**机器学习纳米学位毕业论文**

**猫狗大战**

**刘珅珅**

**2019年4月14号**

1 问题的定义

* 1. 项目概述

本项目来源于kaggle竞赛dogs-vs-cats比赛[1]，是通过机器学习的方式来识别出图片是猫还是狗，这是一个图像分类问题。选择这个项目作为毕业项目的原因是本人对计算机图像处理感兴趣，并且项目本身会用到深度学习和卷积神经网络的知识，这也是目前人工智能的热点。

* 1. 问题陈述

该项目是一个监督学习的二分类问题，训练集通过图片名称已经打好了标签，区分是猫还是狗，通过机器学习构建模型并训练，然后预测测试集中的图片是猫还是狗。

* 1. 评价指标

在kaggle竞赛中，该项目采用的评价指标为LogLoss，即对数损失。本项目采用与之相同的评价指标。

对于一个二分类问题，LogLoss的计算公式如下所示：



其中为测试集样本数，代表图片的真实标签，如果是狗，否则。表示模型预测图片为狗的概率。

从LogLoss的计算公式可以看出，模型不仅需要预测正确，而且预测的概率要尽可能的大，即对预测结果非常肯定，否则LogLoss的值就会较大。

2 分析

2.1 数据可视化

本项目的数据集为kaggle提供的数据集，训练集为25000张图片，其中12500张为猫类图片，12500张为狗类图片。测试集为12500张猫狗图片，通过名称来区分类别。

