猫狗大战分类报告

1. 问题的定义

项目概述

这个项目是创建一个分类算法来区分图片中包含的是猫还是狗。这个问题对于人类，甚至是猫狗来说都是简单的。但是对于计算机来说却有点困难。我们把这样的问题称为CAPTCHA[1] (Completely Automated Public Turing test to tell Computers and Humans Apart) 或者 HIP (Human Interactive Proof). HIPs 被用于很多方面，例如减少垃圾邮件和博客，防止对网站密码的硬攻击。

这个项目将会关注各种算法的实现，从经典的卷积神经网络（CNN）到最先进的模型（例如：VGGNet，ResNet，Inception）。我会比较这些方法的准确度和性能表现。

针对这个项目有许多公开的数据集可以使用。我选择的数据集来自Kaggle的Dogs vs. Cats Redux: Kernels Edition 竞赛[2]. 它有两部分数据：训练集和测试集。训练集目录包含25000张猫或者狗的图片。每张图片使用标签作为名字的一部分。测试集目录包含12500张图片，用数字命名。对于测试集中的每张图片，我会预测其包含狗的概率（狗=1，猫=0）。

问题陈述

猫狗大战是一个监督二分类问题。我们有两类图片：猫和狗。我们的目标是抽取合适的特征并创建一个有效的模型用以对每张猫或者狗的图片进行分类。

我们可以通过探索数据集获取其基本信息。我们发现每张图片的尺寸不一致，所以得修改尺寸。因为特定模型对输入尺寸有一定要求（比如：VGGNET 要求图片尺寸为224 × 224， INCEPTION要求的尺寸为229 × 229）。除此以为， 完整的训练集需要拆分成训练集和验证集。当我们开始训练模型时，我们通过验证集评估模型的性能。通过观察损失（Loss）和精确度（Accuracy），我们可以知道模型是否正确并且是否过/欠拟合。最后我们会在测试集上做出预测并且把结果上传到Kaggle来查看具体的名次。

评价指标

通常我们使用ROC curve (receiver operating characteristic curve) 和 PR curve (Precision-Recall curve) 评估二分类模型的性能。然而，为了遵守Kaggle的竞赛规则，我将使用Log Loss作为评估指标。

这里

n 是测试集中的图片数目

是预测某张图片是狗的概率

图片为狗时等于1，为猫时等于0

是以e为底的自然对数

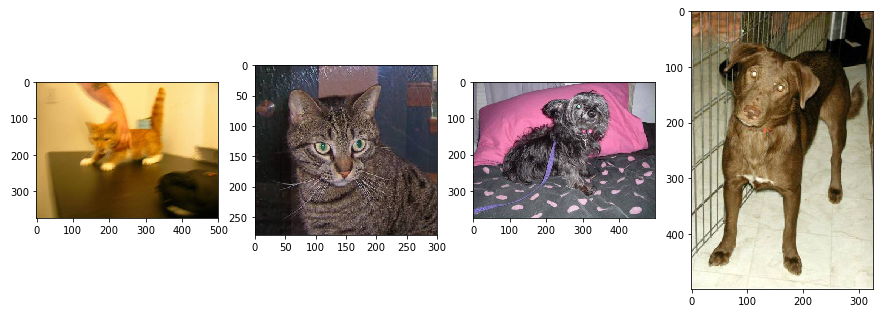
LogLoss的值越小越好。

1. 分析

数据的探索

数据集下载自： <https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data>. 总共两个目录：train 和 test。Train 目录有25000张图片，图片的名字有如下规律： dog.1.jpg， dog.2.jpg， cat.1.jpg 等。Test目录有12500张图片，图片的名字有如下规律：1.jpg， 2.jpg， 3.jpg等。

