机器学习工程师纳米学位毕业项目
基本于深度学习的驾驶员分心状态检测报告

李德新

2018年7月1日

机器学习工程师纳米学位毕业项目 1
基本于深度学习的驾驶员分心状态检测报告 1
1. 定义 2
1.1 项目概述 2
1.2 问题陈述 2
1.3 评价指标 2
1.3.1 2
1.3.2 3
1.3.3 3
1.3.4 3
2. 分析 3
2.1 数据探索与可视化 3
2.2 算法与技术 4
2.3 基准模型 5
3. 方法 6
3.1 数据预处理 6
3.1.1 图片尺寸调整(Resize) 6
3.1.2 图片归一化(Normalization) 6
3.1.3 图片增强处理(Data augmentation) 6
3.2 执行过程 6
3.2.1 第1次执行 6
3.2.2 第2次执行 7
3.2.3 第3次执行 8
3.2.4 第4次执行 9
4. 结果 10
4.1 模型的评价验证与合理性分析 10
5. 项目结论 12
5.1 对项目的思考 12
5.2 需要作出的改进 12
参考文献 12

1. 定义

1.1 项目概述

此项目是在2016年由Kaggle发起的一个竞赛项目，也是Udacity在机器学习纳米学位的毕业项目中的可选项目，名为驾驶员分心状态检测(Distracted Driver Detection)。

分心状态是指驾驶员驾驶时注意力指向与正常驾驶方向不一致，从而导致驾驶操作能力下降的一种状态与现象。因驾驶人视线偏离或分心产生的注意力不集中是引发交通事故的常见且重要的原因。

导致驾驶分心的原因：接打电话，收发短信，饮食，调收音机，整理头发，乘客交谈或人个疲惫等。

此项目目的是通过技术手段(即图像识别)判断驾驶员在驾驶中处于哪种驾驶状态。

1.2 问题陈述

项目的问题是要解决一个驾驶员驾驶状态的多分类问题。

问题的解决方案，使用卷积神经网络(Convolutional Neural Network)训练Kaggle提供的实际驾驶员状态图像集，并构建识别驾驶员状态的模型，使此模型可应用到实际检测中，用以判断驾驶员处于何种驾驶状态(即安全驾驶状态与分心驾驶状态)。

驾驶状态列表：
c0: 安全驾驶
c1: 右手打字
c2: 右手打电话
c3: 左手打字
c4: 左手打电话
c5: 调收音机
c6: 喝饮料
c7: 拿后面的东西
c8: 整理头发和化妆
c9: 和其他乘客说话

我最初的期望结果，模式的准确率在85%以上，logloss在0.4以下。

但毕业项目的最低要求是logloss在0.25634, 对应的在此kaggle比赛上的leaderboard是第144位，对应的准确率大约是在90%左右，通俗的讲，就是你的logloss分数要排进前144位才达到毕业要求，leaderboard上的总人数为1440人。

一个可能的挑战，图像集采集过程中由于受限于阴影与光照等客观条件，同时由于失焦等，会带来图像模糊的可能，从而带来图像不能完全识别的可能。

1.3 评价指标

1.3.1

评价模型的指标使用Kaggle指定的multi-class logarithmic loss, 计算方法如下：

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

logloss是Kaggle竞赛中常用的分类损失函数，N代表测试集中图片的数量，M代表图片分类标签(lables)的数量，log代表以e为底的自然对数，y_{ij} 为1时，表示图片i属于第j类，为0时，表示图片i

不属于第j类，p_{ij} 代表i属性j的预测概率。当模型的预测准确越高时，则置信度越高，则logloss的值就越小。

使用logloss的理由，首先它是对每个输出计算它属于不同类的可能性，而不是输出它最有可能属于哪儿个类， 然后由它的定义可知，它也表示了真实分布与交叉熵相关性，即它可以反映出信息的不确定性，也就是可以反映出预测模型的不确定性，即logloss值越大，预测的结果就越不确定。

1.3.2

辅助的评价指标使用混淆矩阵(Confusion Matrix), 计算方法为混淆矩阵的每一列代表了预测类别，每一列的总数表示预测为该类别的数据的数目，每一行代表了数据的真实归属类别，每一行的数据总数表示该类别的数据实例的数目，每一列中的数值表示真实数据被预测为该类的数目。

