机器学习毕业项目论文

猫狗大战

I. 问题的定义

项目背景

近年来，随着大数据的飞速发展，人们获取海量数据变得越来越容易。再加上，出现了 GPU 等高性能并行计算显卡，大幅提升了计算能力，让深度学习算法发展迅速。深度学习，特别是卷积神经网络 CNN[b] 在图片分类领域取得了惊人的效果。本项目猫狗大战，识别一张图片是猫还是狗，主要使用迁移学习的方法迁移经典的图片分类算法模型，特别是 ImageNet 图片分类比赛 [a] 下方中的经典模型，如 VGG[c], ResNet[d], Xception[g], InceptionV3[e], InceptionResnetV2[f] 等。

问题描述

使用深度学习方法识别一张图片是猫还是狗。

- 输入：一张彩色图片

- 输出：是猫还是狗

由于图片数量有限，计算力也有限，因此采用迁移学习 ImageNet[a] 预训练模型的方式。 主要有以下几种方式，本项目将选择结果最好的作为最终模型。

1. Fine-tuning 单个经典模型
2. 模型融合（融合特征）

评估指标

选用 Kaggle 官方的评估标准 logLoss

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

- 其中，
- n：测试集的图片总数
- $\hat{y}_i$: 图片预测为狗的概率
- y_i：当图片为狗时，y_i=1，当图片为猫时，y_i=0
- log():base 为 e 的自然对数
- Logloss 越小越好。

II. 分析

数据的探索

使用 Kaggle 上 Dogs vs. Cats 的数据集. 此数据集可以从 kaggle 上下载。 Dogs vs. Cats

训练集包括大小不一带标注的猫狗图片各 12500 张，测试集包含不带标注图片猫狗共 12500 张。这些图片都是彩色的，大小不一，尺寸不一，包含不同品种的猫狗，在不同的背景下，远景，近景，还有遮挡。

对数据集做简单探索，发现有以下异常情况，需要作为异常值去掉，从而提高模型的训练效果。

- 有些图片尺寸特别小，十分模糊。
- 有些图片非猫非狗。
- 有些图片背景十分复杂。

探索可视化

猫狗中尺寸特别小，很模糊的图片

cat.4821.jpg:　1106 字节，　尺寸 60*39

cat.6614.jpg:　1310 字节，　尺寸 50*49

猫狗中非猫非狗的图片

cat.10712.jpg

dog.1773.jpg

背景太复杂的图

dog.6725.jpg

cat.3672.jpg

算法和技术

A. 迁移学习

在实际项目中，很少会直接从头训练一个卷积神经网络，因为大部分项目没有大量的标注训练集。一般情况下会从一个预训练卷积神经网络开始（比如 Imagenet 包含 1.2million 图片，1000 种类），将这个卷积网络作为基础模型，或者作为特征提取器进行迁移学习。

参考： https://cs231n.github.io/transfer-learning/#tf

本项目采用了两种方式的迁移学习。

1. 单模型 fine-tuning: 将 Imagenet 经典模型(InceptionV3, Xception, InceptionResnetV2)作为基础模型，去掉 top 层，加上 pooling 层，dropout 层，全连接层 sigmoid 二分类。第一次锁定基础模型所有层，训练剩下的权重，第二次放开基础模型的若干层，再次训练整个模型，从而学会分类猫狗。
2. 多模型特征融合：将 Imagenet 经典模型(InceptionV3, Xception, InceptionResnetV2)作为特征提取器，去掉 top 层，获取各个模型得到的特征向量。再将多个模型得到的特征向量横向拼接，作为输入，加上 pooling 层，dropout 层，全连接层 sigmoid 二分类。训练新构造的模型，从而学会分类猫狗。

迁移学习卷积神经网络之所有有效，是因为卷积神经网络的特点就是能够抽象图片的特征，并且能够共享权重。经过大量图片大量物种预训练的卷积神经网络已经将很多物种的特征抽象出来，比如很多物种都含有底层的特征：点，横线，竖线，不同角度的线等，更高一层的特征：圆形，弧形等。更高层的特征：眼睛，鼻子，耳朵等。猫狗图片属于同样有这些特征，因此使用 Imagenet 上预训练的模型来提取猫狗的特征或者在此基础模型上再训练效率更高，能更快得到很好的效果。

将多个经典 CNN 模型提取到的特征融合，能弥补单个模型可能没有学习到的边缘特征，使特征集合更全面，从而获得更好的分类效果，所以多个模型提取特征融合会比单个模型 fine-tuning 有更好的表现，这在后面的结果中得到证实。