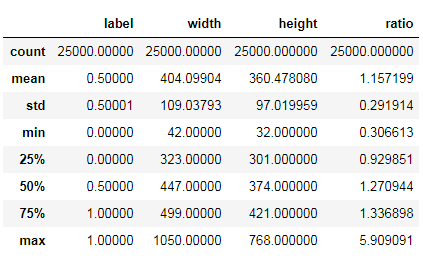
下面是一些图片的例子（标明了高和宽的尺寸）：



通过观察上面的图片我们注意到一些特点：

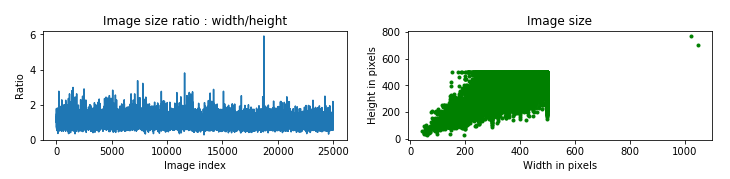
1. 图片中的动物（猫或狗）基本处于图片中心。
2. 图片尺寸不一。
3. 有些图片图像质量较差（比如第一张）。
4. 图片背景各异，种类不同（比如第一张的人手和第三章的粉色枕头）。

下面的表格显示了一些图片的概括信息：宽，高，宽高比。你可以忽略label，那只是个类目值。此表格清晰地显示了图片的尺寸有多么不同。



探索性可视化

我们已经注意到图片的尺寸不一。下面的图像显示了每张图片的宽，高，宽高比的分布。



平均宽高比是1.157。

我们可以看到两个异常点。相比其他数据点，它们的宽和高异常的大。在右边的图上，我们同样可以注意到两个异常点，它们的宽高比有6，但是其他点只有2。

因为大多数模型（例如 VGGNet，Inception）要求特定的图片尺寸，这些图像可以帮助我们决定是否需要改变图片尺寸或者去除掉这些图片。

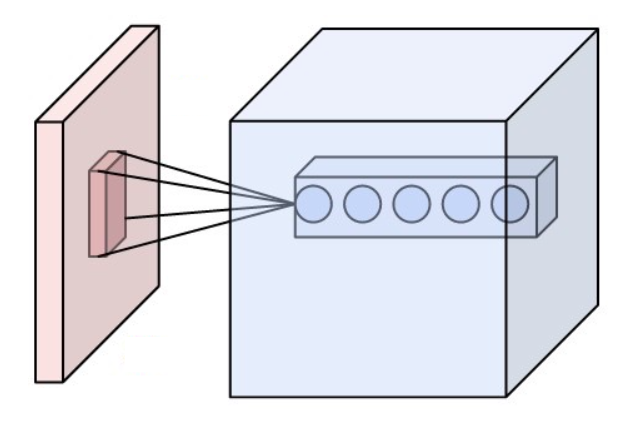
算法和技术

对于分类问题，常用的方法是Logistic回归或者更普遍的softmax回归。然而这两类方法当数据变化剧烈（许多不同的输入但是有同样的标签）时的表现不佳。而这正是我们需要处理的问题：任意两张猫的图片可能完全不一样。同时图片中动物所在位置和背景的不同更加大了难度。

最近一种流行的处理图像分类问题的方法是卷积神经网络（CNN）。当图片中重要的特征位置不定时（例如猫的耳朵的一部分）这种方法特别有效。我们需要处理的正是这种问题：图片中的猫狗可能处于不同的位置并且还可能有多只动物在同一张图片中。

CNN包含一个输入和一个输出层，同时有许多隐藏层。 隐藏层通常包括卷积层，池化层，非线性激活层，全链接层，有时也有Dropout层。[2]

卷积层是CNN的核心构件。 这层的参数由一组可学习的过滤器（或内核）组成，这些过滤器具有较小的接受区域，但是延伸到输入区域的全部深度。 在正向传递过程中，每个过滤器在输入体积的宽度和高度上进行卷积，计算过滤器的输入和输入之间的点积，并生成该过滤器的二维激活图。 因此，网络会学习在输入中的某个空间位置检测到某种特定类型的特征时激活的过滤器。



*Neurons of a convolutional layer (blue), connected to their receptive field (red)*