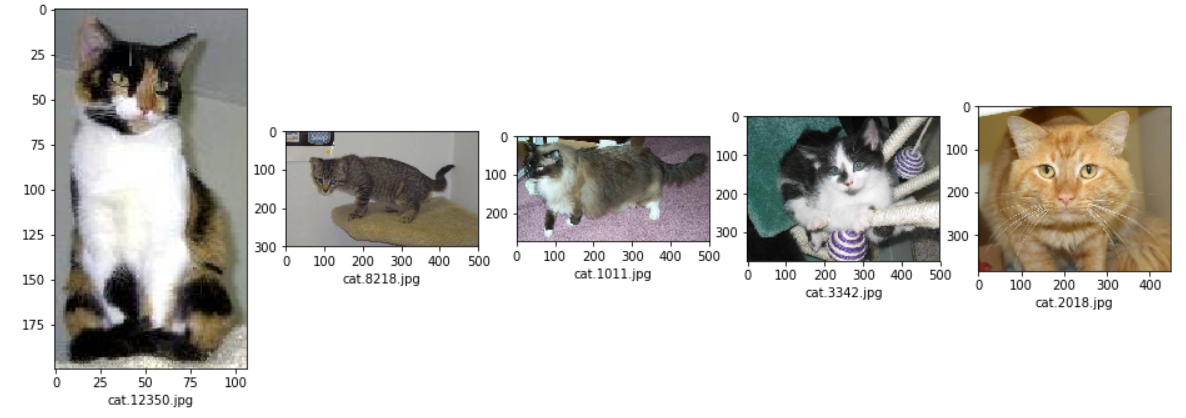


图2-1 训练集中猫类示例图

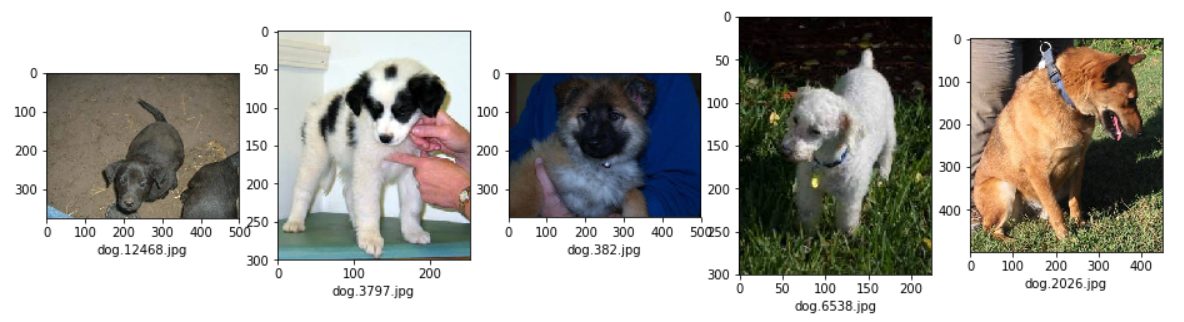


图2-2 训练集中狗类示例图

测试集总共有12500张图片，没有分类。

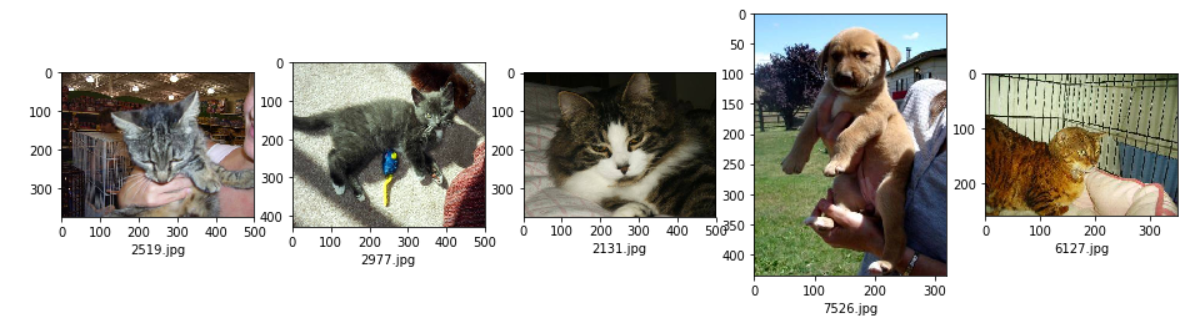


图2-3 测试集示例图

从上图可以看出，大部分图片中猫和狗的特征都非常明显，清晰度和分辨率都良好。另外根据<https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/discussion/47074#latest-479615>描述的情况，图片中存在标识错误的情况，有部分图片与标识不匹配，鉴于这样的图片数目比较少，采用手动删除的方式来剔除掉这些错误的数据。猫类图片总共删除了cat.3216.jpg，cat.7377.jpg，cat.8456.jpg，cat.7564.jpg，cat.9171.jpg，cat.4688.jpg，cat.4085.jpg，cat.5351.jpg，cat.5418.jpg，cat.11184.jpg，cat.10029.jpg。狗类图片总共删除了dog.1043.jpg，dog.1773.jpg，dog.4367.jpg，dog.8736.jpg，dog.8898.jpg，dog.10237.jpg。

2.2 算法和技术

过去几年中，卷积神经网络(Convolutional Neural Network, CNN)在图像分类中有很好的效果，考虑到数据集较大并和ImageNet的图片集有很大的相似性以及Udacity课程中深度神经网络有关迁移学习的介绍，可以采用迁移学习[1]的方式来解决问题。将问题分为两部分：特征提取和模型分类。特征提取是采用预训练模型提取实际图片的边缘特征、轮廓特征以及更高级的能区分猫狗的特征等。模型分类是利用之前提取的特征，输入到一个模型中进行分类训练和预测，并根据训练过程中的参数如准确率、损失大小等来判断是否满足要求来调整模型参数，最终得到符合要求的分类结果。

CNN的研究开始于20世纪90年代。1998年，Lecun等人提出了LeNet-5的卷积网络结构[2]，在字符识别应用中取得良好的效果。随着硬件技术尤其是GPU的快速发展，一些深层的CNN网络相继被提出，2012年由Krizhevsky等人提出的AlexNet模型[3]在ImageNet的图像分类竞赛中获得了冠军，之后牛津大学提出了VGG模型[4]，谷歌提出了GooLeNet模型[5]，微软提出了ResNet模型[6]，[Christian Szegedy](https://arxiv.org/search/cs?searchtype=author&query=Szegedy%2C+C)等人提出了Inception V3模型[7]，Francois Chollet提出了Xception模型[8]。

CNN通常包含以下2层：

1. 卷积层（Convolutional Layer）
2. 池化层（Pooling Layer）

2.2.1卷积层(特征提取层)