使用理由可以直观的看到分类结果，以及可以清楚哪些类被误判。

1.3.3

辅助的评价指标使用分类准确率分数(即accuracy_score)，计算公式：
其中冒y表示预测y值，yi表示实际的y值，I函数为指示函数。

使用理由，这样可以给出一个比较直观模型准确度判断。

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

1.3.4

为了在训练模型的过程中，训练效果，如是否过拟合等情况，引入cost function函数，即对训练集引入loss曲线，对验证集引入val_loss的曲线。

2. 分析

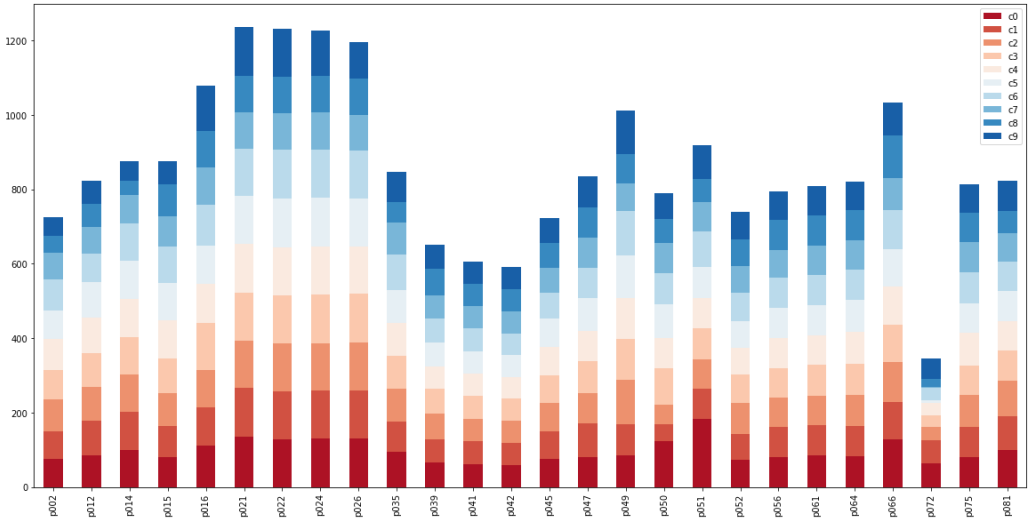
2.1 数据探索与可视化

Kaggle的数据集无法下载，所以我使用的Udacity在百度网盘上提供数据集，数据集分两部分imgs.zip与driver_imgs_list.csv。

imgs.zip解压后，包含两个文件夹，文件夹一为test，它是测试集包含79729张图片，文件夹二为train，它是训练集，包含22424张图片，其中包含10个不同的训练集分别是c0~c9, 对应着驾驶状态列表中的各个状态。

test与train中的每幅图片大小都为640 × 480像素。

driver_imgs_list.csv，它包含22424行与训练集图片数相同，分为三列，第一列subject代表驾驶员，共用26个驾驶员: p002，p012，p014，p015，p016，p021，p022，p024，p026，p035，p039，p041，p042，p045，p047，p049，p050，p051，p052，p056，p061，p064，p066，p072，p075，p081。第二列classname为当前行驾驶员所处的驾驶状态，分别为c0~c9。第三列img为图片名，每一位驾驶员都在所有的c0~c9的训练集状态中出现过，可视化关系如下，



2.2 算法与技术

因为项目本身提供的数据是为图片数据集，所以我准备使用卷积神经网络(Convolutional Neural Network)与迁移学习的技巧来建立模型解决这个问题。

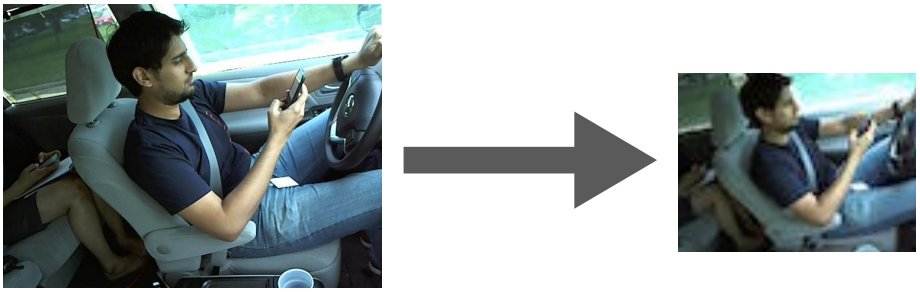
为什么要选择卷积神经网络(CNN)?

因为CNN有三个特性适合解决图像识别问题：

1. 图像中一些特定模式比整张图片小的多，一个神经元不需要看整张图片就能发现这些特定模式，如下图；

2. 图像中一些相同的特定模式可能出现在图片中的不同区域，CNN几乎可以使用相同的参数发现这些相同的特定模式。

3. 针对整张照片的子采样(subsampling)几乎不会影响被识别的目标对象，对整个网络而言，可以使用更少的参数来处理图像。



第一与第二特征的组合等价于CNN中的convolution层，第三个特性等价于CNN中的Max Pooling层。

什么是迁移学习?

