

机器学习工程师- 猫狗大战

韩旭 优达学城

July 8, 2019

1 问题的定义

1.1 项目概述

猫狗大战是2013年kaggle上的一个竞赛题目。目的是通过机器学习或者深度学习技术搭建可以识别图片并且能够正确对猫和狗进行分类的模型，这是一个属于计算机视觉领域二分类的模型。

深度学习技术是近几年兴起的技术，也是这些年人工智能领域取得的重大成就之一。现在已经在搜索引擎、机器翻译、图像识别、自然语言处理、无人驾驶这几个域逐步趋向成熟。

自从Alex和他的导师Hinton在2012年的ImageNet大规模图像识别竞赛中以超过第二名10个百分点的成绩(83.6%的Top5精度)碾压第二名(74.2%，使用传统的计算机视觉方法)后，深度学习真正开始火热，卷积神经网络开始成为家喻户晓的名字，从12年的AlexNet(83.6%)，到2013年ImageNet大规模图像识别竞赛冠军的88.8%，再到2014年VGG的92.7%和同年的GoogLeNet的93.3%，终于，到了2015年，在1000类的图像识别中，微软提出的残差网(ResNet)以96.43%的Top5正确率，达到了超过人类的水平。

伴随着图像分类任务，还有另外一个更加有挑战的任务 - 图像检测，图像检测是指在分类图像的同时把物体用矩形框给圈起来。从14年到16年，先后涌现出R-CNN, Fast R-CNN, Faster R-CNN, YOLO, SSD等知名框架，其检测平均精度，在计算机视觉一个知名数据集上PASCAL VOC上的检测平均精度，也从R-CNN的53.3%，到Fast RCNN的68.4%，再到Faster R-CNN的75.9%，最新实验显示，Faster RCNN结合残差网(Resnet-101)，其检测精度可以达到83.8%。深度学习检测速度也越来越快，从最初的RCNN模型，处理一张图片要用2秒多，到Faster RCNN的198毫秒/张，再到YOLO的155帧/秒(其缺陷是精度较低，只有52.7%)，最后出来了精度和速度都较高的SSD，精度75.1%，速度23帧/秒。

1.2 问题陈述

这个项目需要识别出猫和狗，从本质上讲是一个二分类的问题，通过搭建卷积神经网络训练模型，先对图像进行打标签，然后使用神经网络进行训练，通过给定任何一张图片，达到识别出猫和狗的目的。

整个流程如下：

1.2.1 数据预处理

- 从kaggle官方下载数据
- 使用opencv对图片进行读取，裁剪成299*299的尺寸方便后续进行训练

1.2.2 模型搭建

对keras里面的预训练模型进行模型融合，对原有的数据集的权重进行优化，然后搭建最后的Dense层进行预测。

- 使用preprocess_input函数对输入图片进行预测处理
- 对InceptionV3和Xception进行模型融合
- 在搭建好的模型后边搭建自己全链接层

1.3 评价指标

kaggle官方采用对数损失来衡量：

$$\text{LogLoss} = -\frac{1}{n} \sum_i^n [y_i \log(1 - y_i) + (1 - y_i) \log(y_i)]$$

其中：

- n是图片数量
- \hat{y}_i 是模型预测为狗的概率
- y_i 是类别标签，1 对应狗，0 对应猫
- $\log()$ 表示自然对数

LogLoss越小的模型也就更加的鲁棒，所以LogLoss 可以用来评估这个项目。

2 分析

2.1 探索性可视化

kaggle提供的文件含了两个数据集，test.zip and train.zip。train.zip里面包含了12500猫的照片和12500张狗的照片,每张照片的文件名中都包含有dog 或者cat 的标签。test.zip里有有12500张照片，文件名中没有标签。

