

机器学习纳米学位报告 - 毕业项目

Rex 优达学城

2017年12月15日

问题的定义

项目概述

本项目源于State Farm公司发布在Kaggle上的竞赛（1），通过车内装置的摄像头来检测司机不安全驾驶的行为，如打电话、手机打字等。识别这些行为，将有利于对用户的驾驶行为进行评估，对不安全的行为予以警示，甚至通过调整保险价格的方式来激励用户安全驾驶。深度学习技术可以对图像进行分类识别，已经在工业上有广泛的应用，可以很好的解决上述问题。

问题描述

在Kaggle竞赛上，State Farm提供了一批已标注的在视频中截取的彩色图片数据，每个照片对应的标注为十种状态中的一种。

State Farm另提供了若干测试集数据，要求我们对输入数据进行分析，输出该照片对应十种状态的概率。



评估标准

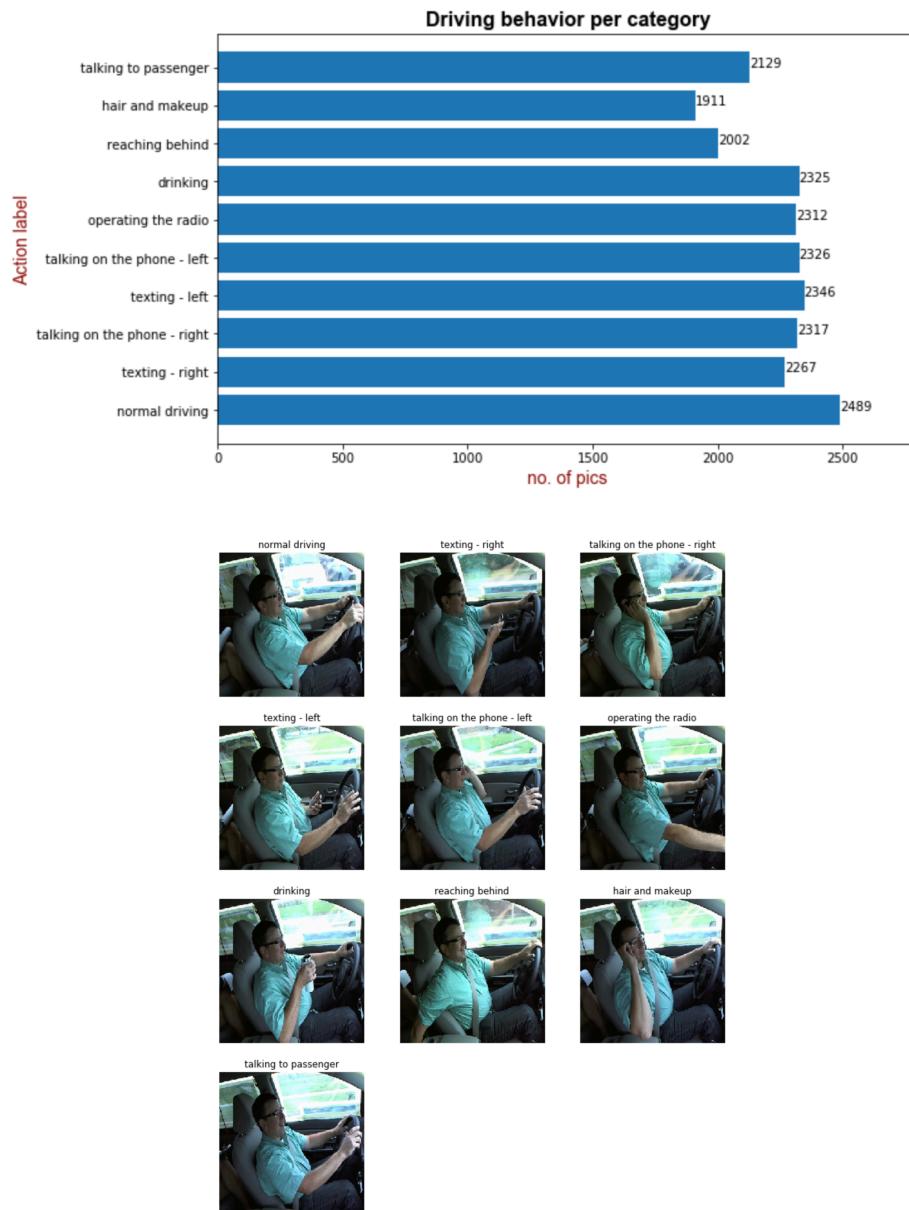
我们将沿用State Farm竞赛中所使用的评分标准：多分类对数损失（multi-class logarithmic loss） 其定义可参见(2)，其公式如下：