传统神经网络如果输入图片是1000x1000，则需要10^6个输入单元，如果隐藏层也有10^6个结点，则权重系数就会有10^12个，根本无法计算。

1. 局部感知

卷积层采用局部感知的方法来解决这一问题，每个隐藏层的结点都只与部分输入结点相连。

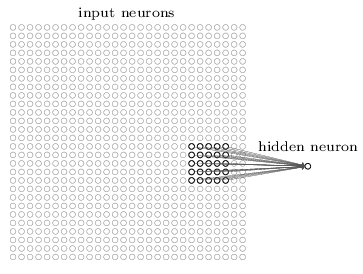


图2-4 卷积层示意图

具体的方式是采用一个filter或kernel对像素矩阵进行卷积来实现。我们移动这样一个filter使其扫描输入矩阵，每次移动1步(stride)都有一个不同的隐藏层结点与之对应。

第1个：

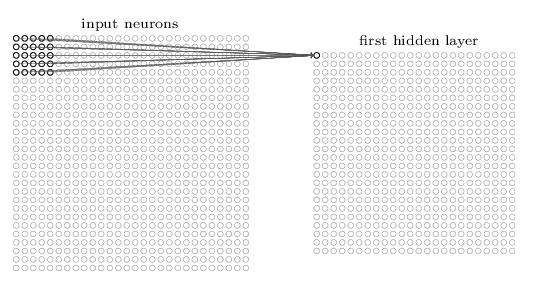


图2-5 卷积窗口(filter)移动示意图

第2个：

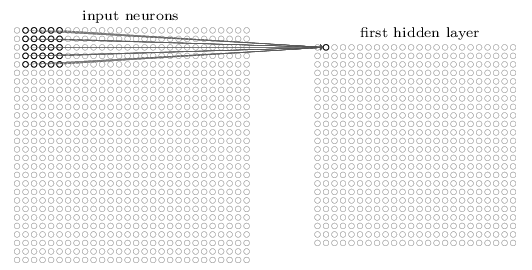


图2-6 卷积窗口(filter)移动示意图

以此类推，可以形成第1个隐藏层，如上图所示，输入为28x28，filter为5x5，得到的隐藏层为24x24(24=28-5+1)。也可以设置每次移动多步。

1. 权值共享

经过局部感知后，隐藏层有24\*24个结点，每个结点对应5\*5个权重系数。



其中代表输入层的矩阵，代表filter中的参数，表示激活函数，一般是Sigmoid或ReLU等，公式表示隐藏层第j行第k列的结点的输入。

虽然比全连接有所降低，但仍然很多，所谓权值共享就是认为隐藏层24\*24个对应的5\*5的权重系数和偏移值都是相同的。这样隐藏层的参数就变成了5\*5.这样认为的原因是：我们认为隐藏层从图像某一部分提取出来的特征也适用于另一部分，这也意味着一个隐藏层只能提取出一种特征。

一般CNN都会有多个卷积核(filter)。例，如果使用3个filter:

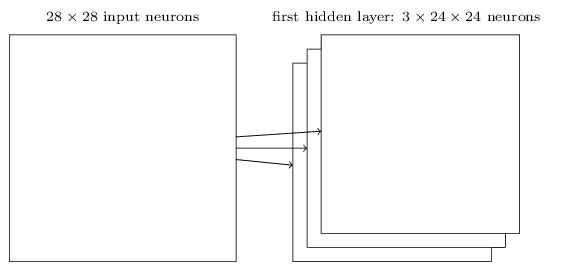


图2-7 多个卷积核

2.2.2 池化层

在卷积层后添加一个池化层，用来简化卷积层的输出，也使用一个filter，但不进行卷积操作，而是通过最大值，最小值、平均值等来进行。

例，2\*2的max filter，选4个参数中最大的一个，从而压缩了卷积层的输出。

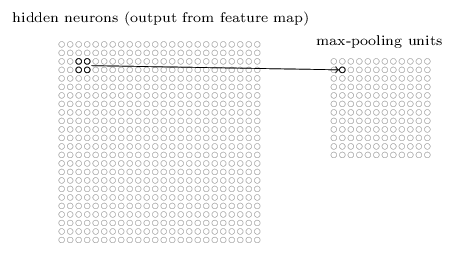


图2-8 最大池化层示意图

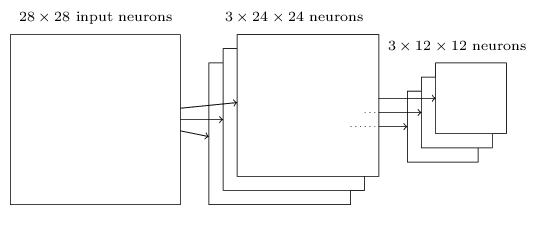


图2-9 卷积层+最大池化层示意图

经过池化层后，会大大减少我们学到的特征。

2.2.3 技术实现

本项目采用基于Tensorflow的开源机器学习库Keras[9]来实现模型的构建、训练和预测。数据集本身包含较多的图片，从头开始构建一个卷积神经网络并进行参数优化非常复杂耗时，在ImageNet中，包含通过猫狗图片进行训练并将参数优化的预训练CNN模型，所以可以采用迁移学习(Tranfer Learning)的方式来进行模型构建。

迁移学习就是将已经预训练CNN模型的参数迁移到新的模型上已提高模型的训练速度和效果。在本项目中使用的训练集与之前ImageNet训练模型的数据集有很高的相似度，所以可以用预训练模型的卷积层参数来进行特征提取，才添加自己的全连接输出层进行分类识别。

在Keras中有5种著名的CNN模型，分别为VGG16，VGG19，ResNet50，InceptionV3，Xception。本项目将分别使用这5种模型进行特征提取，然后对比它们的训练效果，选取最优的模型。

2.3 基准指标

毕业项目要求模型的LogLoss必须在kaggle pubic learnboard中排入前10%，即LogLoss<0.06127。考虑到可以对模型进行优化调整，基准模型的LogLoss可以稍大于这个值，所以设置基准指标LogLoss为0.08.

3 方法

3.1 数据预处理

1) kaggle中提供的训练集图片名称以cat和dog开头，统一放在train文件夹下，所以先将其分类放在cat和dog文件夹。另外需要分出一部分数据作为验证集用于交叉验证，本项目将训练集数据的20%作为验证集，存放在valid文件夹的cat和dog文件夹下。之前讨论过要删除异常数据，删除后训练集猫类图片有12489张，狗类图片有12494张，总共有24983张图片，最终train文件夹中的cat和dog文件夹总共有图片19988张，valid文件夹下的cat和dog文件夹总共有图片4995张，测试集test有图片12500张。另外考虑到本项目采用神经网络迁移学习来进行，所以创建一个小批量数据集来进行参数的调整，随机抽取并复制猫类图片800张和狗类图片800张到small\_train文件夹下的cat和dog文件夹中，随机抽取并复制猫类图片200张和狗类图片200张到small\_valid文件夹下的cat和dog文件夹中，从测试集随机抽取500张图片到small\_test文件夹的test文件夹中，这样就构建了一个小批量数据集。

2) 本项目采用预训练模型VGG16，VGG19，ResNet50，InceptionV3，Xception进行特征提取，这几种模型对图片的预处理要求有所不同。根据keras.applications的源代码可以看出，VGG16，VGG19，ResNet50的预处理方式是相同的，需要图片每个通道都减去ImageNet的mean值[103.939, 116.779, 123.68]，并将图片从RGB转换为BGR；InceptionV3，Xception的预处理方式是相同的，将图片数据缩放到[-1, 1]之间。

3) 本项目使用keras中的ImageDataGenerator类通过实时数据增强批量将图片转换为张量图像数据，其中VGG16，VGG19，ResNet50默认的图像尺寸为(224, 224)，InceptionV3，Xception默认的图像尺寸为(299, 299)。