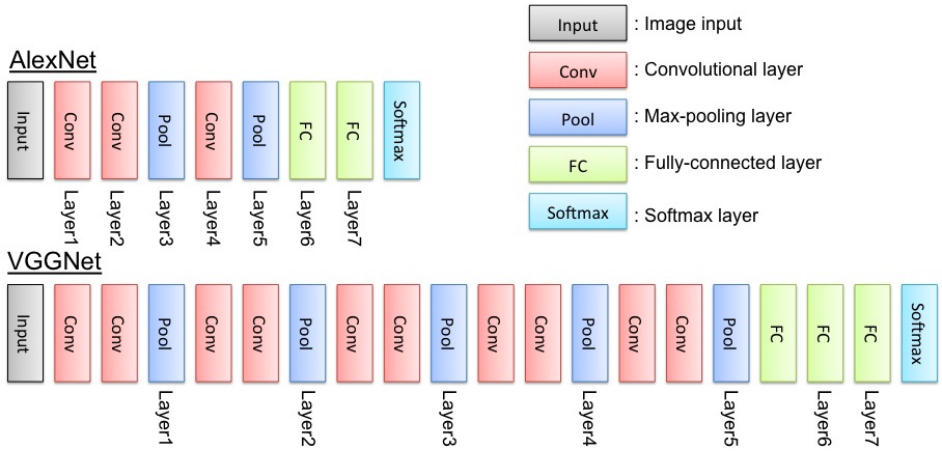
迁移学习(Transfer Learning)的目标是将从一个环境中学到的知识用来帮助新环境中的学习任务，即将模型从源任务上训练得到的知识迁移到目标任务上。

为什么要在这个项目中使用迁移学习?

如果从头构建一个卷积神经网络是完全可以的，但这会花费大量的时间，而且为了效果，同样也会花费更多的时间来调整参数，这样做性价比不高，最好是”站在巨人的肩膀上”，所以选择使用迁移学习。

在迁移学习中我将使用VGG模型，VGG 是视觉领域竞赛ILSVRC在2014年的获胜模型，以 7.3% 的错误率在 ImageNet 数据集上大幅刷新了前一年 11.7% 的世界纪录，它基本上是继承了 AlexNet 深的思想，AlexNet 只用到了8层网络，而VGG的两个版本分别是16层网络版和19层网络版，在此项目就我将使用VGG Net 16，

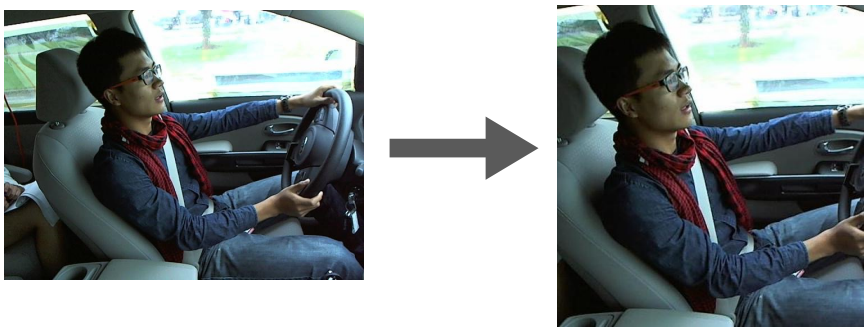
VGG的算法流程与结构图如下：



VGG 的输入数据格式是 224 * 224 * 3 的像素数据，经过一系列的卷积神经网络和池化网络处理之后，输出的是一个 4096 维的特征数据，然后再通过 3 层全连接的神经网络处理，最终由 softmax 规范化得到分类结果。

针对此项目，我共做过5次完整的实验，其中有3次是较有意义的，且在第三次实验中，在数据预处理时，使用到了基于VGG16结构的SSD300(Single Shot MultiBox Detector)目标检测算法，为什么要使用它?

在第一次数据预处理中没有使用目标检测算法，只是把源数据数据直接读取到VGG16模型中进行训练与预测，经过两次试验，所能得到的log_loss最优分数为0.27117(Private Score)与0.28612 (Public Score)， 没有达到基准模型，后思考如何能降低log_loss分数，发现输入图片中一些信息对于模型的训练是无用的，比如左下图(c0/img_101859.jpg)中最左侧的女子下半身，这有可能会形成噪音



所以我想, 如果仅把驾驶员提取出来后，再做为训练输入是不是会更好(减少了无用信息)，所以再对比已有的目标检测算法后就使用了速度较快的SSD300算法，针对左上图，使用SSD300后识别与裁剪后，得到右上图。

2.3 基准模型

基准模型以Kaggle上相同竞赛上的Leaderboarder得分为基准模型，第一这是因为在Leaderboarder上每个人的得分都是以相同的评价指标评测的，第二可以客观的清楚自身模型的竞争力。我的目标是可以进过前20%，即全体1440名中的前288名，对应的score分值小于0.44534。