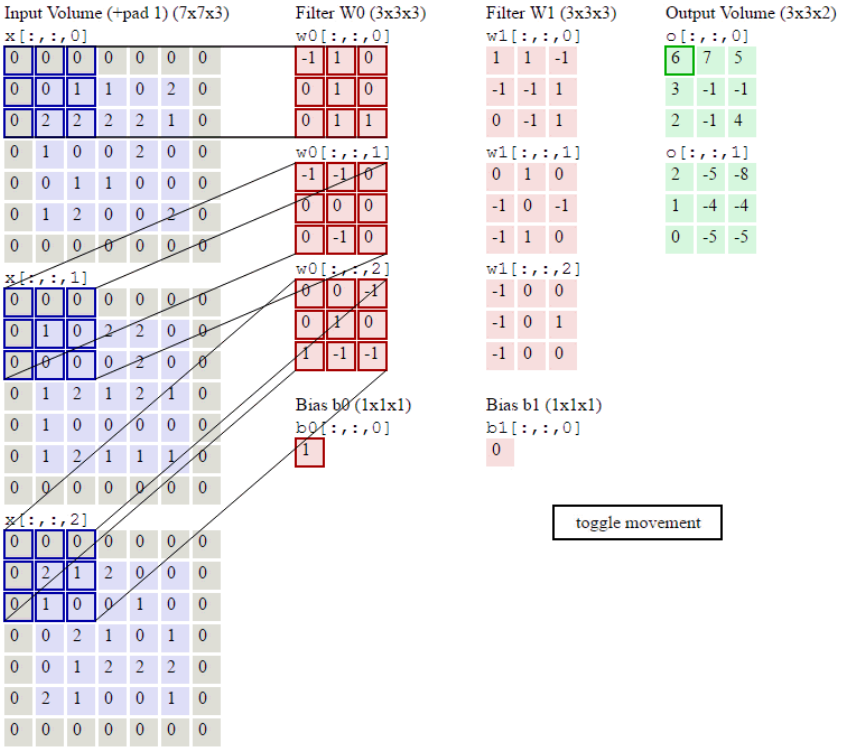
B. 模型中用到的重点知识

❖ 卷积神经网络

本项目用到的三个经典模型 InceptionV3, Xception, InceptionResnetV2， 都是由卷积神经网络 CNN 演化而来。

一个典型的卷积神经网络包括以下几个部分：输入层，卷积层，池化层，全连接层 Dropout 层，输出层。

❖ 卷积计算



卷积过程图，图片来源：https://www.jianshu.com/p/544e2354b30a

卷积是以大小为(Fw,Fh)比如上图中 3*3 这样的卷积核(filter)作为共享权重，以步长 Fs 比如 1 扫描输入层大小为（Iw, Ih, In）并作卷积操作，即输入层对应位置与卷积核对应位置相乘并相加得到对应位置的输出。其中 Fw 表示卷积核宽度，Fh 表示卷积核高度，Iw 表示输入层的宽度，Ih 表示输入层的高度，In 表示输入层的深度。

上面的卷积过程用数学公式表达出来就是:

$$s(i,j) = (X * W)(i,j) + b = \sum_{k=1}^{n\_in} (X_k * W_k)(i,j) + b$$

其中，n_in 为输入矩阵的深度，即改张量最后一维的维数。Xk 表示第 k 各输入矩阵。Wk 表示卷积核的第 k 个子卷积核矩阵。s(i,j)表示卷积核 W 对应的输出矩阵的对应位置元素的值。

假设输入层宽度， 高度和深度分别为 Iw*Ih*Id, 卷积核的宽度和高度和深度分别为 Fw*Fh*Fd, 其中必有 Id=Fd， 若 padding 记为 p, 步长记为 s,则输出层的宽度， 高度和深度分别为:

$$Ow = \frac{Iw - Fw + 2p}{s} + 1$$

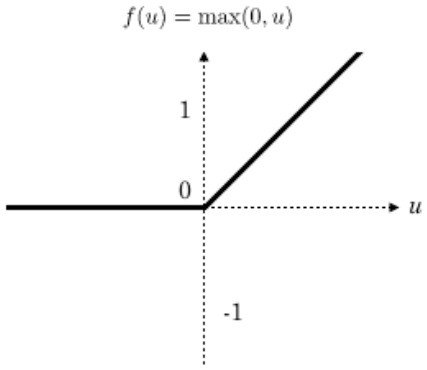
$$Oh = \frac{Ih - Fh + 2p}{s} + 1$$

Od = 卷积核的个数

卷积层的特点是可以共享权重，从而能更好的抽象图片的特征。

❖ Relu 激活函数

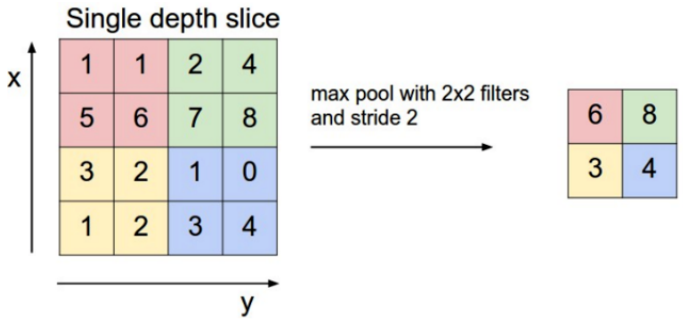
对于卷积之后的输出， 一般会用 Relu 作为激活函数， 将输入的张良中的小于 0 的位置对应的元素值都变为 0.



Relu 函数， 图片来源： https://www.jianshu.com/p/544e2354b30a

❖ 池化 Pooling

在一个或者多个卷积层之后通常会接 Pooling 层， 将图片进行压缩降维。最常用的是 MaxPooling， 如图所示， 一个 4*4 的图片经过 2*2 的 filter, stride=2 压缩后降为 2*2 的图片，其中 filter 扫描过 Input 上的对应位置取其中的最大值作为输出。

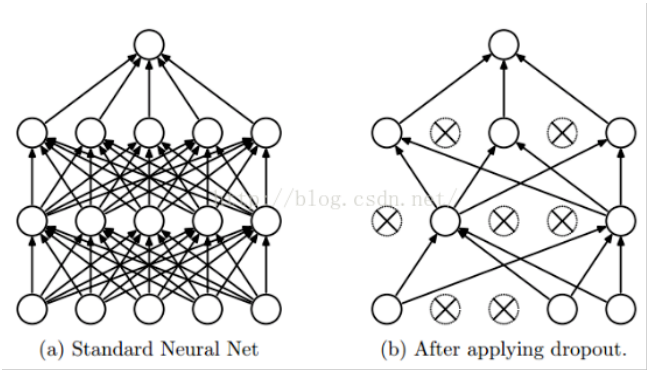


Maxpooling， 图片来源： https://www.jianshu.com/p/544e2354b30a

❖ Dropout 层

Dropout 层随机将一些神经元去掉，防止过拟合，Dropout 层在所选用的三个经典模型中都有使用。本项目由于猫狗数据集只有 25000 张图片，数量不多，为了避免过拟合，在经典模型之上也加了 dropout 层。