这里是随机对图片进行读取：



Figure 1: 随机读取未处理的数据

刚刚的随机读取发现图片尺寸不一，有些图片还有遮挡物，还有些不是猫狗的图片混在一起了。尝试对数据集里的所有数据进行裁剪成299*299，来适应后边的神经网络模型。



Figure 2: resize之后的数据

2.2 算法和技术

2.2.1 深度学习、神经网络简介

深度学习

深度学习是机器学习的一个子领域，是一种以人工神经网络为基础，不断从错误中学习的一种计算机算法，今年来深度学习甚至独立于机器学习成为了一个全新的领域，主要用来处理线性不可分的问题。

深度学习和神经网络的概念最早可以追溯到上个世纪，最早来自于心理学和控制论的知识。随着时间的发展，逐渐诞生了从感知器为基础的卷积神经网络，循环神经网络，长短期神经网络这些技巧。随着alpha和李世石一战成名，深度学习越来越为大家所熟知，

而今天，深度学习在计算机视觉，自然语言处理，数据挖掘，无人驾驶，音视频算法等领域都有了广泛的应用。

人工神经网络

人工神经网络是深度学习的基础，从最早的感知器概念的提出到现在 CNN, RNN, LSTM 经历了很长一段时间。早的感知器的提出是基于动物神经元的模型，由此奠定了深度学习的基础。

神经网络由众多个神经元组成，大致上可以分为三个部分，第一层的神经网络叫做输入层，最后一层的网络叫做输出层，中间的部分是最复杂的，通常把它看作一个黑箱，这个就是隐藏层。通常来说，每一个神经元在参与训练的时候有一个权重，而训练神经网络的目的就是寻找合适的神经元权重，来提高神经网络对回归和分类的泛化能力。这个这些权重经过这层计算之后又会传导到下一层神经元，这就类似生物体的神经冲动。这个时候往往还有一些激活函数，常见的激活函数及其功能如下：

激活函数	功能
sigmoid	用在二分类问题中，预测正负样本的概率
softmax	用于多分类问题的概率预测
relu	用来加快训练速度，防止梯度消失
tanh	把sigmoid的图像缩放到 (-1, 1) 之间
softplus	relu的平滑，比神经元更接近于脑神经元的激活模型

神经元

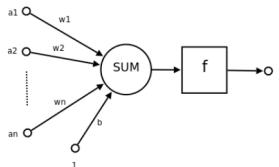


Figure 3: 单个神经元模型

- $a_1 \sim a_n$ 为输入向量的各个分量
- $w_1 \sim w_n$ 为神经元各个突触的权值

- b为偏置
- f为激活函数，通常为非线性函数。一般有relu, sigmoid, softmax, tanh等
- t为神经元输出

数学表示为: $t = f(W^T * A + b)$

- W^T 代表权重
- A代表输入
- b为偏置
- f为激活函数
- t为神经元输出

单层神经元网络

单层神经网络是最简的神经网络模型，有且仅有一层神经元组成，每个的神经元和输入向量的每个元素都有一一对应的关闭，按照这个方法来说，单层神经元网络的输入向量的维数和网络的维数相同。

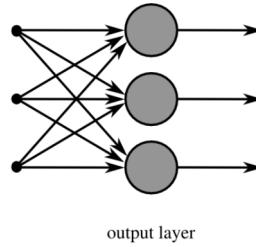


Figure 4: 单个神经元模型

多层神经元网络

一种常见的多层结构的前馈网络由三部分组成。

- 输入层：神经元接受来自样本的数据（图片或者数据），这个被称之为输入向量。
- 输出层：输出神经网络经过计算返回的结果。输出的消息称为输出向量。
- 隐藏层：通常有很多层组成，神经网络中间的部分，中小型神经网络中，隐藏层的数量常常有几十万到几百万不等。