但毕业项目的最低要求是logloss在0.25634, 对应的在此kaggle比赛上的leaderboard是第144位，对应的准确率大约是在90%左右，通俗的讲，就是你的logloss分数要排进前144位才达到毕业要求，leaderboard上的总人数为1440人。所以最终以logloss在0.25634的分数为基准。

3. 方法

3.1 数据预处理

3.1.1 图片尺寸调整(Resize)

因为要基于VGG16 做迁移学习，图片的输入要与预训练模型VGG16输入要求(224 × 224)的应该相同，所以首先需要把图片的(640 × 480)变为shape的(244 × 244)。

3.1.2 图片归一化(Normalization)

为了加快训练网络的收敛性，对所有图片进行归一化处理，方法是像素中的色彩值除以255。

3.1.3 图片增强处理(Data augmentation)

图片增强的目的，在观察完测试集与训练集后，发现测试集数量为训练集数量的3.56倍(79726:22424), 极易出现过拟合(overfitting)的情况，所以使用图片增强处理来减小过拟，

第一rotation_range，设图片的随机转动的角度
第二width_shift_range，设图片的随机平移比例
第三height_shift_range，设图片的随机垂直移动比例
第四zoom_range，设图片的随机缩放因子

3.2 执行过程

总体的执行过程：原始图片(640*480)->剪裁(224*224)->数据预处理(归一化，随机旋转/偏移/缩放)->使用交叉验证法与调参法训练模型->挑选泛化误差(generalization error)最小的模型参数

3.2.1 第1次执行

数据预处理：
zoom_range=0.4, 设图片的随机缩放因子为0.4
width_shift_range=0.2, 设图片的随机平移因子为0.2
height_shift_range=0.2, 设图片的随机垂直移动因子为0.2
rotation_range=20，图片的随机转动角度为20%

交叉验证：
nfolds=8
epochs=8
batch_size=64

调参：
1) 使用Keras的vgg16模型, 优化器(optimizer)使用自适应矩估计(adam, adaptive moment estimation)，learning rate = 1e-5，其余参数使用默认值即(beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)

2) 设置vgg16的前9层为冻结层(即不可训练)，后7层为非冻结层(即可训练). 具体代码可看distracted_driver_detection_1st.html中的VGG16_arch函数

3) 在callbacks中设置EarlyStopping函数, EarlyStopping(monitor='val_loss', patience=3, verbose=0)，即当log_loss连续3次(epochs)都无改进时，终止此轮训练。
在callbacks中设置ReduceLROnPlateau函数，ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2)，即当log_loss连续2次无改进时，更新learning rate = learning rate * 0.2

实验结果：
将每个folds后的模型log_loss得分相加除以共8，得到Validation Average Score: 0.42821977159174485.
从8 folds中，选择log_loss最优的两个模型
weights_kfold_augmented_VGG16_4.h5(0.24228694829385156) 与
weights_kfold_augmented_VGG16_5.h5(0.22967805627827453) 进行集成学习(ensemble learning) , 得到submission_10_vgg16.csv，上传kaggle测试，

submission_10_vgg16.csv a month ago by Michael	0.37820	0.37461

分数0.37820(Private Score), 0.37461(Public Score)

代码：distracted_driver_detection_1st.html

3.2.2 第2次执行

数据预处理：
zoom_range=0.4, 设图片的随机缩放因子为0.4
width_shift_range=0.3, 设图片的随机平移因子为0.3
height_shift_range=0.3, 设图片的随机垂直因子为0.3
rotation_range=30，图片的随机旋转角度为30%

交叉验证：
nfolds=8
epochs=12（比较第1次试验，提升了epochs次数)
batch_size=64

调参：
1) 使用Keras的vgg16模型, 优化器(optimizer)使用自适应矩估计(adam, adaptive moment estimation learning rate = 1e-4，其余参数使用默认值即(beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)

2) 设置vgg16的前9层为冻结层(即不可训练)，后7层为非冻结层(即可训练). 具体代码可看distracted_driver_detection_2rd.html中的VGG16_arch函数

3) 在callbacks中设置EarlyStopping函数, EarlyStopping(monitor='val_loss', patience=5, verbose=0)，即当log_loss连续5次(epochs)都无改进时，终止此轮训练。
　在callbacks中设置ReduceLROnPlateau函数，ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=2)，即当log_loss连续2次无改进时，更新learning rate = learning rate * 0.1

实验结果：
虽然交叉验证设nfolds=8，但最终由于内存问题(ResourceExhaustedError: OOM)，我只跑了前7轮，得到Validation Average Score=(0.3789902361521193+ 0.47316775249603626+ 0.48276387258016984+ 0.14978538320541174+ 0.21092788194156215+ 0.4216778239278152+ 0.36178192757799593)/7 =0.3541564111258729，比第一次执行提升了0.0741.