$$\text{logloss} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij}),$$

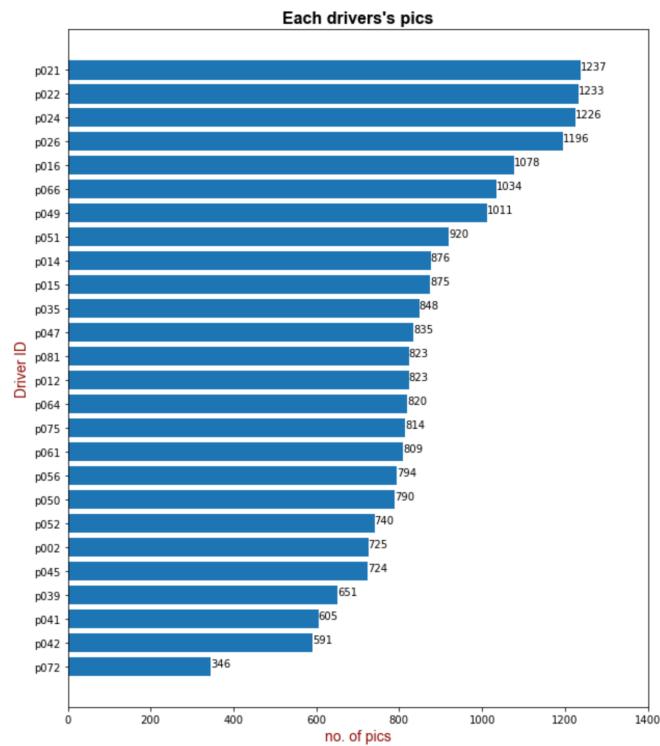
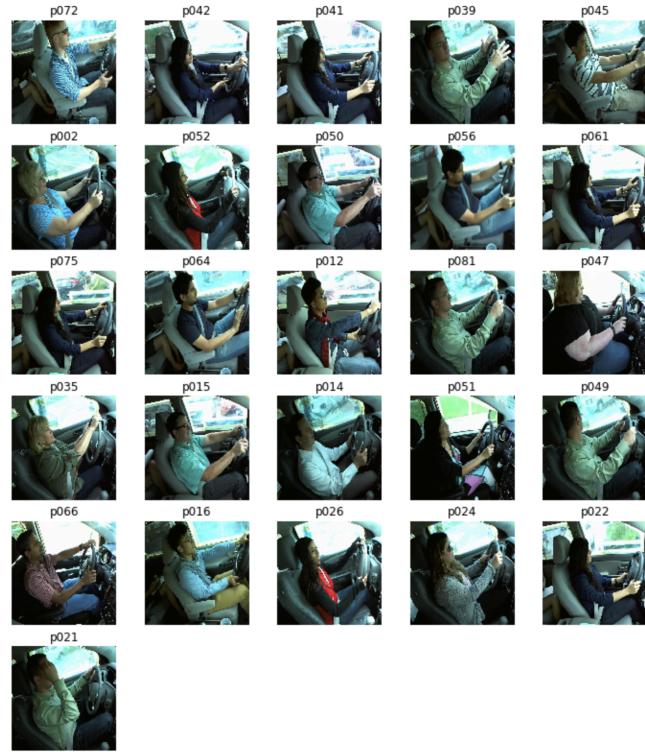
分析

数据的探索

State Farm提供了22425张已标注分类的车内摄像头拍摄的司机姿势的彩色jpeg图片，分辨率为640*480，数据分为10类，每类约2000张，分别放在对应文件夹名为train/c0-tain/c10的文件夹中。