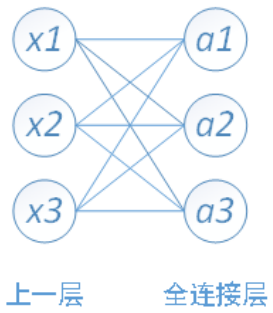
Dropout 层之所以能预防过拟合 *Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , 2012.*，是因为 dropout 层相当于训练了很多个只有部分单元的神经网络，每一个这样的部分网络，都可以给出一个分类结果，这些结果有些事正确的，有些是错误的，随着训练的进行，大部分网络都可以给出正确的分类结果，那么少数错误的分类结果就不会对最终结果造成大的影响。



Dropout 图片来源： https://www.jianshu.com/p/544e2354b30a

❖ 全连接层

在经典模型的最后通常需要加上一些全连接层，更进一步训练，全连接层的节点数通常以一定的速度下降，平稳过渡到最后一个全连接层用 Softmax 函数作多分类，或者用 sigmoid 函数作二分类。



全连接层，图片来源：https://www.jianshu.com/p/544e2354b30a

❖ GlobalAveragePooling 层

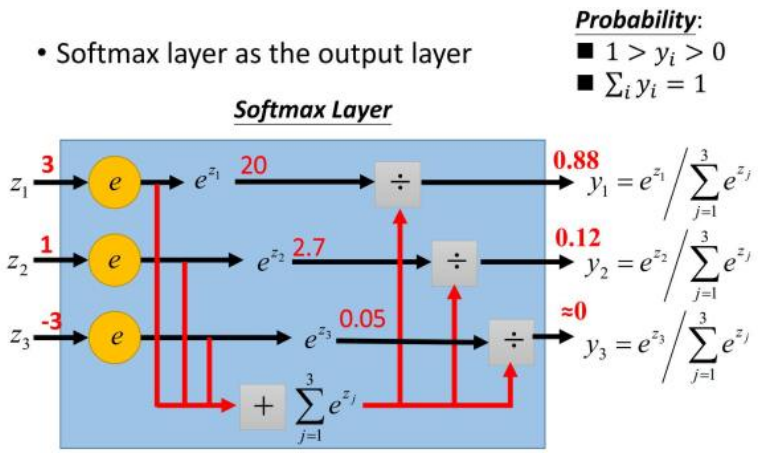
全连接层由于有大量的参数，容易导致过拟合，同时也会降低训练速度，所以本项目采用更有效的方法 GlobalAveragePooling，直接去掉全连接层，然后在最后一层，将卷积层设为与类别书目一直，然后全局 Pooling，从而输出类别个数结果。本项目猫狗二分类，最后分类层只有一个节点。

❖ Softmax 多分类

Imagenet 数据集有 1000 个分类，因此几个经典模型最有一个全连接层都是 softmax 作 1000 分类，公式为每种分类的概率

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{j=1}^{n} e^{z_j}}$$

其中，n 为类别数。



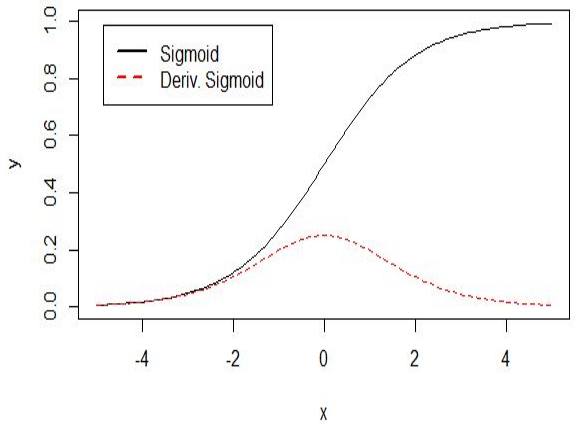
Softmax 多分类，图片来源：https://www.jianshu.com/p/544e2354b30a

❖ Sigmoid 二分类

本项目只需要分出猫狗两种类别，因此使用 sigmoid 函数作二分类。公式为

$$f(z) = 1/(1 + e^{-z})$$

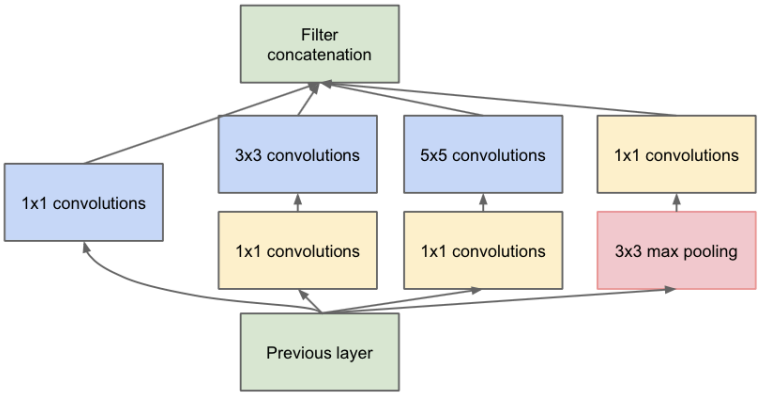
sigmoid 函数是以前也常作为激活函数，但是因为导数最大值为 0.25，很容易造成梯度消失从而无法完成训练，因此现在很少使用，但常作为二分类的激活函数。本项目是猫狗二分类，因此在经典模型去掉 top 层后最后的全连接层使用它作为二分类激活函数。



