机器学习工程师纳米学位 – 猫狗大战

-项目报告

-张海鹰
-v1 2018.07.16
-v2 2018.07.18

1. 定义

1.1 项目概述

猫狗大战是 kaggle 平台上的一个比赛项目，最终的要求是提供一个模型来识别图片中的对象是猫还是狗，所以抽象来看这是一个图像识别的问题。

图像识别是人工智能的一个重要领域，目前发展十分迅速，应用范围相当广泛，手写数字识别、邮政编码识别、汽车牌号识别、汉字识别、条形码识别，以及如人脸、指纹、虹膜识别等已经在人类日常生活中广泛应用，对经济、军事、文化及人们的日常生活产生重大影响。

支撑其应用实现的重要技术就是深度学习，深度学习被誉为通往人工智能的必经之路，在 2016 年 3 月 Google DeepMind 研发的 AlphaGo 4:1 战胜了世界冠军李世石后，深度学习已经成为现今最为火热的人工智能技术。

1.2 问题陈述

项目要求的最终输出是辨别图片是猫还是狗，因此属于机器学习领域的分类问题，由于没有要求对狗和猫的品种再次细分，因此是二分类； 同时训练数据给出了标签-猫狗，所以是监督学习。

具体的量化方法就是在训练集上训练，同时观察验证集的 loss 以确认模型在某一组参数下的表现,最后在测试集上测试模型，以测试集的 loss 为评价标准。

具体实施方法可以在训练数据集上使用深度学习来进行模型训练，让模型通过给定的训练数据，不断学习到如何识别猫狗的特征。

1.3 评价指标

采用 log loss [1]作为模型评估标准；

其定义如下：

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

评估标准说明：

n 是图片数量；

y_i 是类别标签，1 表示狗，0 表示猫；

$\hat{y_i}$是预测为狗的概率；

Log() 表示自然对数；

loss 值越小，表明模型表现越好；但在模型训练的时候需要观察的是验证集的 loss，不可以使用训练集的 loss 来判断，因为训练集的 loss 会一直下降，如果以此为标准会导

致模型过拟合.

2. 分析

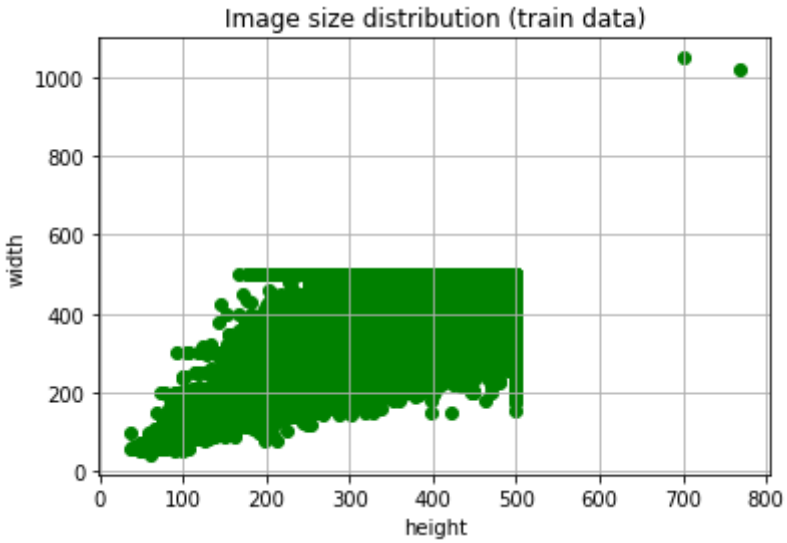
2.1 数据的探索

全部数据都由 kaggle 平台提供，分为了训练集和测试集；

训练集共有图片 25000 张，其中猫 12500 张，狗 12500 张；各占一半；

测试集共有图片 12500 张；

通过散点图查看训练集图片尺寸的分布情况，如下：



训练图片尺寸主要都分布在（200，500）这个范围内，极少数图片尺寸过大（见上图右上角），有部分图片尺寸过小，小于（100，100），见上图左下角。

训练集中有的图片特别模糊，如下：


cat.4821


cat.6402


cat.2433

还有的图片的大小存在不一致，如下：

原图大小 299 x 225　　原图大小 249 x 368　　原图大小 500 x 374
在训练的时候需要 resize;

还有一些明显异常图片（图片中没有猫狗），如下:

在训练时需要删除

2.2　算法和技术

2.2.1　深度学习 & 神经网络 [7]

机器学习学科不断发展过程中,为了完成更加复杂的学习任务,模型的参数越来越多,复杂度越来越高,容量(capacity)越来越大,但一般情况下,复杂模型的训练率低,易陷入过拟合,因此难以受到人们青睐.而随着云计算,大数据时代时代的到来,计算能力的大幅提高可缓解训练低效性,训练数据的大幅增加则可降低过拟合风险,因此,以深度学习(deep learning)为代表的复杂模型走上舞台，开始不断受到人们的密切关注.