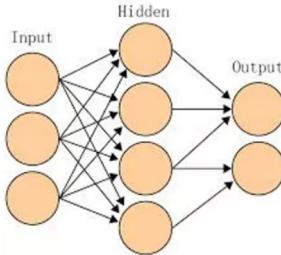


Figure 5: 神经网络基本结构

2.2.2 卷积神经网络原理

卷积神经网络相对传统神经网络来说，多出了一个或者多个的卷积层，同时也在网络中添加了池化层来把网络适应输入的二维数据。利用卷积核可以图片的特征来进行提取。

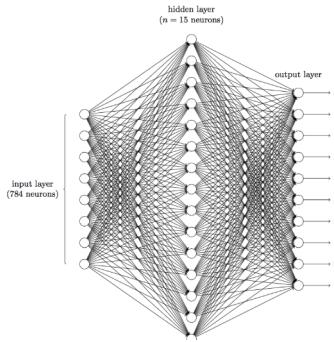


Figure 6: 卷积神经网络模型

卷积层(Convolutional Layer)

卷积层由若干个卷积算子组成，主要用于提取图片的特征，而卷积有层层递进的效果，比如第一次提取边缘，第二次提取稍微清晰的特征，每一个提取的特征都会比上一次的特征更加鲜明。

目标检测中常常把全连接层换成卷积层，因为全连接层的权重是不变的，所以输入的图片大小不能变。而卷积层的感受野是不断变化的，随着卷积核的移动来获取不同位置对应的特征信息，这样就突破了输入尺寸的限制，就获得了目标的位置信息。可以高效地对测试图像做滑动窗式的预测。可以高效的检测多个目标和给出位置信息。

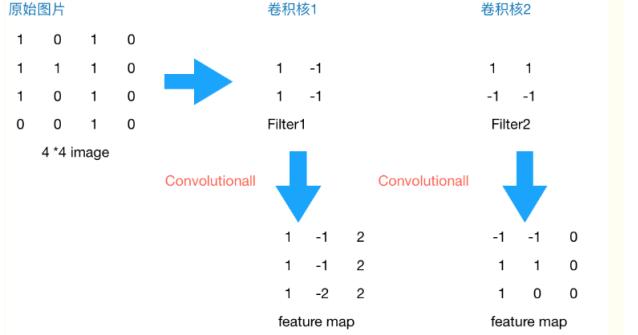


Figure 7: 卷积层

池化层(Pooling Layer)

池化层是一种有效防止过拟合的方法，大致分为全局池化，最大池化和平均池化。最大池化取消滑动窗口元素的最大值，平均池化取消滑动窗口元素的平均值。为全局池化的窗口和这个整个视野窗口一样大，用于改变网络的维度。随身近些年来神经网络的发展，最大池化的应用已经越来越广泛。经过池化层会减小图片的空间大小，进而把重要的特征保留出来，这样一来就有效的降低了神经网络中参数的个数，在一定程度上防止了过拟合网络的产生。

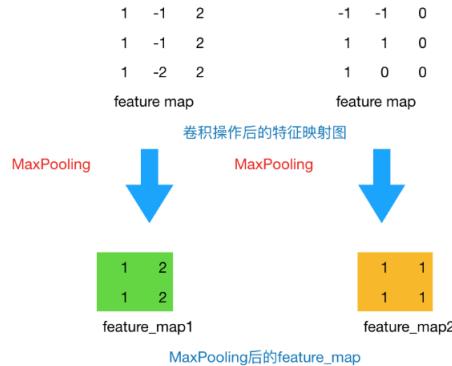


Figure 8: 池化层

Dropout层

大型的深度神经网络的深度往往很大，模型参数的数量也很多。然而过拟合往往是神经网络经常遇到的一些问题，神经网络中常常会用到防止过拟合的方法，Dropout就是其中之一。Dropout的主要思想就是在训练神经网络的过程中随机丢弃一些神经元，测试阶段，用训练好的网络对所有的测试样本进行前向操作.如softmax的实际输出值与理想输出值一致。减弱了相邻神经元之间的适应能力，提高了泛化能力。这就有效的降低了过拟合，而且相对其他正则化方法有了重大的改进。Dropout提高了基于监督学习的神经网络在视觉、语音识别和生物计算的准确率。