Sigmoid 函数及其导数 图片来源：https://www.jianshu.com/p/22d9720dbf1a

❖ Inception 模块

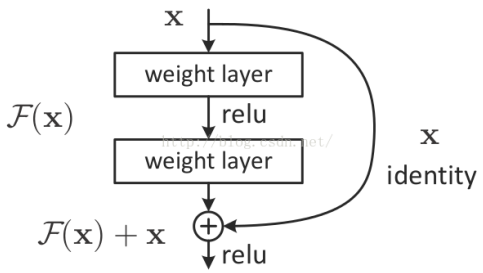
为了更进一步降低参数数量减少过拟合，同时又利用 GPU 针对密集矩阵计算的性能，引入了 Inception 模块，它由 1*1，3*3，5*5 多个不同的卷积核以及 maxpooling 堆叠而成，同时为了更进一步降低参数在这些层前后加上了 1*1 的卷积核层用于降维。Inception 模块的概念在本项目选用的经典模型 InceptionV3 和 InceptionResnetV2 模型中大量使用。



Inception 模块，图片来源：https://blog.csdn.net/stdcoutzyx/article/details/51052847

❖ Residual 模块

网络的深度是实现好的效果的重要因素。然而梯度弥散/爆炸成为训练深层次的网络的障碍，导致无法收敛。归一初始化，各层输入归一化，使得可以收敛的网络的深度提升网络的深度，但容易导致网络退化了，即增加网络层数却导致更大的误差。残差网络能很好的解决退化问题。

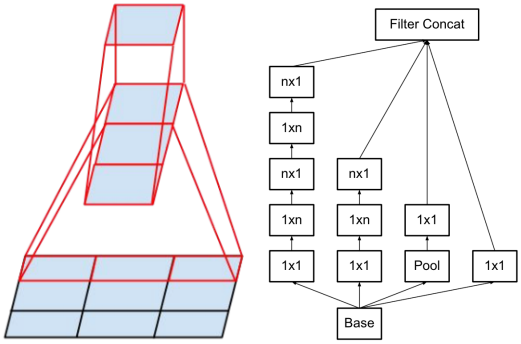


a. 残差模块，图片来源：*Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition arXiv:1512.03385 [cs.CV]*

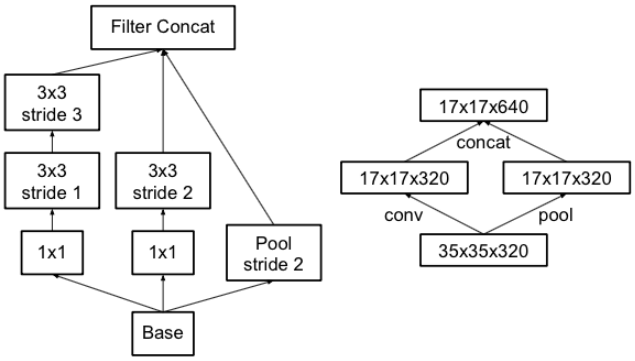
C. 本项目选用的经典 CNN 预训练模型介绍

❖ InceptionV3 模型

InceptionV3 在 Googlenet 的基础上，使用了更小的卷积核代替较大的卷积核，即分解，将 7*7 分解成 1*7 和 7*1， 3*3 分解成 1*3，3*1。这种方式既可以加速计算，有可以更进一步加深网络深度，增加网络的非线性。比如 1*3 和 3*1 两种来代替 3*3， 这种方式能节省 33%的计算量。