神经网络方面的研究很早就已出现,今天的"神经网络"已经是一个相当大的,多学科交叉的学科领域.目前使用的最为广泛的定义是"神经网络是由具有适应性的简单单元组成的广泛并行互连的网络,它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应"[kohonen,1988],我们在机器学习中谈论神经网络时指的是"神经网络学习",或者说是机器学习与神经网络这两个学科的交叉部分.

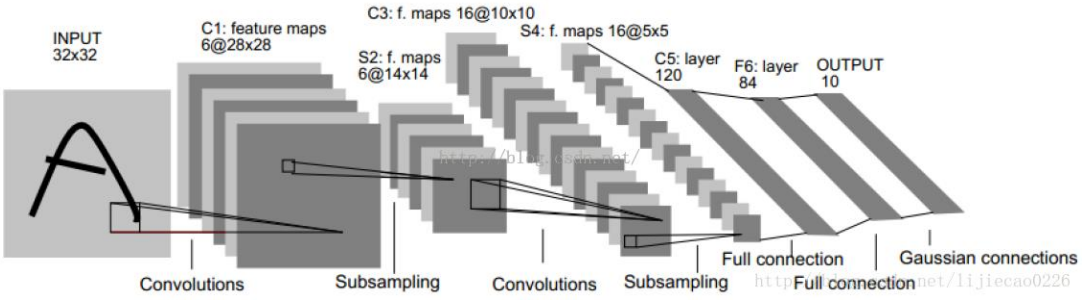
2.2.2　深度神经网络 & 卷积神经网络 [7]

典型的深度学习模型就是很深层的神经网络.神经网络中的隐藏层多了,深度加深的同时,相应的神经元连接权重,阈值等参数就会更多.不断增加的隐藏层数量不仅增加了拥有激活函数的神经元数目,还增加了激活函数嵌套层数.多层神经网络难以直接用经典算法(例如标准 BP 算法)进行训练,因为误差在多隐藏层内逆向传播时,往往会发散而无法收敛到稳定状态.

有两种有效手段来处理深层网络的训练:

其一是无监督逐层训练(unsupervised layer-wise trainning),其基本思想是每次训练一层隐结点,训练时将上一层隐结点的输出作为输入,而本层隐结点的输出作为下一层隐结点的输入,这称为"预训练"(pre-training),在预训练全部完成后,再对整个网络进行微调(fine-turing)训练.这种做法其实就是将大量参数分组,对每组先找到局部看来比较好的设置,然后再基于这些局部较优的结果联合起来进行全局寻优,这样就在利用了模型大量参数所提供的自由度的同时,有效节省了训练开销.

其二就是本项目中要使用到的卷积神经网络,它的策略是"共享权重"(weight sharing),即让一组神经元使用相同的连接权重.以 CNN 进行手写数字识别任务为例（如下图所示）



网络输入是一个 32x32 的手写数字图像,输出是其识别结果,CNN 复合多个"卷积层"和"采样层"对信号输入进行加工,然后在连接层实现与输出目标之间的映射,每个卷积层都包含多个特征图(feature map),每个特征图是一个由多个神经元构成的"平面",通过一种卷积滤波器提取输入的一种特征.例如,图中第一个卷积层由 6 个特征图构成,每个特征图是一个 28x28 的神经元阵列,其中每个神经元负责从 5x5 的区域通过卷积滤波器提取局部特征.采样层亦称为"池化"(pooling)层,其作用是基于局部相关性原理进行亚采样,从而在减少数据量的同时保留有用信息.例如图中第一个采样层有 6 个 14x14 的特征图,其中每个神经元与上一层中对应特征图的 2x2 邻域相连,并据此计算输出.通过复合卷积层和采样层,图中的 CNN 将原始图像映射成 120 维特征向量,最后通过一个由 84 个神经元构成的连接层和输出层连接完成识别.CNN 可用 BP 算法进行训练,但在训练中,无论是卷积层还是采样层,其每一组神经元都是用相同的连接权重,从而大幅减少了需要训练的参数数量.

