提高分类模型表现的手段在项目中并没有使用到，但这些改进可能可以进一步提高模型的分类能力，这些手段包括：对训练子集的图片进行增强从而提高旋转不变性和平移不变形，从只使用一个神经网路提取的特征扩展到使用多个神经网络提取的特征。

项目参考

1. Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson (2014). How transferable are features in deep neural networks, *ARXIV*, https://arxiv.org/pdf/1411.1792.pdf

2. Karen Simonyan, Andrew Zisserman (2014). very deep convolutional networks for large-scale image recognition, *ARXIV*, https://arxiv.org/pdf/1409.1556.pdf

3. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). Deep Residual Learning for Image Recognition, *ARXIV*, https://arxiv.org/pdf/1512.03385v1.pdf

4. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna (2015). Rethinking the Inception Architecture for Computer Vision, *ARXIV*, https://arxiv.org/abs/1512.00567

5. François Chollet (2016). Xception: Deep Learning with Depthwise Separable Convolutions, *ARXIV*, https://arxiv.org/abs/1610.02357

6. Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, *ARXIV*, https://arxiv.org/abs/1602.07261

7. Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger (2016). Densely Connected Convolutional Networks, *ARXIV,* https://arxiv.org/abs/1608.06993

8. Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le (2017). Learning Transferable Architectures for Scalable Image Recognition, *ARXIV*, https://arxiv.org/abs/1707.07012

9. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, *ARXIV*, https://arxiv.org/abs/1704.04861

10. Barret Zoph, Quoc V. Le (2016). Neural Architecture Search with Reinforcement Learning, *ARXIV*, https://arxiv.org/abs/1611.01578