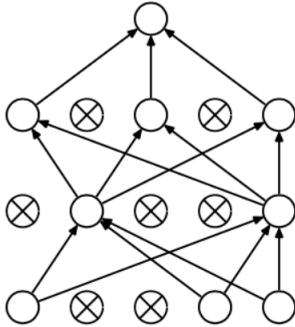


Figure 9: Dropout层

Dense层

Dense用来对上一层的神经元进行全连接，对神经元的权重和神经元实现特征的非线性组合。

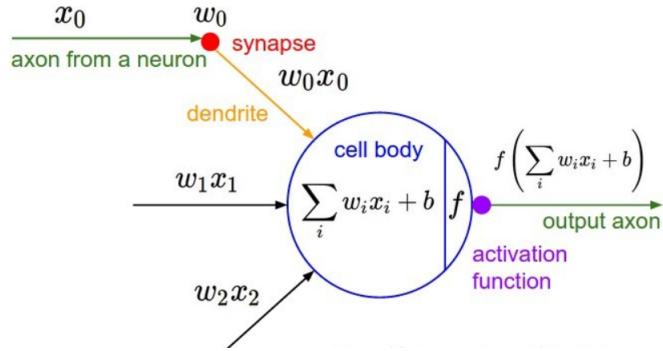


Figure 10: 带有激活函数的Dense层

2.2.3 迁移学习和Fine tune

迁移学习是属于深度学习的一个子领域。它会用别人已经训练好的权重对自己功能相似的神经网络。比如这次的神经网络的预训练，猫狗大战是一个二分类的问题，那么我们可以使用ImageNet上边多分类的预训练模型和进行迁移学习。

Fine-tuning和迁移学习大致相似，只不过fine-tuning会去掉网络的最后一层，然后加上自己的全连接层对网络结果进行输出。

预训练的模型花费时间更短，准确率更高。

VGG16

VGG16是由牛津大学提出的。亮点在于使用两个 3×3 的卷积核代替传统的 5×5 的卷积核。 3×3 卷积核有利于更好地保持图像性质。

ConvNet Configuration					
A 11 weight layers	A-LRN	B 13 weight layers	C 16 weight layers	D 16 weight layers	E 19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 11: VGG网络架构

Resnet50

ResNet主要使用3x3卷积，这点与VGG类似。在VGG基础上，短路连接插入进入形成残差网络。

利用残差模块，可以训练152层的残差网络。其准确度比VGG和GoogLeNet要高，但是计算效率也比VGG高。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer	
conv1	112×112			7×7, 64, stride 2			
				3×3 max pool, stride 2			
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$	
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$	
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	
	1×1			average pool, 1000-d fc, softmax			
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9	

Figure 12: ResNet几种网络

InceptionV3

从InceptionV2到InceptionV3，图片的输入从 224×244 变成了 299×299 。另外还基于Inception的结构设计了采用一种并行的降维结构。

相比Inception之前的版本，有如下更改：

- 我们将传统的 7×7 卷积分解为3个 3×3 卷积
- 对于网络的Inception部分，我们在 35×35 处有3个传统的Inception模块，每个模块有288个滤波器
- 使用第5节中描述的网格缩减技术，这将缩减为 17×17 的网格，具有768个滤波器
- 这之后是图5所示的5个分解的Inception模块实例。使用图10所示的网格缩减技术，这被缩减为 $8 \times 8 \times 1280$ 的网格

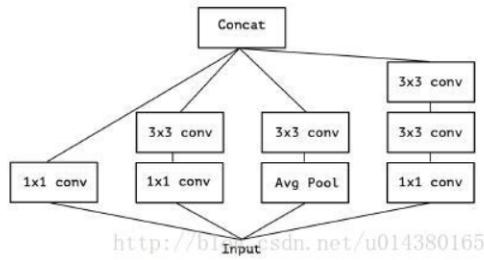


Figure 13: InceptionV3网络构架

Xception

Xception是google继Inception后提出的对Inception v3的另一种改进，主要是采用depthwise separable convolution来替换原来Inception v3中的卷积操作。达到了在减少参数量的情况下增加模型的层数，既减少了存储空间，还增强了模型的表达能力。