2.2.3 迁移学习

我们在处理现实生活中诸如图像识别、声音辨识等实际问题的时候。一旦你的模型中包含一些隐藏层时，增添多一层隐藏层将会花费巨大的计算资源。这个时候我们就可以使用迁移学习,即我们在他人训练过的模型基础上进行小改动便可快速投入使用。其实就是把已经训练好的模型的权重提取出来,迁移到新网络中,由于新网络中已经有了迁移过来的特征,就不用从零开始训练了.

在猫狗项目中可以使用在 ImageNet 上已经训练好的模型来做迁移学习,可以理解为在 ImageNet 上的预训练模型已经有一个很广泛的视野,我们的新模型可以在此基础上进行训练,从而可以在某一个细分领域(猫狗)能够更加专注.该项目的训练数据是猫狗,因此我们训练出来的新模型是一个猫狗识别"专家",如果有一个水果图片的数据集,我们还可以在此基础上训练一个"水果"识别专家.

2.2.4 模型融合

模型融合就是训练多个模型，然后按照一定的方法把他们都集成在一起，因为它容易理解、实现简单，同时效果也很好，在天池、kaggle 比赛中被经常使用。可以理解为先产生一组个体学习器,再用某种策略将它们结合起来,加强模型的效果,具体的策略有平均法，投票法，学习法.

在猫狗项目中,我会融合多个在 ImageNet 上经过预训练的模型，然后使用学习法再次训练一个新模型.

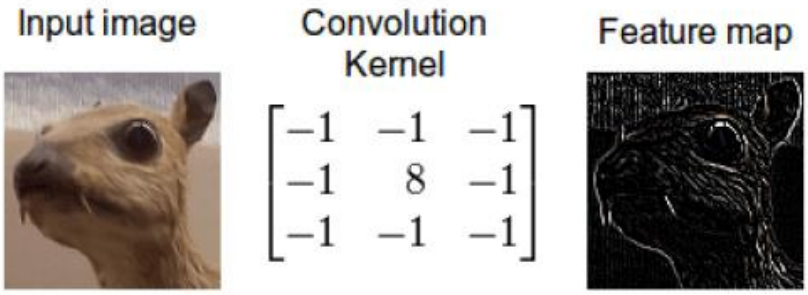
可以将多个不同的网络输出的特征向量先保存下来，导出的时候记得利用池化层将卷积层输出的每个激活图直接求平均值，这样保证输出文件不是特别大，也不易过拟合， 由于这些都是在 ImageNet 上预训练过的，所以

每一个模型都有很好的图像识别能力， 所以把他们任意组合起来， 或者多个叠加起来可以做到眼观六路，耳听八方。有了这个思路，我们接着载入这些特征向量，把它们合成一条特征向量，然后模型搭建就在后面添加 dropout，再添加分类就完成了。

2.2.5 算法之卷积神经网络 [8]

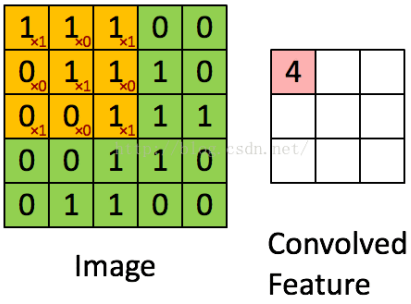
本项目主要使用的是卷积神经网络（Convolutional Neural Network, CNN），其中最重要的就是“卷积”的概念，可以把卷积想象成一种混合信息的手段。想象一下装满信息的两个桶，我们把它们倒入一个桶中并且通过某种规则搅拌搅拌。也就是说卷积是一种混合两种信息的流程。卷积也可以形式化地描述，事实上，它就是一种数学运算，跟减加乘除没有本质的区别。虽然这种运算本身很复杂，但它非常有助于简化更复杂的表达式。

当我们在图像上应用卷积时，我们在两个维度上执行卷积——水平和竖直方向。我们混合两桶信息：第一桶是输入的图像，由三个矩阵构成——RGB 三通道，其中每个元素都是 0 到 255 之间的一个整数。第二个桶是卷积核（kernel），单个浮点数矩阵。可以将卷积核的大小和模式想象成一个搅拌图像的方法。卷积核的输出是一幅修改后的图像，在深度学习中经常被称作 feature map。对每个颜色通道都有一个 feature map。