从8 folds中，选择log_loss最优的两个模型
weights_kfold_augmented_VGG16_4.h5(0.14978538320541174) 与
weights_kfold_augmented_VGG16_5.h5(0.21092788194156215) 进行集成学习(ensemble learning) , 得到submission_10_vgg16_20180724.csv，上传kaggle测试，
kaggle分数为0.27117(private score)，0.28612(public score)

submission_10_vgg16_20180724.csv 0.27117 0.28612
a month ago by Michael

对比分析第1次执行与第2次执行，结果第2次比第1次的结果要好，第2次比第我1次执行有参数设置上的不同，主要包括
1) epochs每轮次数的提高(8改为12)，
2) adam中learning rate由1e-5改为1e-4，
3) ReduceLROnPlateau中factor由0.2改为0.1

代码：distracted_driver_detection_2nd.html

3.2.3 第3次执行

思考与完善，第1次与第2次执行的最优结果，只达到了0.27117(private score)，0.28612(public score)，都没有达到基准模型的要求，重新思考，得到如下两点：
1) 手动翻看了训练图集照片，发现有些图片中，存在一些无用或干扰信息，所以使用SSD300目标识别算法，针对训练集与测试集提取出所有驾驶员信息，然后再训练与测试。
2) 依然存过拟合(overfitting)现象，所以想加增加训练集图片数量，方法是使用第二次得到的最优模式，对测试集进行测试，再从预测结果的.csv中进行随机分层采样(stratified sampling)，即从每个类别中随机采样800个用例，一共10个类别，即采8000个样本，做为pseudo sample放入到每一轮的测试中。

第3次执行的过程：
原始图片(640*480)->使用目标识别算法提取出数据集中的驾驶员->剪裁(224*224)->从之前的测试结果中随机分层采样出8000个pseudo样本->数据预处理(归一化，随机旋转/偏移/缩放)->使用交叉验证法且与调参法训练模型->挑选泛化误差(generalization error)最小的模型参数

识别驾驶员并保存：
1) 方法使用基于VGG16的SSD300算法提取并保存驾驶员信息
2) 使用深度学习的计算机视觉库ChainerCV中自带的SSD300算法
   安装ChainerCV库：
   ```
   # Download ChainerCV and go to the root directory of ChainerCV
   git clone https://github.com/chainer/chainercv
   cd chainercv
   conda env create -f environment.yml
   source activate chainercv
   # Install ChainerCV
   pip install -e .
   ```
3) 运行代码Get_ROI_From_Test_and_Train.ipynb，得到并保存训练集与测试集的驾驶员信息。

采样8000个样本：
1) 根据第2次执行时得到的模型，生成了针对测试集的Submission文件(csv)，再从csv文件中，对每个类别(c0~c9)，随机地采样出800个样本，共采样8000个样本，在函数read_and_normalize_pseudo_train_data(img_rows=224, img_cols=224, color_type=3, use_roi_picture = True, pseudo_number = 8000)中实现，因为这8000个样本原本不是训练集中数据，而是来自测试集，虽然是针对每一类随机抽样，但模型本存在一定的错误率(人工估计错误率应在10%左右)，所以抽取来的样本，大约只有90%左右的准确率，对这8000个样本，我称之为pseudo sample(伪样本)
2) 参数use_roi_picture表示是否使用经过目标识别后的数据集，True表示使用，False表示使用未经处理过的数据集。
3) 参数pseudo_number，指定使用多少pseudo sample，默认是8000个。

数据预处理：
zoom_range=0.2, 设图片的随机缩放因子为0.2
width_shift_range=0.1, 设图片的随机平移因子为0.1
height_shift_range=0.1, 设图片的随机垂直因子为0.1
rotation_range=10，图片的随机旋转角度为10%

交叉验证：
nfolds=7, 因为每轮都加入了8000个，所以每轮的训练时间就变长了，费钱，就减小一轮由8变7
epochs=10，原因如上，由12变成10
batch_size=32，batch_size为64时到第4轮后会有遇到内存问题(ResourceExhaustedError: OOM)，影响程序持续执行，改为32后，就没有遇到ResourceExhaustedError问题了。

调参：
1) 使用Keras的vgg16模型, 优化器(optimizer)使用自适应矩估计(adam, adaptive moment estimation learning rate = 1e-4，其余参数使用默认值即(beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
2) 设置vgg16的前5层为冻结层(即不可训练)，后11层为非冻结层(即可训练)，去除全链接层，以GlobalMaxPooling2D()代替全链接层。具体代码可看distracted_driver_detection_3rd.html中的VGG16_arch函数。
3) 在callbacks中设置EarlyStopping函数, EarlyStopping(monitor='val_loss', patience=4, verbose=0)，即当log_loss连续4次(epochs)都无改进时，终止此轮训练。
　在callbacks中设置ReduceLROnPlateau函数，ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=2)，即当log_loss连续2次无改进时，更新learning rate = learning rate * 0.1。