这些数据包括了不同性别、肤色、年龄、体型、衣着、发型的26名司机。各司机的数据集差异较大，最少的一名司机只有346张图片，最多的一名则有1237张。



需要特别注意的是这些图片都是通过视频截取的。因此同一司机同一类别下的图片会非常接近，这
对我们的验证集设置提出了要求，后面我们会对数据进行相应的处理。

算法和技术

在此项目中，我们将使用深度学习方法，会搭建一个多层的神经网络来学习训练集中的图片和对应的分类标签的关系，并在验证集上验证其准确率。通过调整参数设置和使用不同神经网络模型，并分别进行多轮训练，选出在验证集及测试集上均能取得较为理想准确率的模型和参数设置。

卷积神经网络（3）

卷积神经网络(CNN)是一种包含卷积层和池化层特殊的神经网络。在卷积神经网络的卷积层中，一个神经元只与部分邻层神经元连接。在CNN的一个卷积层中，通常包含若干个特征平面(featureMap)，每个特征平面由一些矩形排列的神经元组成，同一特征平面的神经元共享权值，这里共享的权值就是卷积核。卷积核一般以随机小数矩阵的形式初始化，在网络的训练过程中卷积核将学习得到合理的权值。共享权值(卷积核)带来的直接好处是减少网络各层之间的连接，同时又降低了过拟合的风险。池化层(pooling)，通常使用max pooling的形式。池化可以看作一种特殊的卷积过程。卷积和子采样大大简化了模型复杂度，减少了模型的参数。

激活函数和Relu

在神经网络中如果不用激励函数（其实相当于激励函数是 $f(x) = x$ ），那么每一层输出都是上层输入的线性函数，无论神经网络有多少层，输出都是输入的线性组合，与只有一个隐藏层效果相当。我们在网络中会使用ReLU(Rectified Linear Units)(4)来作为激活函数，它的数学表达式如下：

$$f(x) = \max(0, x)$$

即输入信号 <0 时，输出都是0，而当输入信号 >0 的情况下，输出等于输入。

ReLU的优点：

第一，采用sigmoid等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法，计算量相对大，而采用Relu激活函数，整个过程的计算量节省很多。

第二，对于深层网络，sigmoid函数反向传播时，很容易就会出现梯度消失的情况，从而无法完成深层网络的训练。

第三，Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生。

Keras API

Keras是一个高层神经网络API，Keras由纯Python编写而成并基Tensorflow、Theano以及CNTK后端。Keras 为支持快速实验而生，能够把你的idea迅速转换为结果。（5）

使用不同的模型

本项目中使用在imagenet上预先训练过的VGG16, ResNet50, InceptionV3, Xception等深度学习模型。这些模型在历年的imagenet上都取得过理想的成绩，是被广泛参考应用的深度学习模型。Keras的应用模块Application提供了带有预训练权重的上述模型，这些模型可以用来进行预测、特征提取和finetune。可以大幅减少我们的训练和调试时间。

VGG 模型由 Simonyan 和 Zisserman 在2014年提出(6)。VGG 模型前几层使用 3×3 卷积核来增加网络深度，通过 max pooling(最大池化)依次减少每层的 神经元数量，最后三层分别是 2 个有 4096 个神经元的全连接层和一个 softmax 层。

ResNet 即“残差网络”，由He等人在2015年提出(7)。与传统的顺序网络架构（如AlexNet和VGG）不同，其加入了 $y=x$ 层(恒等映射层)，可以让网络在深度增加情况下却不退化，从而使训练更深的网络成为可能。

Inception(8)，名字来源于电影《盗梦空间》，取其意为“We need to go deeper”，由Szegedy 等人在 2014 年提出。Inception模块的目的是充当“多级特征提取器”，使用 1×1 、 3×3 和 5×5 的卷积核，最后把这些卷积输出连接起来，当做下一层的输入。Inception V3 模型提出了对 Inception 模块的更新，进一步提高了分类效果。