CNN 为什么对图像领域更加有效，因为它不但关注了全局特征，更是利用了图像识别领域非常重要的局部特征，它将局部特征抽取的算法融入到了神经网络中。图像本身的局部数据存在关联性，而这种局部关联性的特征是其他算法无法提取的。深度学习很重要的是对全局和局部特征的综合把握。

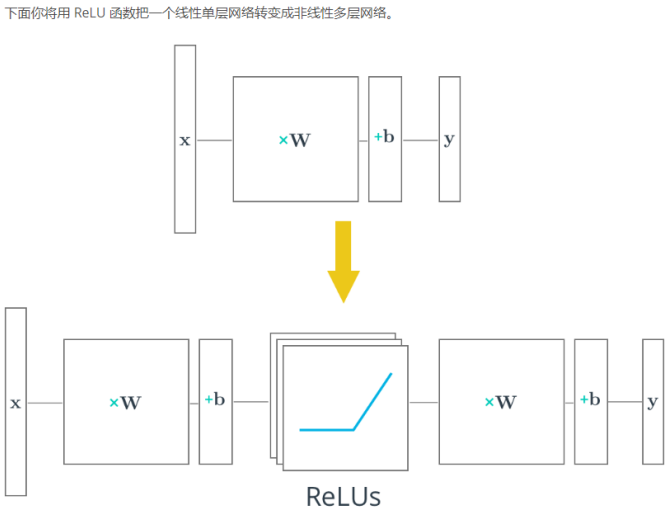
CNN 中最常见的就是卷积层（Convolution layer），池化层（pooling layer）：卷积层主要用来进行特征提取（局部连接，权值共享），从图像中随机选取一小块区域作为训练样本，从该样本中学习到一些特征，然后将这些特征作为滤波器，与原始整个图像作卷积运算，从而得到原始图像中任意位置上的不同特征的激活值，如下图：



通过将卷积层提取的特征输入至分类器中进行训练，可以实现输出最终

的分类结果。理论上可以直接输出，然而这将需要非常大的计算开销，特别是对于大尺寸高分辨率图像，由于图像具有一种“静态性”的属性，在图像的一个局部区域得到的特征极有可能在另一个局部区域同样适用。因此，对图像的一个局部区域中不同位置的特征进行聚合统计操作，这种操作统称为池化，因此池化层可以理解为在减少数据量的同时保留有用信息，池化有平均池化和最大池化的不同实现方式。

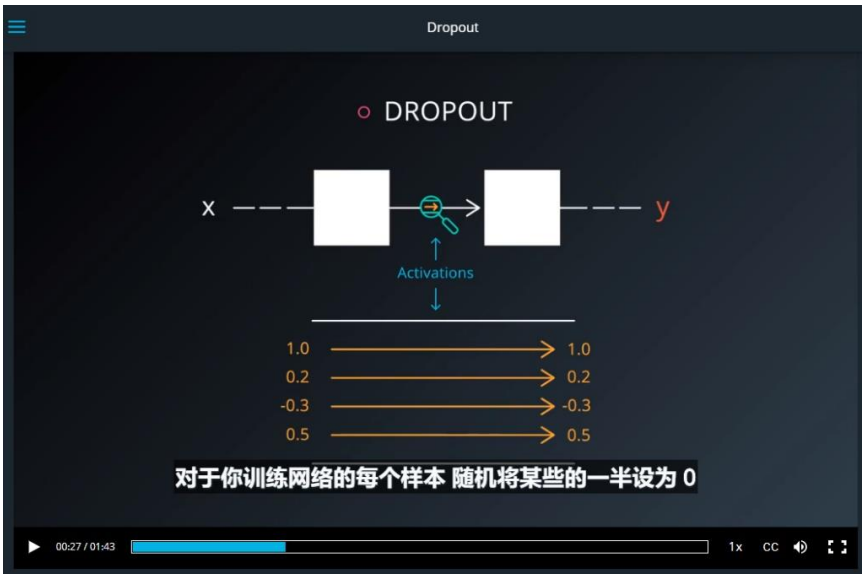
CNN 中每一个隐藏层都会有激活函数，一个常用的非线性函数叫 ReLU（rectified linear unit）。ReLU 函数对所有负的输入，返回 0；所有 x>0 的输入，返回 x。把 ReLU 函数放到隐藏层，就像开关一样把负权重关掉了。在激活函数之后，添加像输出层这样额外的层，就把模型变成了非线性函数。这个非线性的特征使得网络可以解决更复杂的问题，如下图：