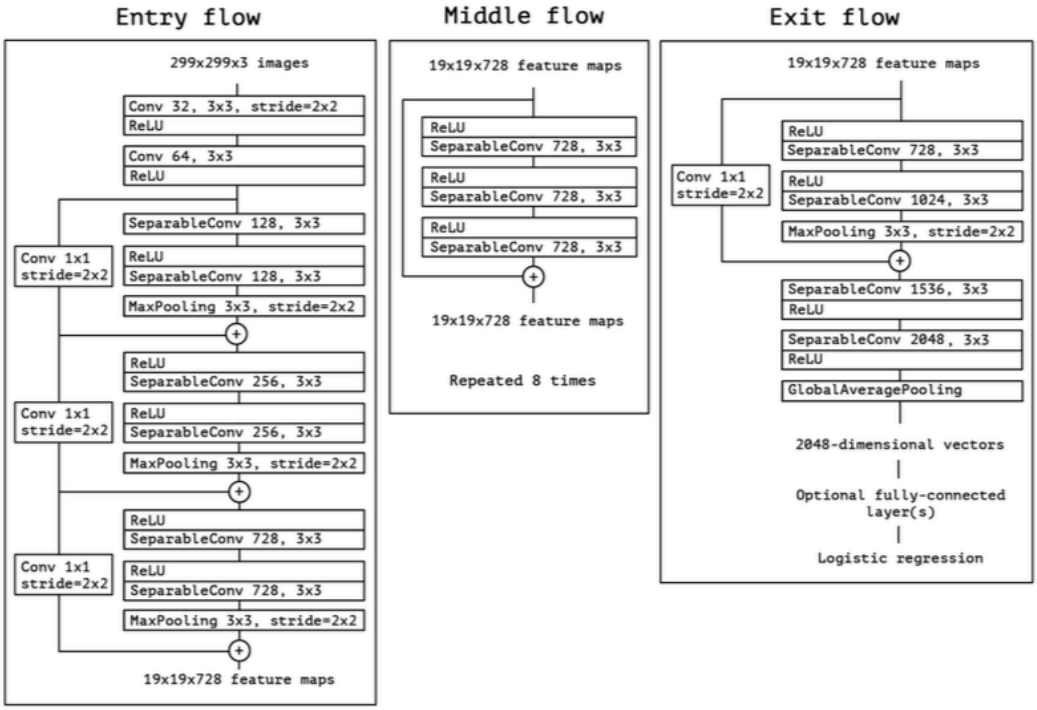
使用两个并行的模块可以进一步降低计算量



图片来源： https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf

❖ Xception 模型

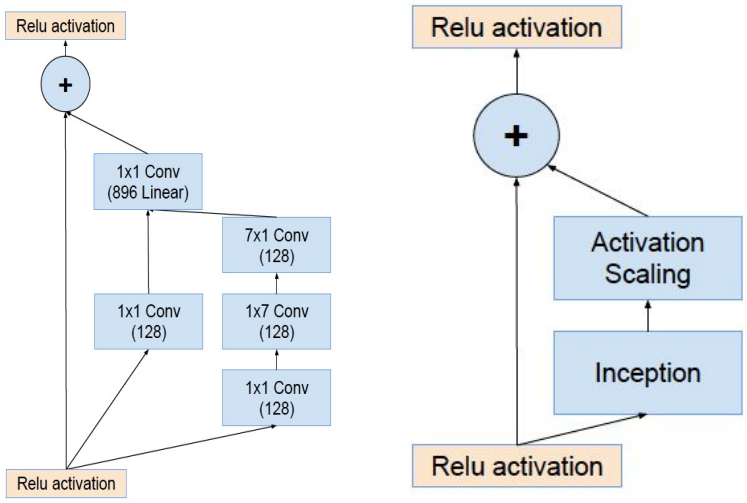
Xception 在 InceptionV3 的基础上引入了 depthwise separable convolution，在基本不增加网络复杂度的前提下提高了模型的效果。 Xception 的目的不在于模型压缩，而是提高性能。



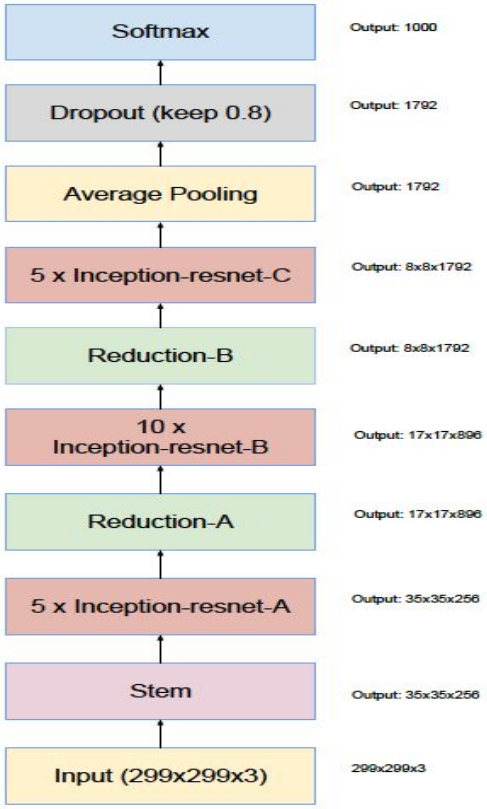
Xception 结构图，图片来源： https://arxiv.org/pdf/1610.02357.pdf

❖ InceptionResnetV2 模型

Inception 和 Resnet 相结合能加快模型训练的收敛，但是网络训练不稳定，于是引入了 scale，scale 不会降低网络精度，而会使网络更稳定。



InceptionV3 结构太过于复杂，在此基础上得到更优化的 InceptionV4, InceptionV4 与 Resnet 以上改进的 residual 模块相结合即为 InceptionResnetV2，是所选三个经典模型中精度最高的模型。



图片来源 Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning arXiv:1602.07261 [cs.CV]

基准模型

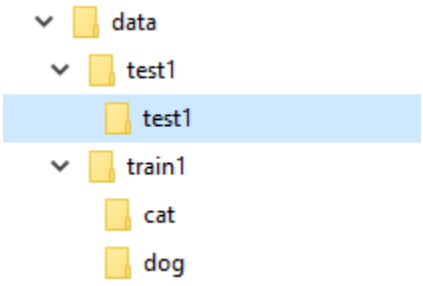
Kaggle 猫狗大战项目 10%作为基准, 即 logloss 要低于基准阈值 0.06127.

III. 方法

数据预处理

1. 将压缩文件解压并按照设计好的目录放好

- ❖ 训练集：解压，分别将文件名含猫狗的放进对应子文件夹，执行两次，其中 train2 中的数据用来去除异常值
- ❖ 测试集：解压，将所有文件放入一个子文件夹（方便使用 ImageGenerator）
- ❖ 处理后的文件夹结果如下



2. 清除异常值（参考：https://zhuanlan.zhihu.com/p/34068451）

- 使用 ImageNet 的预训练模型 InceptionV3 直接对猫狗训练图片分别进行预测
- 获取 ImageNet 猫狗分类，狗：118 种，猫：7 种。参考：https://blog.csdn.net/zhangjunbob/article/details/53258524
- 选取 TopN=10， 打印 TopN 预测结果中没有包含 ImageNet 的 118 种狗或者 7 种猫的图片
- 肉眼观察会发现有一些非猫非狗，背景复杂的异常值被找出来了，同时也有很多肉眼可以看出是猫狗的图片也被找出来了
- 分别逐步调大 TopN 值，观察打印出来的异常值，直到绝大部分异常值被找出来了，而正确的猫狗不会被打印出来。
- 由于 ImageNet 中狗种类较多较全而猫的种类较少，实际调参后对于 TopN=40 比较合适，对于猫 TopN=150 比较合适。具体的猫狗异常值见结果可视化部分。
- 确定了异常值图片后，将这些异常图片移除。