实验结果：

将7轮交叉验证训练后的模型log_loss得分相加除以7，得到
Validation Average Score: 0.2211248497001405。

从7个模型中选择log_loss分值最优的前5个模型，分别为：
weights_kfold_augmented_VGG16_1.h5(0.19838330690032746)
weights_kfold_augmented_VGG16_3.h5(0.21280435669769293)
weights_kfold_augmented_VGG16_4.h5(0.08709168978878683)
weights_kfold_augmented_VGG16_6.h5(0.1336115817067305)
weights_kfold_augmented_VGG16_7.h5(0.28973210362586893) 进行集成学习(ensemble learning) , 得到submission_10_vgg16_20180810_Use13467_VGG16.csv，上传kaggle测试，分数0.22914(Private Score), 0.23629(Public Score)

submission_10_vgg16_20180810_Use13467_VGG16.csv	0.22914	0.23629
13 days ago by Michael		

代码：distracted_driver_detection_3rd.html

3.2.4 第4次执行

针对第3次的执行结果，两位审阅者针对图片预处理阶段的图片归一化，给出的参考如下，
即因为使用了以imagenet为权重的预处理模型VGG16，所以图片的预处理方法建议与VGG16处理imagenet时的处理方法对齐，具体方法，先中心化(第1与第2步)，再归一化(第3步):
1) 先改变图片通道顺序，RGB -> BGR
2) 后减均值，[103.939, 116.779, 123.68]
3) 再图片归一化，像素中的色彩值除以255.0

数据预处理：与第3次执行相同。

交叉验证：与第3次执行相同。

调参：与第3次执行相同。

实验结果：
将7轮交叉验证训练后的模型log_loss得分相加除以7，得到
Validation Average Score: 0.199296084929985

选择这7轮的所有结果做为模型，分别为：
weights_kfold_augmented_VGG16_1.h5(0.17328886596258297)
weights_kfold_augmented_VGG16_2.h5(0.28775123497219235)
weights_kfold_augmented_VGG16_3.h5(0.15942442468966941)
weights_kfold_augmented_VGG16_4.h5(0.0768067412470244)
weights_kfold_augmented_VGG16_5.h5(0.25205739187296405)
weights_kfold_augmented_VGG16_6.h5(0.20033607819937477)
weights_kfold_augmented_VGG16_7.h5(0.24540785806256105)
进行集成学习(ensemble learning)，得到
submission_10_vgg16_20180830_Use1234567_VGG16.csv，上传kaggle测试，
评分：0.22344(Private Score), 0.23230(Public Score)

submission_10_vgg16_20180830_Use1234567_VGG16.csv a minute ago by Michael	0.22344	0.23230

代码：distracted_driver_detection_4th.html

此4次执行时所需参数的对比表格与总结，请参见《模型及参数对比与总结》的PDF文件。

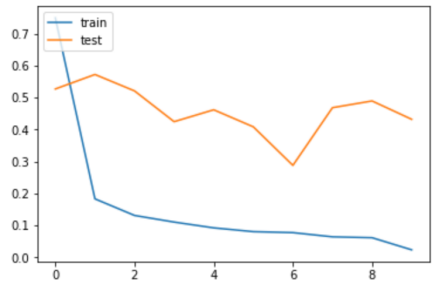
4. 结果

4.1 模型的评价验证与合理性分析

经过多轮的训练与选取，最终模型使用的是迁移学习的VGG16模型，前5层为冻结层，后11层为训练层，训练权重使用的是imagenet，选它是因为这个结构最终的log_loss值最低。

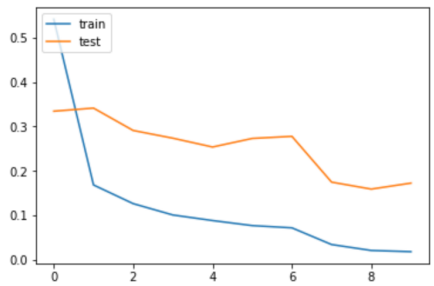
从kaggle的最终测试结果分析，模型的稳健与可靠性都可以接受，因为log_loss的得分无论是在public 还是private上都稳定在0.2278左右，且训练数据的一些微小改变是不会极大的影响结果，比如随机的平稳与旋转。

从第4轮实验结果看，所选的7个模型进行集成学习后的在测试集上的预期log_loss值应该是在0.1992左右，但实际在kaggle上的测试结果在0.2278左右，仍然存在的0.0286的偏差，这说明模型过拟合的情况依然存在，同时从loss与val_loss中也能看出过拟合的情况存在，分析所选7个模型的验证集的confusion_matrix, accuracy_score，loss与val_loss如下(对第1个模型图片，由于我误操作，只保存下来了confusion_matrix与accoracy_score)：