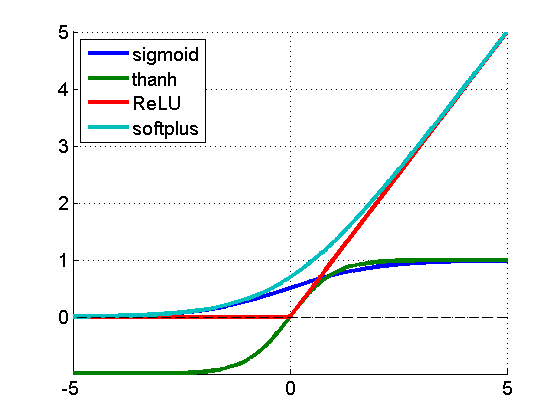
CNN的另一个重要概念是池化层，这是一种非线性下采样的形式。 有几个非线性函数来实现池化，其中最大池化（Max Pooling）是最常见的。 它将输入图像分成一组不重叠的矩形，并为每个这样的子区域输出最大值。 这是基于这样的直觉：某个特征的确切位置不如其相对于其他特征粗略的位置重要。 池化层用于明显减小表示的空间大小，以减少网络中的参数数量和计算量，并因此也控制过拟合。 周期性地在CNN体系结构中的连续卷积层之间插入池化层是常见的。 另外池化层操作提供了另一种形式的转换不变性。



*Max pooling with a 2\*2 filter and stride = 2*

ReLU 是 Rectified Linear Units的缩写。. 该层使用非饱和的激活函数：。它增加了决策函数和整个网络的非线性特性，而不影响卷积层的接受域。

其他函数也用于增加非线性，例如饱和双曲正切：,, 和sigmoid函数： . ReLU 通常比其他函数更受欢迎，因为它训练训练神经网络很快，而不会对泛化精度造成严重损失。



*Comparison of different activation functions*

最后，经过几个卷积和最大池化层之后，神经网络中的高层推理是通过全链接层进行的。 全链接层中的神经元与前一层中的所有激活都有联系，正如常规神经网络中所见。 因此它们的激活可以用一个矩阵乘法和一个偏置偏移来计算。

有时会有一个Dropout层。 由于全链接层包含了大部分参数，因此容易出现过度拟合。 减少过拟合的一种方法是Dropout。 在每个训练阶段，单个节点或者以概率 丢弃单个节点或者以概率p保留，从而生成调整后的网络; 输入和输出边缘到一个被丢弃的节点也被删除。 只有调整后的网络在这个阶段对数据进行训练。 之后删除的节点与他们的原始权重被重新插入网络。

因为每张图片都被打过标签（比如这张猫的图片），并且我们知道每个神经元权重的影响，我们就可以调整这些权重以使模型预测这张是猫的图片达到接近100%的确信度。这种调整方式称之为反向传播（backpropagation）。为了实现这个，我们需要计算一个损失函数，并且用一个优化器找到此函数的最小值，这样我们就找到了这些权重的最佳组合。

除了使用手写的CNN模型，我也尝试了一些预训练模型用作迁移学习。它们包括VGGNet， ResNet 和Inception/Xception。

VGGNet起源自论文Very Deep Convolutional Networks for Large-Scale Image Recognition作者是Karen Simonyan 和 Andrew Zisserman [3]. 在论文中，他们研究了卷积网络深度和大规模图像识别设定下的准确度之间的关系。他们的主要贡献是对一种使用非常小（3×3）的卷积过滤器的加深网络结构彻底的评估。它表明通过将深度推到16-19个权重层可以实现对现有技术配置的显着改进。 这些发现是2014年ImageNet挑战赛提交的基础，他们的团队分别获得了本地化和分类赛的第一名和第二名。 他们还表明这种网络结构对其他数据集也有很好的普适性，并且达到了很好的预测的结果。

ResNet是由Kaiming He, etc [4] 创建的。他们提出了一个残差学习框架，以翻遍网络训练，这种结构比以前使用的网络要深得多。 他们明确地将层次重新定义为根据输入学习residual，而不是学习不相关的功能。 它们提供了全面的经验证据，表明这些残差网络更易于优化，并且可以从深度上获得更高的精度。 在ImageNet数据集上，他们评估了深度达152层的网络 - 比VGG网络深8倍，但复杂度较低。 这些残余网络的集合在ImageNet测试集上达到了3.57％的误差。 这个结果在ILSVRC 2015分类任务中赢得了第一名。