3. 数据预处理采用各个模型各自的 Preprocess_input()方法处理适合各自的模型的数据以适应各自模型（参考：https://keras.io/applications/）

执行过程

- 选取基础模型

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

参考：https://keras.io/applications/

参考 keras 中集成的 Imagenetb 的经典模型的表现，从模型 size, Top-1 Accuracy, Top-5 Accuracy 综合考虑，选取 InceptionResNetV2g, Xceptionh 和 InceptionV3f 三个模型作为基础模型。

- 单模型 fine-tuning

  - 创建 base_model,如 InceptionV3, 使用 imagenet 已经训练好的权重，去掉 top 层
  - 加上 AveragePooling 层
  - 加上 dropout 层
  - 输出层 activation="sigmoid"，做二分类
  - 锁定 base_model 所有层训练，optimizer="rmsprop"
  - 放开 base_model 部分层再训练一遍，optimizer=SGD

- 各模型分别在未去除异常值的训练集和已去除异常值的模型上分别执行一遍以观察差别
- 参考: https://keras.io/applications/

- 模型融合（融合特征）

  - 创建 base_model, 使用 imagenet 已经训练好的权重，去掉 top 层
  - 加上 averagepooling 层
  - 生成训练集特征向量
  - 循环以上步骤生成多个模型的训练集的特征向量
  - 生成测试集特征向量
  - 将训练集多个模型的特征向量融合，生成新的训练集
  - 构建简单模型，输入为训练集融合后的特征向量，加上 dropout 层，输出层（activation="sigmoid"）,作二分类
  - 训练模型，optimizer='adadelta'
  - 各种模型组合分别在未去除异常值和已去除异常值的训练集上分别执行一次，记录结果以观察差别
  - 参考: https://github.com/ypwhs/dogs_vs_cats

- 预测测试集

  - 用各个模型预测测试集的概率
  - 将概率值限制在[0.005, 0.995]区间内
  - 上传对应标号图片的概率值到 kaggle 获取分数

- 遇到的困难

  - 由于数据量大，很容易就爆内存，尽量使用 numpy.array 替代 List
  - Dtype 的选择，dtype 默认选择了 numpy.float64, 虽然精度很高，但是很容易爆内存
  - 输入数据 dtype 选择 numpy.uint8, 会导致训练无法收敛，需要使用 numpy,float32, 保证必要的精度又不至于爆内存

- 数据可视化时 dtype 选择 numpy.float32, 会导致图片显示失真，调整成 numpy.uint8
- ImageGenerator 需要用二级目录
- 保存预测结果到 csv 文件时要避免错位，测试数据的文件名和最终遍历的文件名顺序要保持一致。

完善

原则上各个模型和迁移学习后的表现排名应该与各个模型的表现排名，即 InceptionResnetV2[g] 比 Xception[h] 好，Xception[h] 比 InceptionV3[f] 好，而模型融合比单个模型表现更好。

按照以上原则在原始训练集上训练后各模型在测试集上表现 LogLoss 如下:

Training set	InceptionV3	Xception	InceptionResnetV2	merge features 3models
Training Set with outliers	0.04144	0.05112	0.03839	0.03643

其中表现最好的是融合三个模型特征向量的模型。

IV. 结果

模型的评价与验证

原则上各个模型在去除了异常值的训练集训练的效果比在没有去掉异常值的训练集训练的

Training set	InceptionV3	Xception	InceptionResnetV2	Merge features 3models
Training Set with outliers	0.04144	0.05112	0.03839	0.03643
Training Set without outliers	0.04295	0.05161	0.03838	0.03637

效果要好。下表为各模型在不同训练集训练下测试的 LogLoss.

去除了异常值后， InceptionV3， Xception 的 Logloss 反而稍有上升， InceptionResnetV2 和模型融合 Logloss 略微下降。这个现象我无法解释，看起来异常值对于这些模型影响不是很大，从另一个角度来说，模型对于异常值有一定的容忍能力。从结果综合来看，表现最好的为用去除异常值的数据训练的融合模型。