3.2 特征提取

分别创建上述5种预训练模型用于特征提取，创建时需要将模型的include\_top=False，这样就不包含预训练模型最后的全连接层，只通过卷积层来提取特征。在使用小批量数据集（训练集1600张，验证集400张）进行测试时，发现输出的特征数据文件非常大，超过了1G，因此统一将模型的pooling=’avg’，即在卷积层的最后添加一个GlobalAveragePooling2D层，用于减少特征量，防止过拟合。

3.3 模型分类构建与训练

特征提取完成后，使用特征数据作为输入数据，仿照预训练模型构建3个全连接层的模型进行训练，模型信息如下图所示。

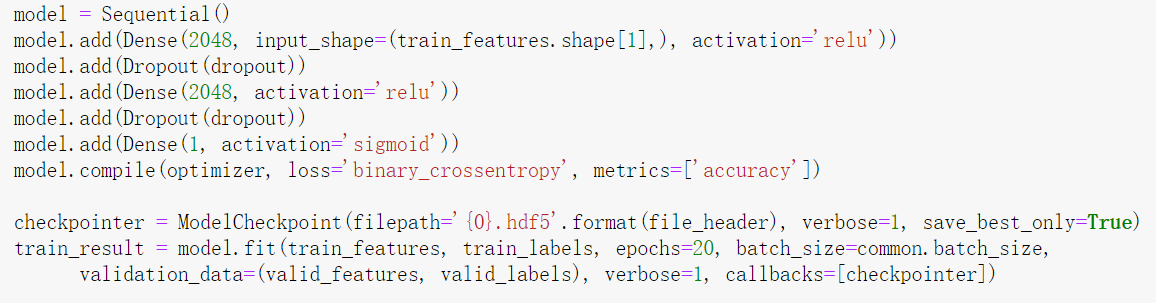


图3-1 分类模型源代码

针对5种特征提取模型，分类模型架构几乎完全一致，各个模型的输入参数input\_shape不同，VGG16和VGG19的input\_shape为512，ResNet50，InceptionV3和Xception的input\_shape为2048.

卷积层输出的特征向量维度为2048，分类模型添加两层全连接层结点数也设置为2048，激活函数使用ReLU函数，并添加Dropout层用于防止过拟合。由于是二分类问题，输出层结点为1，激活函数使用Sigmoid函数，损失函数为二分类的交叉熵，训练期间评估指标设置为准确率。

3.4 遇到的问题及模型参数优化

1) 预处理函数的修改。之前根据Udacity课程讲解及参考网上资料，自己写预处理函数，在ImageDataGenerator函数中添加了图片的平移、旋转和随机缩放等步骤，在小批量的数据集（训练集1600张，测试集400张）发现效果并不理想，模型的训练损失和验证损失都比较高，模型不收敛，后来之间使用VGG16等5种模型自带的预处理函数，发现效果比较理想，推测是由于数据集与ImageNet有较大的相似性，使用自带的预处理函数，效果反而更好。

2) dropout参数和优化器的选择。在图像分类中，常用的优化器有RMSprop，Adam和SGD等。最初优化器选择的是RMSprop，在小批量数据集训练时，模型很难收敛，增加训练次数训练损失和验证损失几乎没有改进，因此不采用这种优化器。之后在小批量数据中尝试了SGD和Adam作为优化器，并选择不同的dropout参数，进行对比进行参数优化。

VGG16：

表3-1 VGG16 LogLoss曲线对比图

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 |  |  |
| 0.5 |  |  |
| 0.7 |  |  |

从表格3-1对比图中，VGG16作为预训练模型，选择dropout=0.5，optimizer=SGD(lr=0.001)

VGG19：

表3-2 VGG19 LogLoss曲线对比图

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 |  |  |
| 0.5 |  |  |
| 0.7 |  |  |

从表格3-2对比图中，VGG19作为预训练模型，选择dropout=0.5，optimizer=SGD(lr=0.001).

ResNet50：

表3-3 ResNet50 LogLoss曲线对比图

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 |  |  |
| 0.5 |  |  |
| 0.7 |  |  |

从表格3-3对比图中，ResNet50作为预训练模型，选择dropout=0.5，optimizer=SGD(lr=0.001).