Xception 是由 François Chollet 提出的(9)。Xception 是 Inception 架构的扩展，它用深度可分离的卷积代替了标准的 Inception 模块。

迁移学习

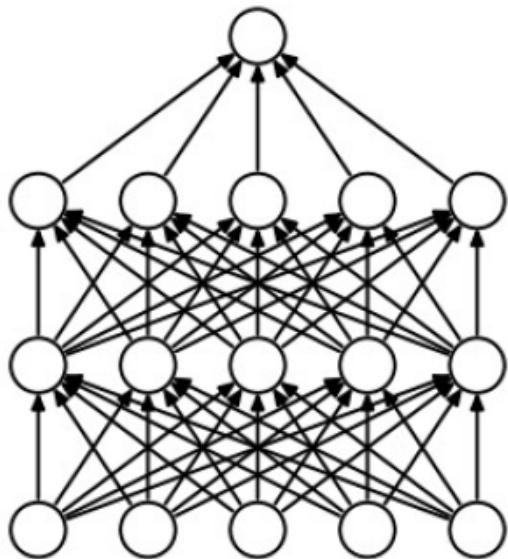
迁移学习 (Transfer learning) (10) 顾名思义就是把已学训练好的模型参数迁移到新的模型来帮助新模型训练数据集。由于大部分数据或任务是存在相关性的，所以通过transfer learning 我们可以将已经学到的parameter 分享给新模型从而加快并优化模型的学习不用像之前那样learn from zero.

多模型融合

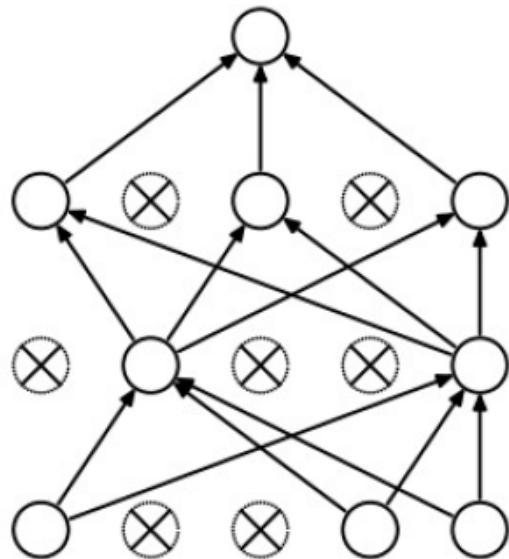
多模型的概念起源与 ‘The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations’ (11) 一书。作者James Surowiecki提出由群体决策通常要优于群体中的个体做出的决策。在实践中，我们可以选取多个不同的模型，将其得到的结果取平均值，这样可以减少单独一个模型在预测时过拟合的现象。

Dropout

dropout是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。如下图所示。



(a) Standard Neural Net



(b) After applying dropout.

根据G. E Hinton的论述(12)，在标准神经网络中，每个参数的导数告诉其应该如何改变，以致损失函数最后被减少。因此神经元可以通过这种方式修正其他单元的错误。但这可能导致复杂的协调，反过来导致过拟合，因为这些协调没有推广到未知数据。Dropout通过使其他隐藏单元存在不可靠性来防止过拟合。

Clip

由于Kaggle的评分机制原因，我们将每个类的预测概率限制在0.001-1之间，以避免多类对数对错将某个分类的概率定义为0时的过大惩罚，这样可以提高最终得分。

AWS

此项目要求的计算量较大，宜使用GPU来进行计算。由于自己没有带高性能GPU卡的电脑主机，故使用亚马逊 p2.xlarge 云服务器来完成该项目，利用[杨培文](#)在弗吉尼亚北部已经配置好的 AMI 。(1
3)

基准模型

我们将在此沿用State Farm的在线竞赛的Private Leader Board成绩表来和我们的实际结果进行对比。该比赛共有1440支队伍参加比赛。我们希望达到前30%的成绩，对照榜单排名在400名以内。即最后的log loss<0.67。关于该榜单的信息可详见[这里](#)