CNN 的训练过程中常常会出现“紧身裤”问题，即过拟合，有两种方式可以解决这个问题，一个是 L2 Regularization， 它的思想是向损失函数添加另一个项，用于惩罚大的权重，如下图：

另外一个就是我们项目中使用到的 Dropout，基本上随机的将流经网络的一半数据完全摧毁了，然后再一次，再一次的随机重复 dropout，这样的网络将永远不依赖于任何给定的激活而存在，所以它被迫学习一切冗余的表

示，以确保至少将一些信息保存下来，如下图：



CNN 的训练过程中可以尝试不同的优化器：本项目使用 keras 搭建网络，因此重点介绍 keras 中的 SGD， Adagrad，Adadelta， Adam。

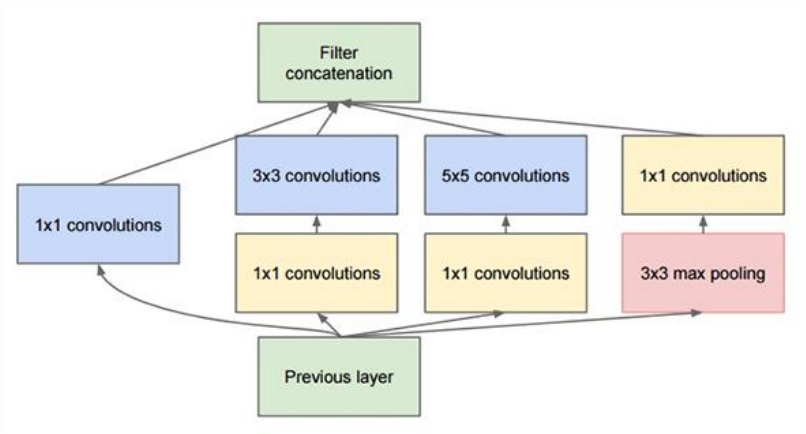
SGD 是随机梯度下降优化器，主要调节参数是 lr-学习率，momentum-用于加速 SGD 在相关方向上前进，并抑制震荡，decay-每次参数更新后学习率衰减值，nesterov-是否使用 Nesterov 动量。缺点是选择合适的 learning rate 比较困难，以及容易收敛到局部最优。

Adagrad，主要调节参数 lr-学习率， decay-每次参数更新后学习率衰减值， 该算法其实是对学习率进行了一个约束，缺点是仍依赖于人工设置一个全局学习率。

Adadelta 是对 Adagrad 的扩展，主要调节参数是 lr-学习率，roh-Adadelta 梯度平方移动均值的衰减率，decay-每次参数更新后学习率衰减值，最初方案依然是对学习率进行自适应约束，但是进行了计算上的简化。其特点是训练初中期，加速效果不错，很快，训练后期，反复在局部最小值附近抖动。

Adam(Adaptive Moment Estimation)本质上是带有动量项的 RMSprop，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。Adam 的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。主要调节参数 lr-学习率，beta_1， beta_2, epsilon-模糊因子，decay-每次参数更新后学习率衰减值

项目中使用了 InceptionV3 来检测异常值，InceptionV3 其实是一个衍化的过程：之前的一些网络，只是一味增加卷积层的深度，但是在单层上卷积核却只有一种，比如 VGG，单层卷积核只有 3x3 大小的，这样特征提取的功能可能就比较弱。GoogLenet 想的就是能不能增加单层卷积层的宽度，即在单层卷积层上使用不同尺度的卷积核，GoogLenet 构建了 Inception module 这个基本单元，基本的 Inception module 中有 1x1 卷积核，3x3 卷积核，5x5 卷积核，还有一个 3x3 池化层，从而产生了 Inception v1 模型，如下图所示：



Inception v2 的网络在 v1 的基础上，进行了改进，一方面了加入了 BN 层，减少了 Internal Covariate Shift（内部 neuron 的数据分布发生变化），使每一层的输出都规范化到一个 N(0, 1)的高斯，另外一方面学习 VGG 用 2 个 3x3 的 conv 替代 inception 模块中的 5x5，既降低了参数数量，也加速计算；

在 v2 的基础上，在进行如下实验改进，使用大尺寸滤波器卷机分解，将大卷积分解成小卷积，附加分类器，降低特征图大小，继续得到 V3 版本。

2.3 基准模型

准备使用 InceptionResNetV2 [6], Xception [4], InceptionV3 [5]作为基准模型，期望能在项目中自己搭建的模型成绩优于这三个单独的基准模型。

另外本项目要求是进入 kaggle 比赛前 10%的水平，该竞赛有 1314 支队伍，即最终结果需要排名在 131 位之前，即成绩要优于 131 名的成绩 0.06127.