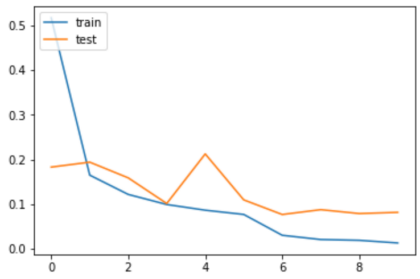

```
Score accuracy_score:  0.9103573591762568
[[327   0   0   0   1   0   0   0   1  20]
 [  1 344   0   0   0   0   0   0  12   2]
 [  0   2 341   0   0   0   1   0   8   0]
 [ 22   0   0 326   2   0   0   0   0   0]
 [  0   0   0   1 359   0   1   0   0   0]
 [  3   0   0   0   0 357   0   0   0   0]
 [  0   6   0   0   0   2 297   0  27   4]
 [  0   0   0   0   0   1   0 292   1   8]
 [ 13   0   0   2  22   0   5  28 178  16]
 [ 77   0   0   1   0   0   0   0   6 185]]
Score log_loss:  0.28775123497219235
```

model 2



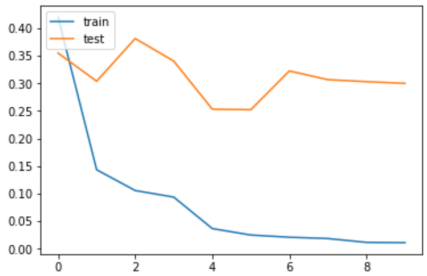

```
Score accuracy_score:  0.9586848996217632
[[317   0   0   0   0   0   0   0  13  13]
 [  6 347   0   0   0   0   0   0   0   0]
 [  0   0 342   0   0   0   0   0  10   0]
 [  0   0   0 358   0   0   0   0   0   0]
 [  0   0   0   0 360   0   0   0   6   0]
 [  0   1   1   0   1 353   0   0   3   9]
 [  0   0   0   0   0   0 353   0   9   0]
 [  0   0   0   0   0   0   0 307   0   0]
 [  0   2   2   0   0   0  18   0 287   0]
 [ 38   0   0   0   0   6   0   0   4 271]]
Score log_loss:  0.15942442468966941
```

model 3



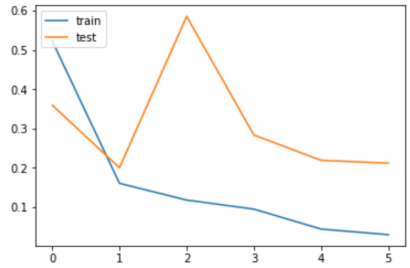

```
Score accuracy_score:  0.9819444444444444
[[401   1   0   0   0   0   0   0  10  24]
 [  0 327   0   0   0   0   0   0   0   0]
 [  0   0 361   0   0   0   0   0   0   0]
 [  0   0   0 374   6   0   0   0   0   0]
 [  0   0   0   0 352   0   0   0   0   0]
 [  2   0   0   0   0 357   0   0   0   3]
 [  0   0   0   0   0   0 379   0   1   1]
 [  0   0   0   0   0   0   0 313   1   0]
 [  2   1   1   0   0   2   6   0 286   1]
 [  2   0   0   0   0   0   0   0   1 385]]
Score log_loss:  0.0768067412470244
```

model 4



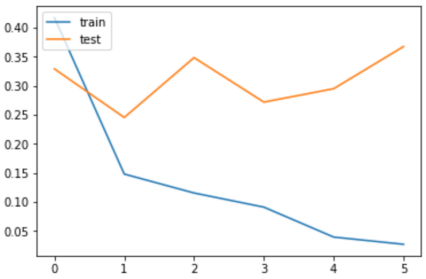

```
Score accuracy_score:  0.9340915718868474
[[345   0   0   0   0   3   0   0   0  16]
 [  0 340   0   0   0   0   0   2   0   1]
 [  0   0 345   0   0   0   0   1   8   0]
 [ 11   0   0 335   0   0   0   0   0   0]
 [  3   0   0   2 336   0   0   0   1   0]
 [ 14   0   0   0   0 330   0   0   0   1]
 [  0   3   1   0   0   0 332   0   9   0]
 [  1   1   0   0   0   0   0 311   1   4]
 [  0   9   9   0   4   0   5  23 289   9]
 [ 43   1   1   0   0  11   1   3  24 240]]
Score log_loss:  0.25205739187296405
```