方法

数据预处理

分割训练集和验证集

由于Kaggle的测试集上并未提供标签，且上传验证成绩每天有次数限制，为了更快、更好的检验训练效果，我们将原始的训练集数据分为训练集和验证集。由于前述的图片都是来自视频，如果仅仅是随机从训练集中抽取部分数据用于验证，会导致模型实际上已经在训练集“看到过”类似测试集中图片的问题，故我们会每次抽取2-3个司机的数据用于验证，其他司机的数据用于训练。

根据不同预训练模型对图片进行预处理

对于InceptionV3和Xception模型，可以直接使用Keras的preprocess_input函数对图片进行预处理。VGG16和ResNet使用上述函数时会遇到问题。经查阅论文，得知VGG16和ResNet50当初训练时分别在RGB三个通道上减去了均值 103.939, 116.779, 123.68，故我们对数据做相应的手动处理。训练集的原始图片尺寸为640*480，直接训练非常耗时，故需将图片调小。我们可以直接调整为预训练模型当时使用的图片大小，如ResNet50的224*224，InceptionV3使用的299*299，也可以调整为其他更大一些的尺寸。

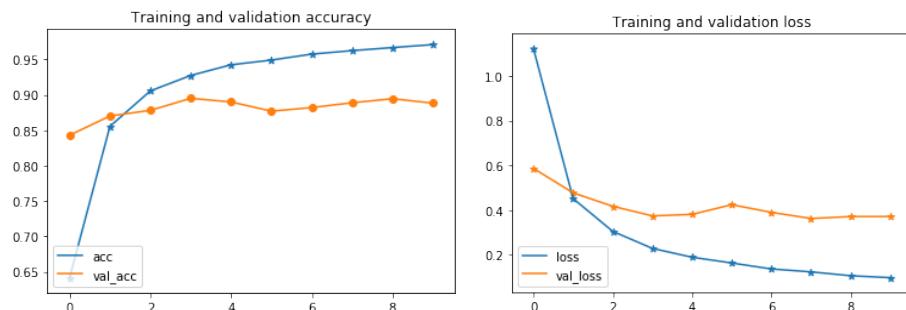
数据提升

由于原始的训练数据不多，对数据集进行切分后数量更少。故我们会采用keras.preprocessing.image.ImageGenerator来对数据进行一系列的随机变换，如小范围的旋转，在水平和垂直方向进行平移，剪切或放大。进行数据提升还可增强模型在测试集上的表现：因为摄像头的角度、司机的体型、坐姿、动作等均有可能不同，通过随机变换可适当模拟这些现象。考虑到相机都是在一个相对固定的位置，所以我们不对图像进行任何水平或者垂直的翻转。

执行过程

参考了Keras的官方文档《面向小数据集构建图像分类模型》 首先尝试使用已经在imagenet上预训练过的VGG16模型来进行训练。(14)

通过keras加载不含top的VGG16模型，并载入其在imagenet上预训练的权重。然后在该模型上加上一个GlobalAveragingPooling2D的层，再加上一个dropout层，以降低过拟合(7)。最后，加上一个dense层用做10分类的输出。先锁定新加层以外的层，进行一轮训练，然后打开模型中原VGG16最后一个卷积块，和我们新加的层一起训练。训练10轮之后的结果如下：



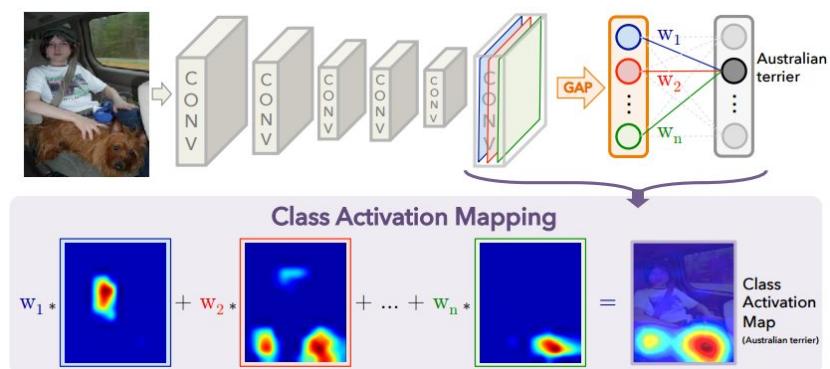
我们可以看到，在训练集上经过3轮的训练后模型的准确率即提高到0.95以上，并随着轮次的增加逐步小幅提高，同时log loss不断下降。但在验证集上，准确率和log loss的收敛情况均较差，出现了显著的过拟合现象。