InceptionV3是由Google发布的，他们的工作已被记录在“Rethinking the Inception Architecture for Computer Vision”[5]。 他们探索扩大网络的方法，旨在通过适当分解卷积和积极正则化，尽可能有效地利用附加计算。 他们在ILSVRC 2012分类挑战验证集上对他们的方法进行了基准测试，结果表明：在使用网络计算成本为50亿次乘法的单个网络评估中，前21.2％的前1位和前5位的误差为5.6％ 增加了每个推断和使用少于25万个参数。 对于4个模型和多作物评估的集合，他们报告验证集（测试集上3.6％的错误）的3.5％（排名前5）和验证集上的17.3％ （排名第一）。

基准模型

Kaggle记录板显示数据科学家们能以0.05629的Log Loss达到前100名。计算Log Loss的方法已经在评估指标部分描述过了。现在让我们看看我们的模型是否能超过这个成绩。

1. 方法

数据预处理

理想情况下，模型需要一些高质量的输入，同时图像尺寸不能太大，以至于模型无法处理。 为了在大量信息中找到模式，我们需要大型模型，由于我们有限的计算能力，这是不可能的。 另外，如果过滤的信息较少，则更容易理解图像内容。 所以具有较低分辨率的图像很重要，我们可以通过调整大小来实现这一点。

我们可以看到在“探索性可视化”部分中提到的最大宽度和高度的两个异常值图像。 但是，我不想删除它们，因为模型着重于理解图像的内容，而尺寸不影响结果。 所以我只需要调整图像大小，使它们保持一致。

此外，根据VGG和ResNet相关论文所说，每张图片还需要减去它们在ImageNets上得到的平均RGB值。

我将把train文件夹下的图像分成训练和验证集。 验证集用于验证模型是否过度适应训练集。 我将提取20％的数据作为验证集。 由于整个测试数据都在test文件夹中，因此我可以将其直接作为测试集。

|  |  |  |
| --- | --- | --- |
|  | Cats Number | Dogs Number |
| Training Set | 10000 | 10000 |
| Validation Set | 2500 | 2500 |

所有数据集都以numpy数组存储。我尝试将其保存为pickle文件，这样我们可以很容易地导入数据。然而由于AWS的存储大小限制，我不得不删除这些pickle文件。

执行过程

数据探索部分存放在单独的notebook（data exploration.ipynb）中。 在那我列出了数据集的基本统计信息也通过一些图像展示了数据分布情况。

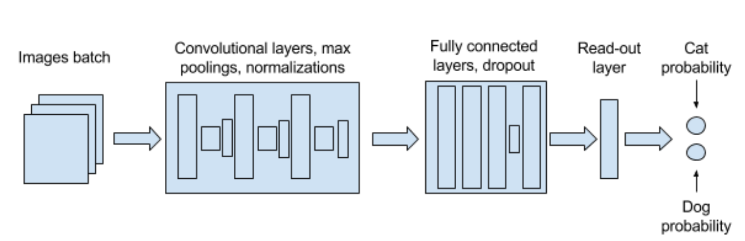
然后我创建了另一个notebook用以演示手工创建CNN。这里我参考了VGG16的结构。它有4个连接组成方法（CCM）。每个CCM由以下结构：

卷积层->规范层->卷积层->规范层->最大池化层

规范层（Batch Normalization）是一种减少神经网络内部协变量的方法，首先在[6]中描述，从而导致更高学习率的可能使用。 原则上，该方法在层之间添加额外的步骤，其中层之前的输出被标准化。

在4个CCM之后，最后3层分别是Flatten层，Dropout层和Dense层。

完整模型结构如下：



每种层次类型的数量是迭代确定的，基于测试过的每种模型结构的性能表现。

为了方便地迭代不同的算法体系结构和参数组合以找到合适的模型结构，我们可以利用从检查点保存和加载训练模型的功能，因为很容易将训练步骤添加到已经训练好的模型中。通过保存Jupyter notebook的副本存储成功的培训课程的模型-性能对，以供以后分析。

我选择Keras来建立模型。 Keras是一个高级神经网络API，用Python编写，能够在TensorFlow，CNTK或Theano之上运行。 它的开发重点是快速实验。 能够从最初的想法到尽可能延迟的结果是做好研究的关键。 所以我不需要从头开始构建模型，而是关注模型结构。