合理性分析

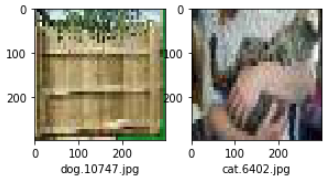
模型融合的表现远超 Kaggle10%，已达到预期。

Kaggle 10%	Merge_3_models(without outlier)
0.06127	0.03637

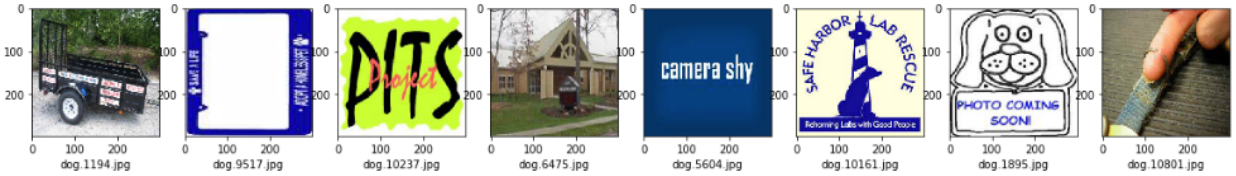
V. 项目结论

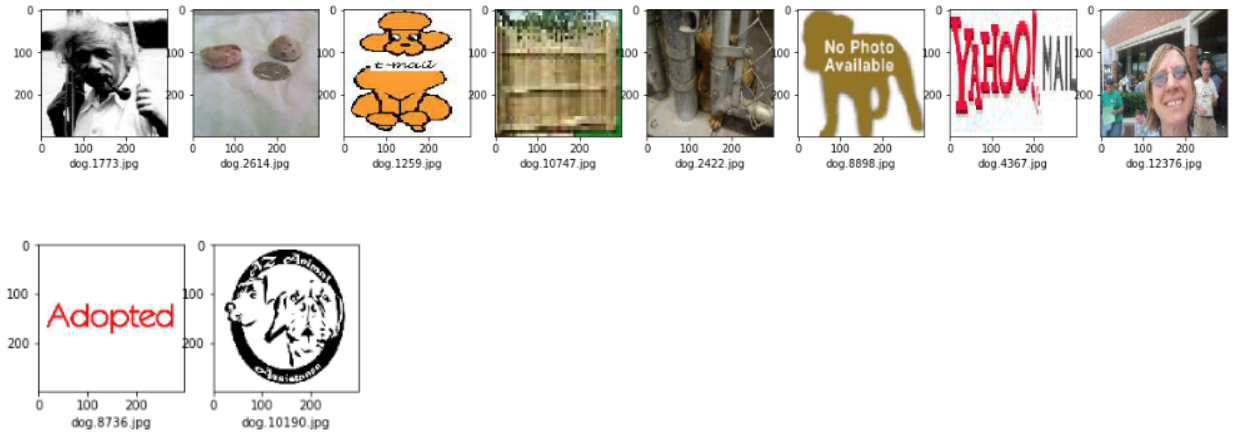
结果可视化

图片尺寸非常小的图片

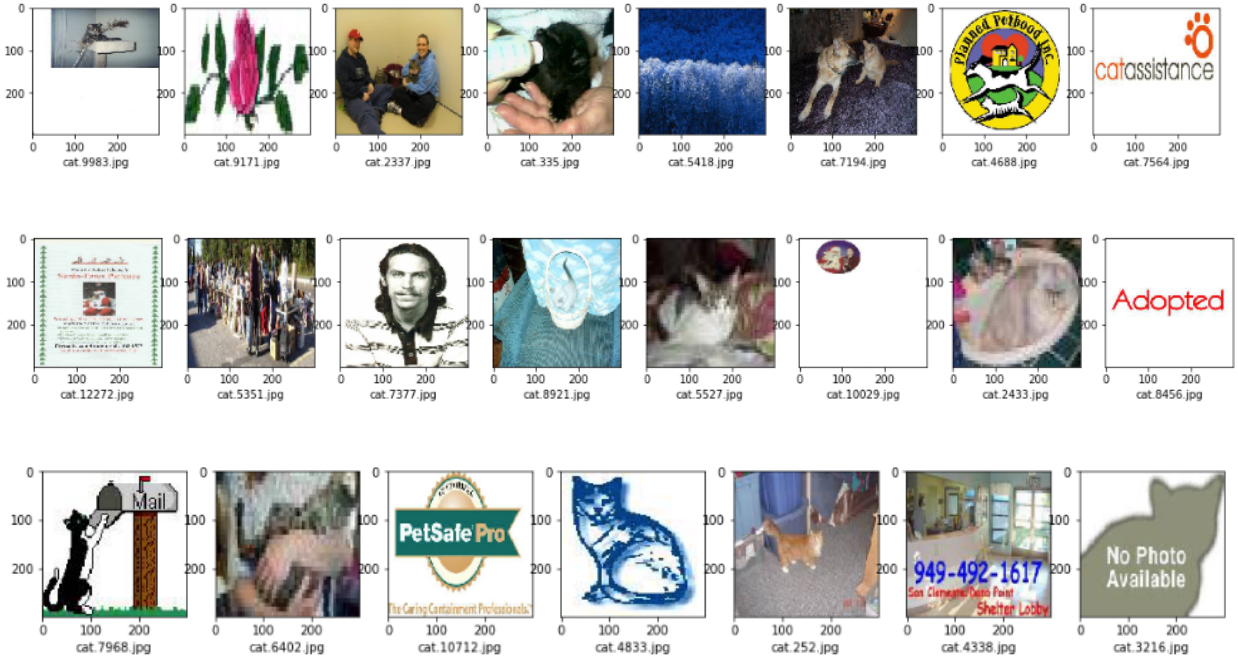


狗中异常图片



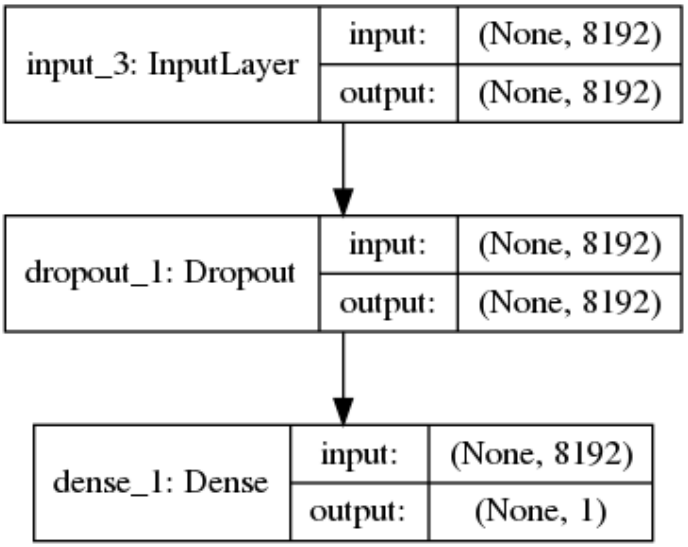


猫中异常图片

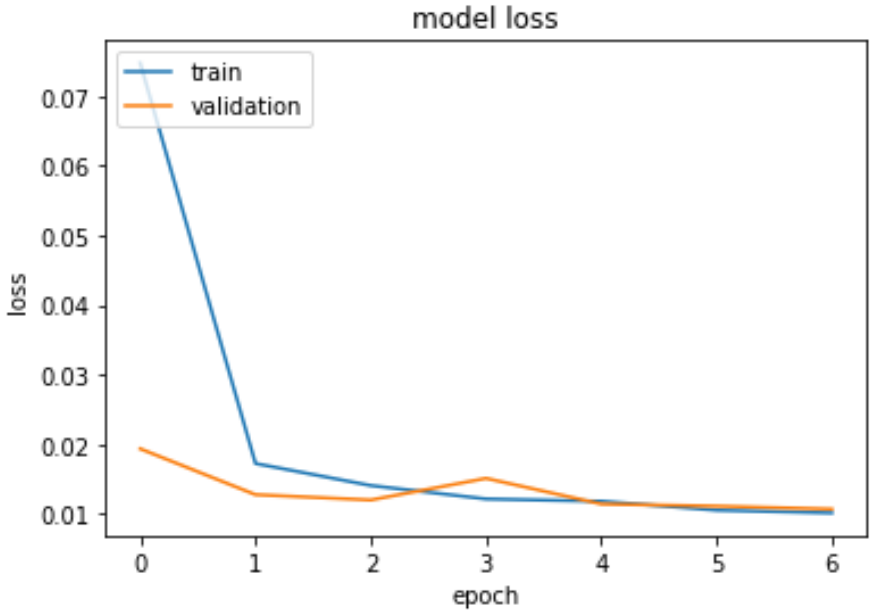


实际上尺寸非常小的图片，Imagenet 经典模型 InceptionV3 也很难预测出来，最终也被视为异常值，因此将这些异常值去掉即可。

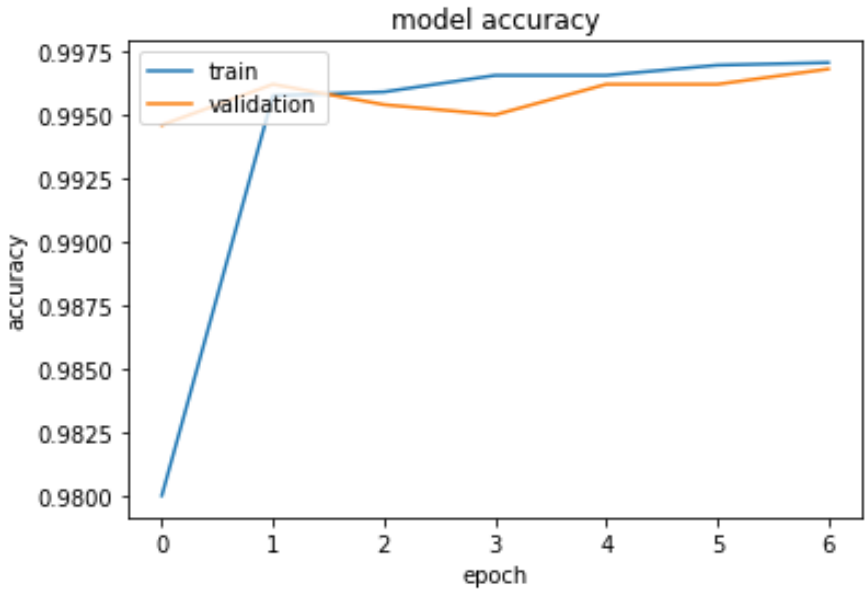
特征值融合的模型结构非常简单



模型特征融合的 Loss



模型特征融合的 Accuracy



对项目的思考

本项目流程:

- 准备数据将，将数据集按照需要放入对应文件夹，其中一份训练集不作处理，一份训练集去除异常值。
- 数据预处理，调用各自模型的预处理方法。
- 构造不同模型,
- 用不同数据集训练不同模型
- 用各个训练好的模型预测测试集
- 上传 kaggle 获取评分

从结果来看, 模型特征融合表现最好训练效率最高。而单模型 fine-tuning 的方式，表现都不如模型融合。单模型 fine-tuning 的表现不佳，有可能是放开训练的层太多，训练数据有限，模型过早过拟合，由于单模型调参耗时较长，这里就不作更多尝试了。

项目由 AWS 的 p3.x2large 实例进行训练，由于显存有限，刚开始时经常爆内存，需要选择合适的数据精度，尽量用 numpy.array 代替 List。

需要做出的改进

本项目只选取了 InceptioV3, Xception, InceptionResnetV2 三个表现较好的经典模型尝试，也可以尝试其它经典模型，特别是尝试其它经典模型的组合融合特征向量， 另外对于单个模型 fine-tuning,由于放开了一些层再次训练，因此也可以对训练集作数据增强再进行训练，可能能取得更好的效果。

参考文献

b. *O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein,et al. Imagenet large scale visual recognition challenge. 2014.*

c. *Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems , 2012.*

d. *Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition arXiv:1409.1556 [cs.CV], 2014*

e. *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition arXiv:1512.03385 [cs.CV]*

f. *Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna Rethinking the Inception Architecture for Computer Vision arXiv:1512.00567 [cs.CV]*

g. *Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning arXiv:1602.07261 [cs.CV]*

h. *François Chollet Xception: Deep Learning with Depthwise Separable Convolutions arXiv:1610.02357 [cs.CV]*

i.