InceptionV3：

表3-4 InceptionV3 LogLoss曲线对比图

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 |  |  |
| 0.5 |  |  |
| 0.7 |  |  |

从表格3-4对比图中，InceptionV3作为预训练模型，选择dropout=0.5，optimizer=Adam(lr=0.001).

Xception：

表3-5 Xception LogLoss曲线对比图

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 |  |  |
| 0.5 |  |  |
| 0.7 |  |  |

从表格3-5对比图中，Xception作为预训练模型，选择dropout=0.3，optimizer=Adam(lr=0.001).

4 结果

4.1 模型评价与验证

使用VGG16，VGG19，ResNet50，InceptionV3和Xception5种模型进行特征提取，然后在自建分类模型上训练得到的结果如下：

1. VGG16的训练准确率和验证准确率均超过0.98。
2. VGG19的训练准确率超过和验证准确率均超过0.98.
3. ResNet50的训练准确率和验证准确率均超过0.99.
4. InceptionV3的训练准确率和验证准确率均超过0.99.
5. Xception的训练准确率和验证准确率均超过0.99.

4.2 合理性分析

将5种模型训练的最优参数保存下来然后载入到分类模型中，对测试数据集进行预测并将结果上传到kaggle，得到的结果如下图所示。

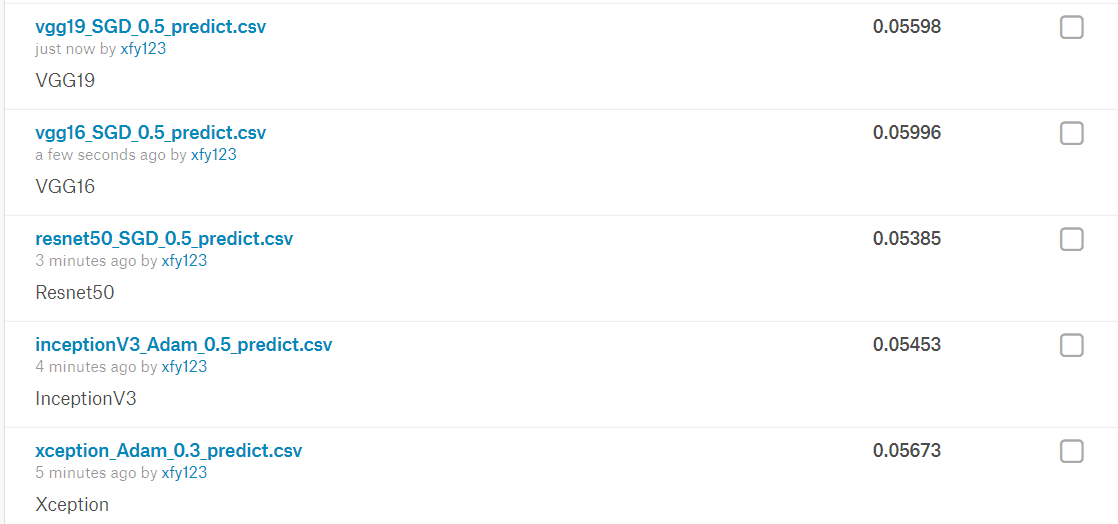


图4-1 kaggle评分结果图

由上图可以看出，5种模型均满足LogLoss<0.06127，符合项目设置的要求，其中ResNet50的效果最好。

5 项目结论

5.1 结果可视化

ResNet50模型的准确率与LogLoss的曲线图：

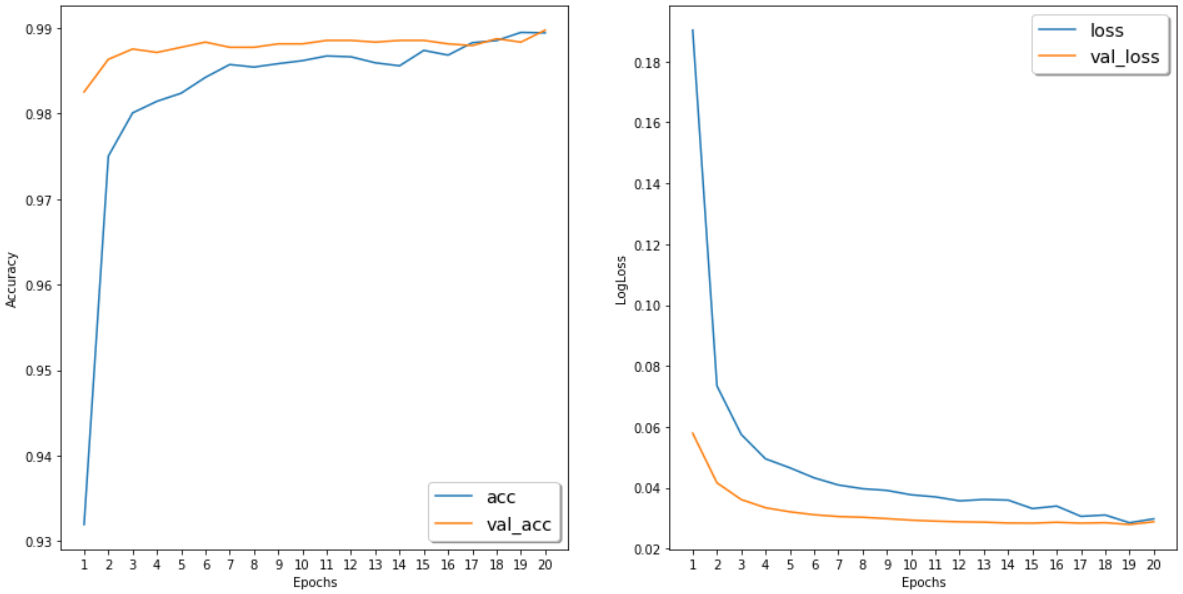


图5-1 ResNet50模型的训练准确率和LogLoss的对比图

5.2 对项目的思考

本次项目，从数据预处理到模型选择，再到参数的调整都完整的经历了一遍，我有以下收获：

1) 从头在阿里云上搭建了整个训练环境，中间经历了安装Tensorflow失败，Keras版本不匹配的错误，代码也爆出了各种与环境配置相关的错误，最终都成功解决。

2) 在迁移学习中，对于不同的数据集，要多选择几种模型进行比较操作，这样可以选择出最优的模型，没有那个模型可以适合所有类型的数据集。

3) 对优化器的选择一定要多尝试几种不同的情况，之前根据Udacity中项目的经验选择的RMSprop由于结果不收敛，一度让我怀疑迁移学习是否能成功，在以后的机器学习项目中，针对优化器要多选几种进行比较，也没有那一种优化器可以适合所有的项目。