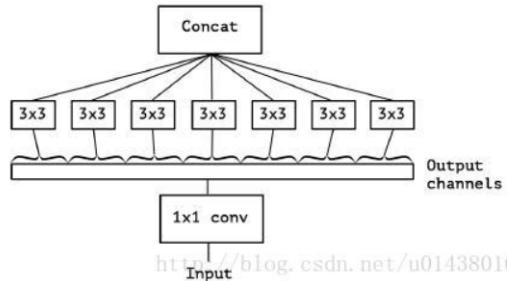


Figure 14: Xception网络

2.3 基准模型

根据项目的要求，需要在kaggle比赛上获得前10% 的成绩，使用Xception和InceptionV3网络模型去完成项目。这个比赛总共有1314只队伍参加，而此次毕业项目的要求是kaggle Public Leaderboard 前10%。所以logloss比第134名的0.06320 小就好。

3 方法

3.1 数据预处理

- 使用opencv读取图像
- 把图片裁剪成299*299 的大小来适应后续训练的模型
- 使用sk-learn对训练集和测试集进行4:1的拆分

3.2 执行过程

- 导入Xception和InceptionV3和模型
 1. 对连个模型进行简单的相加融合
 2. 在去掉了最后一层的融合模型之后搭建Dropout层，参数使用0.3，在Dropout层之后搭建一个神经元的全链接层，激活函数是使用sigmoid。
- 编译模型
 1. 优化器使用adam，二分类的loss使用binaryp_crossentropyp
 2. 训练模型，并且使用过早停止，实际上训练了10代
- 使用训练好的模型对测试集进行预测，提交到kaggle Leaderboard



Figure 15: 本次项目搭建的神经网络结构

3.3 总结与完善

一开始尝试了Resnet50和VGG16 去除最后一层然后冻结权重，自己搭建Dense层Dropout层，准确率可以达到98%左右，但是Logloss太高，达不到项目的要求。后来就采用了对Xception和InceptionV3进行融合的方式，在一开始使用了inception_v3 里面preprocess_input的预处理，keras中的preprocess_input()可以函数完成数据预处理的工作，对样本执行逐样本均值消减的归一化，即在每个维度上减去样本的均值，这样就可以能够提高算法的运行效果。也就是由于这样，不同模型的预处理函数不同，之前训练的时候使用自己的预处理函数，不能很好的发挥出来fine-tune的效果。由于Xception 是google 继Inception 后提出的对Inception v3 的另一种改进,所以二者可以使用一样的与处理函数。因为有了这个预处理方法，所以之前的图片通道转化是多余的，可以去掉。

这个模型的准确率最终达到了99% 以上，Logloss 也明显下降许多。其中Dropout 层冻结了0.3的神经元，Dense 层使用了sigmoid 来达到二分类的目的。在过早停止的过程中打算训练100 代，当有5代val_loss 不再下降的时候就停止训练，实际上只训练了10 代，val_loss 稳定在了0.02 左右。在这个基础上分别尝试四个不同的优化器：Adam、RMSprop、Adadelta、SGD，调整了不同的学习率，后来发现Adam的默认参数表现最好。加了BN 层之后虽然会加快模型收敛速度，但是在这里模型里发生了过拟合，所以最后去掉了BN 层。

四个优化器的Logloss表现结果如下：

优化器	Logloss
Adam	0.05560
RMSprop	0.5642
Adadelta	0.0605
SGD	0.05770

4 结果

最后发现还是Adam的表现效果最好，结果得到Logloss为0.05560，在kaggle 上排到96/1314的成绩，相当于Public Leaderboard7.3%，基本上达到了项目要求。