算法主要步骤如下：

1. 按照上面介绍的定义模型结构。
2. 给定输入参数（例如：CCM的个数和核心深度）创建模型实例。
3. 使用合适的损失函数，优化器，性能指标编译模型。
4. 并行使用训练集和验证集训练模型。我们观察损失函数在训练和验证集上是否下降，同时准确度在两者上是否上升，这能告诉我们模型是否欠/过拟合。
5. 将模型用于预测。

图片被随机地划分成多个batch （32张图片一个batch），小batch会更快地收敛。10次迭代（每次都是完整数据集迭代）。

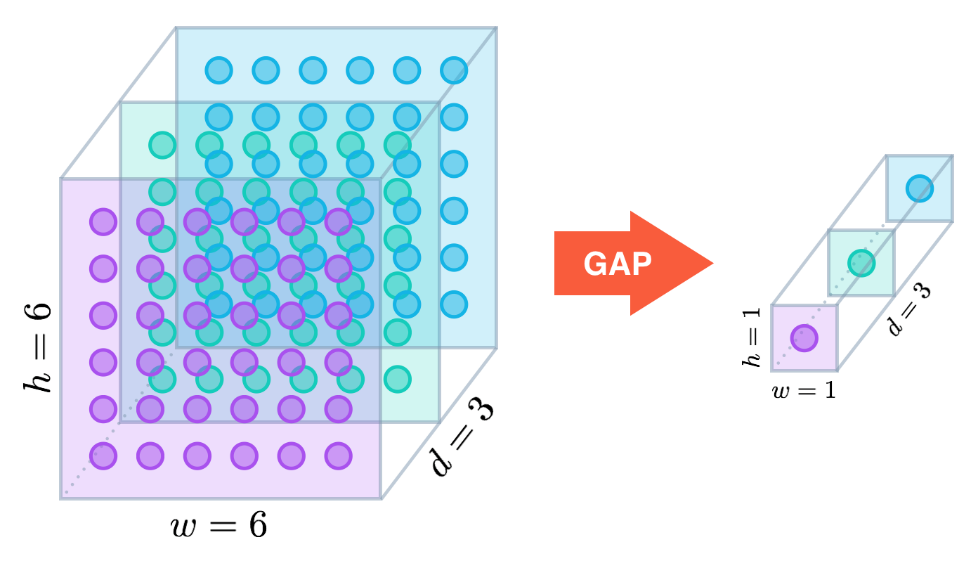
在迭代过程中，我们发现不同模型在使用笔记本电脑训练时都有一些问题。当卷积和全链接层有太多神经元时，或者图片的像素太高，python核心经常会崩溃，很可能是因为内存不足。

我把训练过程挪到AWS上，因为它能提供更大的RAM和更强的CPU。此后每次迭代缩短为130秒左右（虽然第一次迭代有点长）。这样整个训练过程只要20分钟。最后我得到的最好的训练集准确率是0.9479，验证集准确率是0.8936。这不差但是还有提升空间。当我把这样的模型应用在测试集后，得到的Log Loss分数是0.37276， 这在Kaggle上只能拍到890名左右。

完善

迁移学习是一种流行深度学习的方法，其中预训练模型被用作计算机视觉和自然语言处理任务的起点，它们帮助节省了开发神经网络模型所需的大量计算和时间资源，同时帮助我们处于解决相关问题的高起点上。

迁移学习有多种方法。其一是冻结所有预训练模型中的卷积层，用我们自己的分类器替换全链接层。我使用的预训练模型是VGG16（22层）和ResNet50（178层）。两个模型都被冻结了所有卷积层，然后我添加了一个全局平均池化层（GAP）。和最大池化层类似，GAP用于三维张量的空间维度。然而GAP提供了一种更加激进的维度减少方式：一个 维度的张量被缩减为 维度。GAP层将每个 特征地图缩减为一个单点通过求取所有 值的平均。[7]



然后我加上一个丢弃率为0.5的Dropout层。这意味着一半的输入单元将会被丢弃。好处是减少可训练参数的个数从而减少训练时间。最后在Dense层，我使用sigmoid函数作为激励函数。