5.3 改进措施

本项目模型的最终得分满足了项目要求，但仍然有提高的空间。

1. 可以尝试多种预训练模型的融合操作，参考网上的资料，ResNet50，InceptionV3和Xception的融合得到的LogLoss可以减小掉0.0414.
2. 可以尝试预训练模型的微调来提高准确率。

参考文献

[1] 庄福振，罗平，何清，等迁移学习研究进展[J]. 软件学报，2015,26(1):26-39

[2] Y. Lecun, L. Bottou, Y.Bengio, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998,86(11): 2278-2324

[3] A. Krizhevsky, I. Sutskever, G. E. Hinton. ImageNet classification with deep convolutional neural networks[C]. Proceedings of Adavances in Neural Information Processing System, Lake Tahoe, 2012, 1106-1114

[4] Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition[EB/OL] <https://arxiv.org/abs/1409.1556>, April 10, 2015

[5] C. Szegedy, W. Liu, Y. Jia, et al. Going deeper with convolutions[C]. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Boston, 2015, 1-8

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition[EB/OL] <https://arxiv.org/abs/1512.03385>, December 10, 2015

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna Rethinking the Inception Architecture for Computer Vision[EB/OL] <https://arxiv.org/abs/1512.00567>, December 2, 2015

[8] François Chollet Xception: Deep Learning with Depthwise Separable Convolutions[EB/OL] <https://arxiv.org/abs/1610.02357>, October 7, 2016

[9] Keras Documentation[Z] <https://keras.io/>