model 5




```
Score accuracy_score:  0.9327079934747146
[[274   0   0   0   0   8   0   0   1  15]
 [  0 203   0   3   0   0   1   0   0   0]
 [  0   0 203   0   0   0   0   0  22   0]
 [  0   0   0 262   0   0   0   0   4   0]
 [  1   0   0   5 249   0   0   0   6   0]
 [  5   0   0   0   0 254   0   0   0   0]
 [  2   0   0   0   0   0 256   0   0   0]
 [  0   0   0   0   0   1   0 249   0   0]
 [ 17   0   0   0   0   3   0   0 171   9]
 [ 40   3   0   1   0  16   0   0   2 166]]
Score log_loss:  0.20033607819937477
```

model 6




```
Score accuracy_score:  0.9353680430879713
[[287   1   0   0   0   0   0   6  17]
 [  4 283   0   0   0   0   2   0   2   4]
 [  3   0 289   0   0   1   0   0   7   9]
 [  4   0   0 285   0   0   0   0   0   0]
 [  0   0   0   0 290   0   0   0   3   0]
 [  4   0   0   0   0 281   0   0   2   0]
 [  0   1   1   0   0   0 281   0   5   0]
 [  0   0   1   0   0   0   1 242   3   0]
 [ 10   0   4   0   6   0   3   0 178   3]
 [ 48   0   0   0   0   0   0   2  23 189]]
Score log_loss:  0.24540785801708173
```

model 7

```
Score accuracy_score:  0.9502778590231061
[[356   6   0   0   0   0   0   0   7  19]
 [  0 382   0   0   0   0   0   0   0   1]
 [  0   0 325   0   0   0   0   0  39   0]
 [ 14   0   0 335   0   0   0   0   1   7]
 [  0   0   0   2 349   0   0   0   0   0]
 [  1   0   0   0   0 330   0   0   0   0]
 [  0   2   0   0   0   0 340   0  13   0]
 [  0   0   0   0   1   0   0 263   0   0]
 [  8   1   1   0   0   0   1   1 274   1]
 [ 37   0   0   0   3   3   0   0   1 295]]
Score log_loss:  0.17328886596258297
```

model 1

从以上的矩阵中可以看出，第一：c0与c9状态在验证时，最易出现误判，分析两个状态集的图片，可以看出安全驾驶状态与和其他人说话的状态是极相近的，第二，c0~c3,c6,c9易被误判成c8状态。

但同时也能看出模型预期的准确性，所以它仍是合理与可信的。

5. 项目结论

5.1 对项目的思考

在前两次实验中，我所能得到的log_loss的最好实验结果为0.27左右，在我寻找更优解决方案时，我看到了bobutis在这次比赛中提到的建议，就是使用faster R-CNN与pseudo sample的方法，他把log_loss降到了0.12673，所以在参考此建议的基础上，我首先使用了效率比faster R-CNN更快的SSD300，对所有数据集，进行了目标识别并对驾驶员状态进行了提取，之后再使用原有的VGG16进行训练，但发现但无明显提升效果，但之后叠加使用pseudo sample方法，log_loss值就有了明显的改善，所以可以看出：

1) 虽然pseudo sample有一定的样本错误率，但此方法是可行且有有效的;
2) SSD300对改善log_loss没有直接帮助，但在code的实际运行中，比之前未使SSD进行裁剪的数据集运行时，节省了大约50%的内存，所以工程上也是有改善的。

但针对bobutis给出的建议，他可以达到0.12673，但我没有达到，我还没找到这之间的gap在哪儿里，这是需要后期在深度学习课程中进一步思考的地方。

5.2 需要作出的改进

第一，针对深度学习的计算机视觉库chainercv中实现的SSD300，当我大量比对裁剪前后的数据集图片后，发现部分照片在识别裁剪后，没有得到很好的预期效果(双手完全都裁剪掉了)，比如针对img_35976.jpg，在使用SSD300后得到的照片img_35976_ROI.jpg，对比如下，这在后续程序改进中，还需要调整。

第二，针对4.1节提出的两个导致log_loss低的问题，在后续中也要处理。

参考文献

http://www.baike.com/wiki/%E5%88%86%E5%BF%83%E9%A9%BE%E9%A9%B6
https://trl.co.uk/sites/default/files/PA-TR-4224-05.PDF
https://www.nhtsa.gov/risky-driving/distracted-driving
https://en.wikipedia.org/wiki/Distracted_driving
https://datawookie.netlify.com/blog/2015/12/making-sense-of-logarithmic-loss/
https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow/discussion/2499
http://wiki.fast.ai/index.php/Log_Loss
https://cosx.org/2017/10/transfer-learning/
https://en.wikipedia.org/wiki/Transfer_learning#cite_note-1
http://www.gooseeker.com/cn/node/transferlearning
https://cosx.org/2017/10/transfer-learning/
https://www.cs.unc.edu/~wliu/papers/ssd.pdf