出现这样的情况，和我们的数据集小 - 仅20000多张图片，样本单一 - 一共只有26个不同的司机有关。就此，我们需要进一步完善我们的方法和模型。

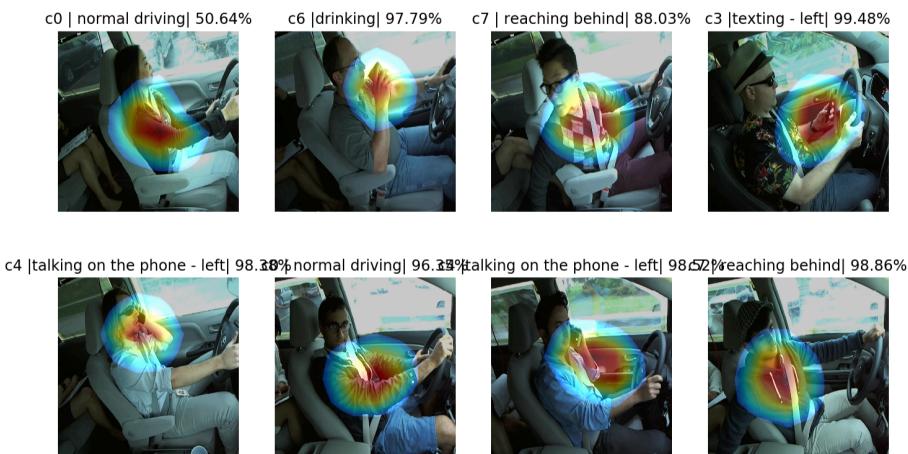
完善

对模型进行可视化

为了希望模型能够解释原因，我们参考Class Activation Mapping(15)这篇论文里的方法，对我们的网络关注的部位可视化。



我们的可视化结果如下。可以看到模型重点关注了司机手部、头部所处的位置。



使用不同的优化器和参数设置

我们尝试使用不同的优化器，如SGD, adadelta, RMSprop, Adam来优化模型，其中Adam(Adaptive Moment Estimation)本质上是带有动量项的RMSprop，实测下来和网上的普遍反映一致，收敛快，训练时间短(16)。在使用Adam时，注意到如果使用默认参数设置，其learning rate为 $1e-3$ ，尽管前期收敛很快，但一般1轮之后就不再进一步收敛。经反复尝试，发现初始可以设为 $1e-4$ ，收敛同

样很快，待不再收敛时（通常是1-2轮后），转而采用 $1e-5$ ，这样可以进一步再提高成绩，如有必要，还可以进一步采用 $1e-6$ 。节约时间起见，在后续的训练中，默认第一轮使用 $1e-4$ ，第二轮使用 $1e-5$ 。

使用不同的图片尺寸

尝试在ResNet50上使用不同的图像尺寸作为输入来训练模型，如下表。其中使用224*224的效果最差，使用320*240和400*300的效果较好。推测是因为原始图像为640*480，压缩太多后会丢失过多细节，导致预训练模型所获取的特征无法很好适用。故在后续集成模型时，均使用400*300的图像作为输入

Model	ResNet50	Batch size	64
Optimizer	Adam	Valid set	p002-p022
Learning rate	$1e-4 / 1e-5 / 1e-6$	Tune level	

img_width	img_height	epochs	val_loss	val_acc
224	224	5	0.374	0.870
240	320	3	0.225	0.922
320	240	2	0.189	0.932
400	300	2	0.219	0.931