3. 方法

3.1 数据预处理

3.1.1 异常值处理

通过观察 keras 预训练模型文档:

模型	大小	Top-1 准确率	Top-5 准确率	参数数量	深度
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

根据 top-5 准确率，决定选用 InceptionResNetV2, Xception, InceptionV3 三个预训练模型来识别训练集中的异常图片；

最终发现 40 张异常图片,列举 16 张如下

异常图片共 40

40 张图片全部移动到 unknown 文件夹下面,并不参与训练;

仔细辨认被确认的 40 张异常图片，其中根本没有猫和狗的多达 27 张，例如:

图像中存在猫狗，但是不够清晰，并且占比例太少的占的有 11 张，例如:

图像分辨率特别小，特别模糊的，有 5 张，例如:

三个预训练模型 InceptionResNetV2, Xception, InceptionV3 检测异常值的准确度相差不大，因为我是根据 keras 文档中 top-5 的准确率前三位来选择的。

最终我选用 top-20 来检测异常值，因为我使用 InceptionResNetV2,在 1000 个样本下实验，top3 检测，结果是 42 pics， 会发现有大量正确的猫狗图片在内, top10 检查，结果是 13 pics，仍有少量正确的猫狗图片, top20 检测， 6 pics, 已经没有正确的猫狗图片。其它两个模型实验数据差不太多，所以最终选择了 top-20。

由于每个预训练模型有自己对图片格式的要求，因此使用不同的模型需

要使用对应模型的预处理，异常值检测使用的三个预训练模型 InceptionResNetV2，Xception，InceptionV3 对应的预处理分别是 inception_resnet_v2.preprocess_input(x)，xception.preprocess_input(x)，inception_v3.preprocess_input(x)，都是将输入归一化至[-1,1]. 后面融合模型使用的 ResNet50 的预处理函数不太一样，它的是 resnet50 preprocess_input(x)，它除了缩放到 224x224，还对图像进行了减均值的操作。

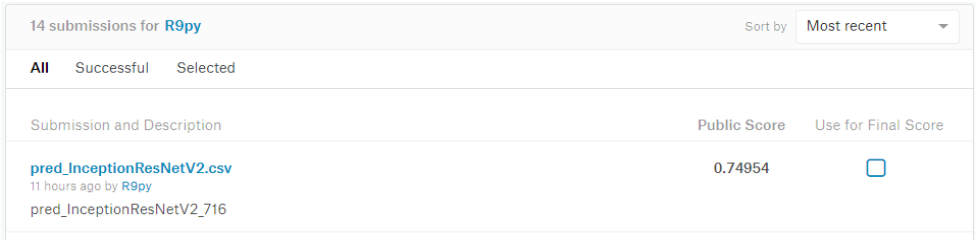
3.1.2 图片分类

因为会使用 keras 的 ImageDataGenerator，因此将猫狗图片分别存放在 cat 和 dog 文件夹下面

3.2 执行过程

Step1：在单独的预训练模型上使用迁移学习

- 在 InceptionResNetV2 的基础上建立自己的全连接层，添加分类器，训练 10 代，使用 SGD 优化器，lr 设置为 0.0001，提交 kaggle 成绩为 0.74954

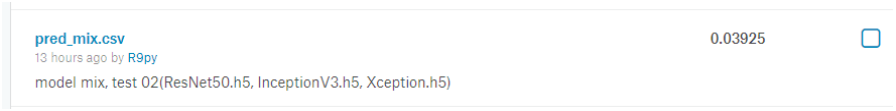


- 在 Xception 的基础上建立自己的全连接层，添加分类器，训练 10 代，使用 SGD 优化器，lr 设置为 0.0001，提交 kaggle 成绩为 0.69482

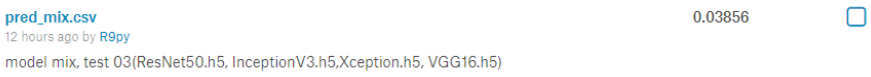
- 在 InceptionV3 的基础上建立自己的全连接层，添加分类器，训练 10 代，使用 SGD 优化器，lr 设置为 0.0001，提交 kaggle 成绩为 0.75665