下面是当batch大小为16，迭代次数为16时的结果：

|  |  |  |
| --- | --- | --- |
|  | VGG16 | ResNet50 |
| 全部参数个数 | 14,715,201 | 23,589,761 |
| 可训练参数个数 | 513 | 2,049 |
| 非训练参数个数 | 14,714,688 | 23,587,712 |
| 每次迭代平均时间 | 484 s | 296 s |
| 训练集损失 | 0.2120 | 0.0868 |
| 训练准确度 | 0.9415 | 0.9657 |
| 验证集损失 | 0.0973 | 0.0566 |
| 验证集准确度 | 0.9680 | 0.9772 |

另一种方法是在上面的模型上继续精细调优（Fine Tune）最后几层卷积层。这里我使用ResNet50作为预训练模型。因为多层能组成一个部件（例如：卷积层+正规层+激励层），所以我需要同时冻结/解冻它们。我从最高层（输出层）开始直到最低层（输入层），每解冻一个部件，迭代5次。然后我会检查损失函数和准确度是否改善，如果是的话，继续训练。否则解冻更多层知道训练结果改善。然而，即使我解冻了全部35层，精细调优的改善任然不明显。训练集损失为0.081，准确度为0.9698；验证集损失为0.0496，准确度为0.9794。

最后一种方法是集成多个模型。先从不同模型计算得到特征向量然后合并这些向量。之后用这个合并后的特征向量训练分类器。这里的网络结构可以很简单，但能达到很高的准确度。

1. 结果

模型验证和评价

我们使用的图片的宽，高和宽高比好到使我们的算法能侦测到有趣的模式，同时它们又足够的小能使算法不用处理太多信息以至于崩溃。

我所创建的模型可以用来区分图片中的猫或狗。我的最终模型（合并多个模型的特征向量后添加分类器得到的）在验证集上可以达到损失为0.0340，准确度为0.9888的成绩。

将原始数据集拆分成训练和验证集给予了我们合适的方法来评估模型。如果只是用训练集可能会误导我们因为这样的模型可能无法预测没有见过的图片。我们通过验证集确认模型是否过拟合或者足够宽泛化。

通过不同的方法：手工创建卷积神经网络，使用迁移学习（冻结，精细调优，特征向量抽取），验证集损失不断下降同时准确度不断上升。

合理性分析

准确度为0.9888对于模型来说很好，但是它还不是完全可信的，因为没有被测试集验证过。我使用最终模型来预测测试集的每张图片包含的是猫或狗，并将结果保存为csv文件。因为Kaggle的评分系统会调整概率区间为10^-15 到 1 - 10^-15，我将输出结果0/1调整为0.005/0.995， 当结果接近0或1时。当我上传结果到公告板上，我能看到成绩（Log Loss）是0.04944， 这相当于当前第57名。 这已达到了我的目标（进入前100名）。

项目总结

这是我第一个使用多种深度学习方法解决实际问题的项目。我学会了如何将理论应用于实际操作上。

在技术方面，我通过一系列数据探索，操作和可视化练习了使用python及其相关package比如numpy, matplotlib, opencv, tqdm还有keras。虽然在此项目中没有直接使用Tensorflow api， 我任然通过阅读Keras文档理解了Keras如何调用Tensorflow作为后台计算引擎。更重要的是，我学会了完整的操作流程：从导入数据，改变图像大小，一层层创建模型结构，训练模型和评估模型。

通过选择不同参数（例如：过滤器尺寸，横移大小，优化器种类）和优化模型结构帮助我理解了每一层在神经网络中的功能。在训练过程中，我注意到了过/欠拟合的发生。我面临各种折中选择，也看到了通过不同评估标准分析改善模型准确度的方法。

至于将来可能的改善方法，合并更多模型的特征向量或者一些其他手段，例如图像增强（调整图像位置）可能会有帮助。另外，我还可以尝试更宽，更深的网络模型，因为它能学习更多的图像特征。这可能会给我带来更好的结果但会花费更多训练时间。

参考资料

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

[2] <https://en.wikipedia.org/wiki/Convolutional_neural_network>

[3] <https://arxiv.org/abs/1409.1556>

[4] <https://arxiv.org/abs/1512.03385>

[5] <https://arxiv.org/abs/1512.00567>

[6] https://arxiv.org/abs/1502.03167

[7] <https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/>