Fine tune不同的层

我们尝试在其他设置不变，仅调整fine-tune level和batch size的情况下对比输出的结果。如下表，分别从1126，166，10开始fine-tune。其中1126，即从最后一个block开始tune的效果最差（因为太差，故略过了在kaggle验证成绩步骤），166其次，10最好。在此设置下，tune-level越多，训练时间越长，得到的准确率也越高。所以在后续集成模型时，我们均从10开始tune。

Model	Xception	Base model lay 132					
Valid set	p041-p042						
Epochs	2	Optimizer adam					
tune from layer	batch size	epochs	train time	test time	val_acc	val_loss	kaggle_score
0	16	2	2344	2688	0.904	0.453	0.262
66	64	2	1205	3432	0.875	0.389	0.297
126	64	2	936	n/a	0.748	0.727	n/a

使用不同的司机来做为验证集

使用不同的司机作为验证集，在训练集上的收敛结果被叫接近。在ResNet50上训练2轮后准确率都会达到0.98以上。但在验证集上的准确率差异巨大，这可能是由于数据样本较少，如果没有把部分司机用于训练，会导致模型的过拟合现象相对更为严重。上传Kaggle后评分结果也差异很大。故在不同的训练中，我们会使用不同的司机来作为验证集。

Model	ResNet50	Base model lay 132			
Batch size	16	Optimizer	adam		
Epochs	2	Learning rate	1e-4 /1e-5/1e-6		
Image size	400x300	Tune layer	0		
		Valid set	train time	test time	val_acc val_loss kaggle_score
		p014_p039	1820	1927	0.939 0.164 0.282
		p012_p035	1815	2845	0.912 0.334 0.445
		p052_p051_p050	1767		0.913 0.369

组合多个不同的模型

我们尝试以组合多个不同的模型的输出，例如一个ResNet50+一个Inception_V3+一个Xception或基于这些模型但使用不同的超参数设置/训练集的组合，对不同模型在若干个不同验证集下的结果进行算术平均。我们发现组合的模型数量越多，原始模型的得分越高，则组合后的输出结果得分越高，且绝大多数情况下，组合多个模型的结果都要优于单个模型。这很符合“三个臭皮匠赛过诸葛亮”的概念，利用不同模型的“群体智慧”，可以获得更理想的效果，而每个模型自身越“聪明”，那么对结果的提升也越大。由于训练较为费时，我们最多仅选取了上述三个模型不同验证集和参数设置下的六个输出进行混合。这获得了0.187的Kaggle private Score，可以排到全部1440多名参赛团队的前50名。

40 submissions for rextong		Sort by Private Score ▾		
All	Successful	Selected	Private Score	Public Score
Submission and Description		Private Score	Public Score	Use for Final Score
6_models_ensembled.csv 11 days ago by rextong		0.18713	0.21757	<input type="checkbox"/>
add submission details				
merge_ResNet50_Resnet50-2_InceptionV3_Xception-I0.csv 11 days ago by rextong		0.19309	0.21980	<input type="checkbox"/>
add submission details				
merge_Resnet50-2_InceptionV3_Xception-I0.csv 11 days ago by rextong		0.19675	0.22401	<input type="checkbox"/>
add submission details				
merge_ResNet50_InceptionV3_Xception-I0.csv 13 days ago by rextong		0.20788	0.22514	<input type="checkbox"/>
add submission details				
merge_ResNet50_InceptionV3_Xception.csv 13 days ago by rextong		0.22804	0.27113	<input type="checkbox"/>
add submission details				

结果

由于训练较为费时，我们最多仅选取了上述三个模型不同验证集和参数设置下的六个输出进行混合。这获得了0.187的Kaggle private Score，可以排到全部1440多名参赛团队的前50名。

在项目开题报告时设定的目标是对照Kaggle榜单排名在400名以内。即最后的 $\log_{10} \text{loss} < 0.67$ 。最终模型的排名时49名， $\log_{10} \text{loss}$ 是0.187，大幅高于原目标。但我们需要注意的是，从90%的推算准确率，或者我们的目测结果来看，模型还是存在不足的。我们可以假想如果将其用提醒司机专注驾驶时，会有10%的机会误报，这可能会来很多争议。但该模型仍有一定实用价值：如果仅仅将其用于评估一段时间内司机的整体专注程度，那么即使有一定误判的概率，由于累积了大量的行为，从统计学上还是可以得出一些司机比另一些司机更专注的结论。