Step2：模型融合

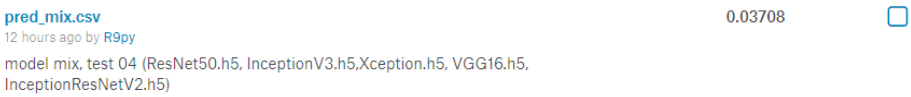
- 考虑使用多个模型进行融合，分别提取 ResNet50 [3], Xception, InceptionV3, InceptionResNetV2, VGG16 模型的特征，保存到 h5 文件
- 融合：载入保存在 h5 文件中的特征向量，合并成一条特征向量
  方案 1：使用 ResNet50, Xception, InceptionV3 三个模型融合，提交 kaggle 成绩为 0.03925



方案 2：使用 ResNet50, Xception, InceptionV3, VGG16 四个模型融合，提交 kaggle 成绩为 0.03856



方案 3：使用 ResNet50, Xception, InceptionV3, InceptionResNetV2,VGG16 五个模型融合，提交 kaggle 成绩为 0.03708



Step3　模型搭建

- 根据上面的成绩，决定使用五个模型一起融合
- 新模型很简单，添加 dropout 防止过拟合，添加一个分类器，激活函数使用 sigmoid，优化器使用 adadelta, loss 使用 logloss
-
-

Step3：模型训练

- 使用 batchSize=128，validation data = 0.2 的参数进行训练

问题记录：

- class_mode='binary'
  出现过这里设置为'catetory'的错误
- Dense(1, activate="sigmoid")
  出现过这里使用的 2 的错误
- Shuffle= false
  提取特征的时候，出现过没有设置 shuffle=false 的错误
- fit_generate()的 steps 参数
  提取特征的时候，出现过没有设置 steps 的错误

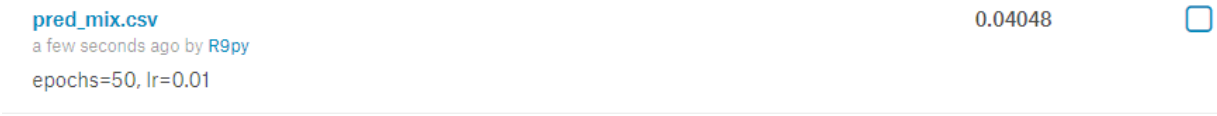
3.3　完善

在融合五个模型的基础上，调整 learn rate 和 dropout 参数，测试模型的效果

Learn rate 参数：

Learn rate=0.1，epochs=16，提交 kaggle 成绩为 0.03841

Learn rate=0.01，epochs=50，提交 kaggle 成绩为 0.04048



Learn rate=0.001，epochs=50，提交 kaggle 成绩为 0.04828

综合上面成绩，learn rate 还是选用默认值 1

Dropout 参数：
Dropout=0.2，epochs=8，提交 kaggle 成绩为 0.03857

Dropout=0.8，epochs=8，提交 kaggle 成绩为 0.03723

综上所诉，使用 Adadelta 默认参数，以及 dropouot=0.5 即可取得最优的成绩。

4. 结果

4.1 模型的评价与验证

4.1.1 单个模型上的迁移学习

Keras 目前集成了 9 种预训练模型：Xception,VGG16,VGG19,ResNet50, Inception50,InceptionResNetV2,MobilNet,DenseNet,NASNet.

ResNet 的出现使得神经网络可以实现更深的深度；而 Inception 在面对更宽的神经网络的时候可以很好的提取特征；Xception 则融合了两者的优点；我们可以针对具体情况使用具体的预训练模型来进行迁移学习。

4.1.2 多个模型的融合

和单一模型上的迁移学习相比，从多个模型中提取特征并将其组合在一起，是希望可以集合每个模型的特长，综合全部模型的优点，全方位利用不同模型提取的不同特征信息。

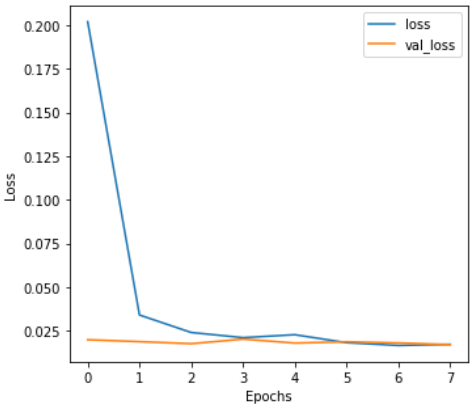
4.2 合理性分析

单独使用预训练模型的迁移学习都没有在 kaggle 取得好成绩，使用融合多个模型的特征，在 kaggle 取得了进入前 10 名的成绩，所以融合模型是可以大幅提高模型预测准确率的。