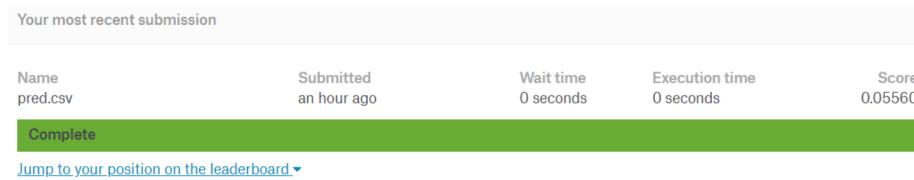


Figure 16: 在kaggle上提交的成绩

4.1 模型的评价与验证

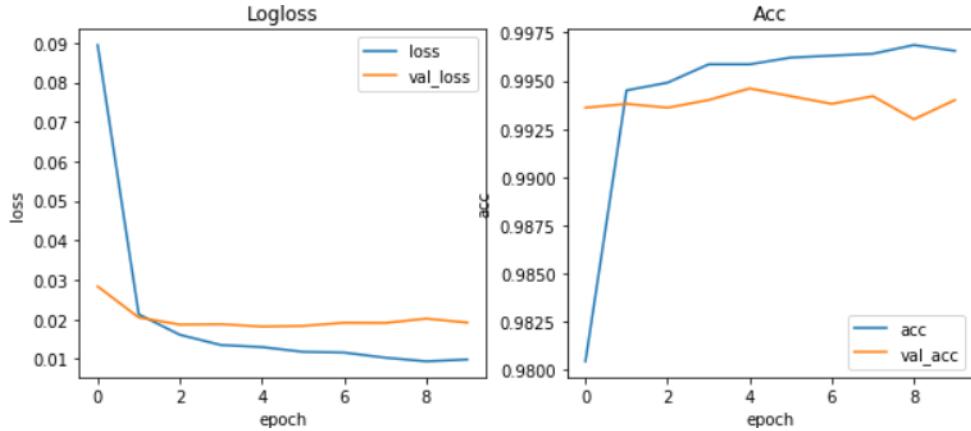


Figure 17: 在kaggle上提交的成绩

从图中可以看出来，利用融合模型的方法模型精度高出来很多，训练两次的时候训练集和验证集的准确度都可以达到99%以上，由于加入了过早停止，当val.loss不再下降的时候就不再训练了，而这里也就训练了10轮不到就已经满足训练要求了，再继续训练下去就会过拟合了。

5 项目结论

5.1 对项目的思考

一开始采用了自己搭建的CNN模型，精度远远不够，后来尝试了Resnet，不过是在本机上跑，训练速度很慢，这里浪费了好多时间，而reset50再没有做异常值处理和数据增强的话也没有办法达到项目的要求。后来多次学习，对本来就表现很好的Xception 和InceptionV3进行简单模型融合，这个时间也比之前的短了很多。在此学习到利用前人发明的轮子会比自己从头开始效率高很多。

5.2 需要作出的改进

这个项目存在许多需要改进的地方，可以采取一下几种方式：

1. 对数据进行异常摘除，应该会使结果好很多
2. 对预训练模型进行微调
3. 对数据集进行数据增强
4. 采用更多正则化方法来防止过拟合
5. 使用更好的预训练模型来搭建网络

参考文献

- [1] 维基百科深度学习[Z] <https://zh.wikipedia.org/wiki/>
- [2] 刘海洋.LATEX入门[M]. 电子工业出版社. 2018年4月
- [3] 杨培文.深度学习技术图像处理入门[M]. 清华大学出版社.2018 年10月: 134-154
- [4] Francois Chollet.Xception: Deep Learning with Depthwise Separable Convolutions[R]. Francois Chollet.7 Oct 2016
- [5] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition[R]. in International Conference on Learning Representations, May 2015.
- [6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision[R]. Christian Szegedy. 2 Dec 2015
- [7] Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting[R]. Journal of Machine Learning Research.15 Jun 2014.
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition[R]. Kaiming He. 10 Dec 2015
- [9] 范晓杰, 宣士斌, 唐凤. 基于Dropout卷积神经网络的行为识别[J]. 广西民族大学学报(自然科学版), 2017, 23(1): 76-82