项目结论

在整个项目中，先后使用了几层的简单神经网络、VGG16迁移学习、Fine-tune VGG16、Fine-tune更多模型、Fine-tune不同层、使用不同的optimizer、调整learning rate、选择不同的验证集、

选择不同的图片尺寸、使用不同的数据增强方法、拼接模型等方法。并借项目的机会，增加了对Python编程和numpy、matplotlib、opencv等库的知识（我此前接触编程甚少），前后累计投入了近两百小时的时间。最终取得了自己比较满意的Kaggle得分。

从对模型的提升效果来看，使用较大的图片尺寸和拼接多个模型所带来的帮助是最大的。由于P2.X large的计算性能有限，且出于时间成本的考虑，没有使用更大的图片尺寸，也没有继续训练更多的模型用于拼接。但可以预计，如果使用这样的方法，将进一步提高成绩。

另外，原始的训练集中有很多图片是标注错误的，这对模型的训练造成了很大的干扰，基于这些错误标注的图片，模型会训练到一些“自以为是”的错误规则。可以考虑使用两种方法来对训练集进行处理：1. 对一共2万张图片进行人工校对，重新标注。我们按每分钟核对10张计算，约需34小时，外包成本几千元以内可以拿下。如果是在比赛期间，为了赢得奖金这么做还是值得一试的。2. 大部分错误标注的图片处于两个动作切换之间。我们可以编程识别连续图片，并将连续图片间中间切换，发生标注变化的若干张去除，这样虽然会损失少量训练数据，但仍是一个值得一试的办法。

需要考虑的是，使用较大的图片尺寸和拼接多个模型尽管可以提高比赛成绩，但在工业应用上是否合适值得商榷：使用较大的图片尺寸，意味着更大的数据处理量和更高的计算性能要求，如果我们需要在一个低成本的车载终端上实现，这样会带来很大的麻烦。而拼接多个模型，更是需要同时载入多个模型并分别计算，这恐怕很难一个低成本的嵌入式终端上实现，如果回传后台，又要考虑传输成本和后台计算开销。

在解决现实问题时，还要综合考虑其最终使用场景。如果用于统计司机的走神时间比例，进行综合评分，那么这可以被认作是一个允许较高的误报/漏检的应用。这时候应该更多考虑成本，可能一个使用小尺寸图片、单一模型，并可获得中等水平准确率的方案会更合适。反之，如果是用作走神行为的实时提醒，那么少量漏检可接受，而频繁的误报是不可接受的。在这种情况下，我们需要将误报率，尤其是将正常行为错判为其他行为的比例降至极低（譬如说平均1小时以上才出现一次）。显然，现有的模型是不能胜任的。我们需要额外采集、标注更多的训练数据。

参考文献

1. [State Farm Distracted Driver Detection \(Kaggle\)](#)
2. [Multi Class Log Loss](#)
3. [卷积神经网络概念与原理](#)
4. [人工神经网络中的activation function的作用具体是什么？为什么ReLU要好过于tanh和sigmoid function?](#)
5. [Keras 中文文档](#)
6. K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556, 2014
7. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015
8. C. Szegedy, W. Liu, Y. Jia, et al. Rabinovich. Going deeper with convolutions. In CVPR, 2015.
9. François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv preprint arXiv:1610.02357, 2016.
10. [什么是迁移学习 \(Transfer Learning\) ?](#)
11. James Surowiecki, ‘The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations’., 2005
12. G. E. Hinton et al. , Improving neural networks by preventing co-adaptation of feature detectors, arXiv:1207.0580, 2012
13. 杨培文, [在aws上配置深度学习主机](#)
14. François Chollet, [Image classification using very little data](#)
15. Bolei Zhou et al. , [Class Activation Mapping and Class-specific Saliency](#)
16. [各种优化方法总结比较 \(sgd/momentum/Nesterov/adagrad/adadelta\)](#)