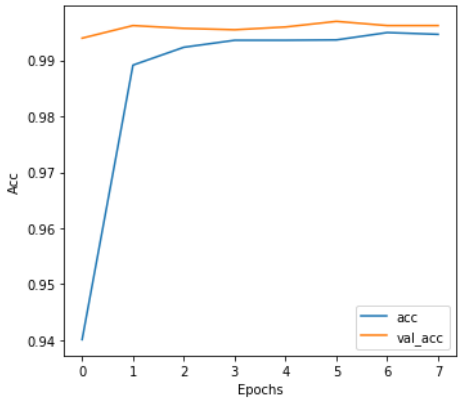
5. 结论

5.1 结果可视化

Loss 的可视化



Acc 的可视化



5.2 对项目的思考

本项目在开发工具上使用了 keras，大大简化了模型的搭建，但实施简单的同时，很容易让自己陷入一个误区，就是误以为机器学习建模很简单，但其实还需要继续阅读更多模型的论文，理解其内部原理，才能在遇到瓶颈的时候知道如何改进。这也是我目前的困难所在，虽然可以参考文档实现项目，但我觉得我还需要多尝试多动手，更多的理解模型原理。

该项目的一大特点是利用了 ImageNet，以及在 ImageNet 在的预训练模型，该项目的加工是在特定的猫狗领域，如果推广至其它一些领域是否会有好的效果，也是值得思考的。

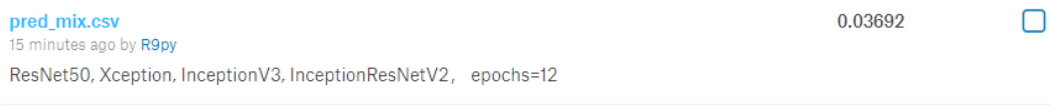
有个很有意思的的地方是，我尝试从融合模型数量上的增加来增加成绩，预期是随着数量的增多，成绩会有大幅度提升，但并没有：

我用 ResNet50, Xception, InceptionV3 三个模型融合提交 kaggle 成绩为 0.03925；

我用 ResNet50, Xception, InceptionV3, VGG16 四个模型融合，提交 kaggle 成绩为 0.03856；

我用 ResNet50, Xception, InceptionV3, InceptionResNetV2,VGG16 五个模型融合，提交 kaggle 成绩为 0.03708；

此时我期望更加进一步提升成绩，但我也开始反向思考，真的是越多越好吗？会不会有些模型在里面反而是拖后腿了？翻看模型相关论文后，发现 VGG16 和 VGG19 在 2014 年的时候是相当先进的模型，16 层和 19 层的深度在当时也是深度很深的，但随着神经网路的发展，目前的网络很多都是上百层，于是想说在当前的条件下，VGG 模型是不是相当于众多博士生中的一个本科生？后面的实验去掉了 VGG，融合 ResNet50, Xception, InceptionV3, InceptionResNetV2 四个模型，提交 kaggle 成绩 0.03692：



因此我明白了融合多个模型可以提升成绩，但同时也要考虑模型自己的特性。

5.3 需要做出的改进

- 可以尝试使用不同的融合方案，例如投票法，平均法
- 可以尝试使用数据增强，会大幅度增加训练数据的数量
- 是否可以尝试网格搜索法来自动寻找一些超参数的最佳值

6. 参考

[1] LogLoss
http://scikit-learn.org/stable/modules/model_evaluation.html#log-loss
[2] 面向小数据集构建图像分类模型
https://keras-cn-docs.readthedocs.io/zh_CN/latest/blog/image_classification_using_very_little_data/
[3] Deep Residual Learning for Image Recognition
https://arxiv.org/abs/1512.03385
[4] Xception: Deep Learning with Depthwise Separable Convolutions
https://arxiv.org/abs/1610.02357
[5] Rethinking the Inception Architecture for Computer Vision
https://arxiv.org/abs/1512.00567
[6] Why do deep convolutional networks generalize so poorly to small image transformations?
https://arxiv.org/abs/1805.12177
[7] 《机器学习》 周志华
[8] Understanding Convolution in Deep Learning
http://timdettmers.com/2015/03/26/convolution-deep-learning/
[9] ImageNet: VGGNet, ResNet, Inception, and Xception with Keras
https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/