

# 司机驾驶行为状态检测

——机器学习（进阶）纳米学位毕业项目

（王瑞 2018 年 4 月 22 日）

## 1 问题的定义

### 1.1 项目概述

该项目是 StateFarm 公司在 Kaggle 上发起的一个竞赛项目[1]，目的是得到一个模型，用来检测驾驶员在驾驶过程中是否走神。

根据疾病预防控制中心（CDC）的研究成果[2]，五分之一的汽车事故是由于驾驶过程中分心导致的，这意味着每年有约 425000 人因此受伤，3000 人丧命。典型的驾驶分心行为主要包括视野离开了路面、双手离开方向盘或注意力不在驾驶上。StateFarm 公司希望通过仪表盘上的摄像头自动检测驾驶过程中的分心行为，进而提醒驾驶员集中注意。为此，StateFarm 公司准备了 22424 张带有标签的图片，解决该问题需要涉及到图像处理和深度学习的知识。

### 1.2 问题陈述

该项目要求给定一张 2D 图片作为输入数据，输出 10 种驾驶状态的概率。本质上是监督学习中的分类问题，提供的作为训练的数据集已经做好了分类标记。

首先按照司机 ID 对测试集进行划分，用 opencv 读取图片数据并进行预处理；然后构建合适的模型，利用 K 折交叉验证对训练集中的数据进行训练；训练过程中监测训练集和验证集的 accuracy 和 loss，最终得到 K 个模型；然后利用 K 个模型对测试集进行测试，取所有模型测试结果的平均值并保存为 csv 文件，每一行记录图片名和 10 种行为的预测概率。

### 1.3 评价指标

评价指标分为两部分，一部分是预测结果的准确性，另一部分是训练和预测的时长。

准确性采用 multi-class logarithmic loss 评价，是对所有测试集预测概率取对数后加权得到的一个值，值越接近于 0 表示模型预测越准确，具体方程为：

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

其中， $N$  是测试集中图片的数量； $M$  是分类标签的数量； $\log$  是自然对数；当输入  $i$  属于  $j$  类时  $y_{ij}$  为 1，否则为 0； $p_{ij}$  为输入  $i$  属于  $j$  类时的预测概率。对于单个确定的输入图片，应该输出 10 种类别对应的预测概率，为避免  $\log$  函数取极端值， $p_{ij}$  取值为  $\max(\min(p, 1 - 10^{-15}), 10^{-15})$ 。

时长可以直接通过记录训练和预测过程中的时间得到。

## 2 分析

### 2.1 数据的探索

此数据集可以从 kaggle 上下载，包含三个文件：

- 1) imgs.zip: train/test 图片文件的压缩包；
- 2) sample\_submission.csv: 给出了提交文件的格式模板；
- 3) driver\_imgs\_list.csv: 训练集中驾驶员 ID、图片编号和标签的对应关系。

数据集为彩色图片，已经划分为训练集和测试集，图片内容是驾驶员在驾驶过程中的不同行为。训练集分为 10 类，标签为 c0 ~ c9，分别表示：

- 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

同时，训练集提供了驾驶员编号(subject)、包含驾驶行为的图片编号(img)和行为类别标签(classname)的对应关系；测试集为待预测图片的集合。图 1 是对“driver\_imgs\_list.csv”文件前三行的展示，其中“subject”代表驾驶员的代号，“classname”代表对应的驾驶状态，取值 c0~c9，img 代表对应的图片名。

	subject	classname	img
0	p002	c0	img_44733.jpg
1	p002	c0	img_72999.jpg
2	p002	c0	img_25094.jpg

图 1 “driver\_imgs\_list.csv” 展示

所提供的数据中，训练集数据共 22424 个，驾驶员 ID 共 26 个，标签共 9 个；测试集数据共 79726 个。

数据集中的图片如图 2 所示，尺寸为 640\*480。



图 2 驾驶状态图示例

2.2 探索性可视化

对数据中的训练集进行分析，如图 3 所示，c0~c9 每个标签对应的数据量比较均衡，均在 1900~2500 之间；根据驾驶员 ID 进行分类，每个 ID 的图片数量分布在 346~1237 之间，且大多数数量为 800 张左右（如图 4）；图 5 展示了每个驾驶员数据中的标签分布状况,说明每个驾驶员类别中都包含 c0~c9 的所有标签，且大部分类别中的标签分布与整个数据集中的标签分布相似。

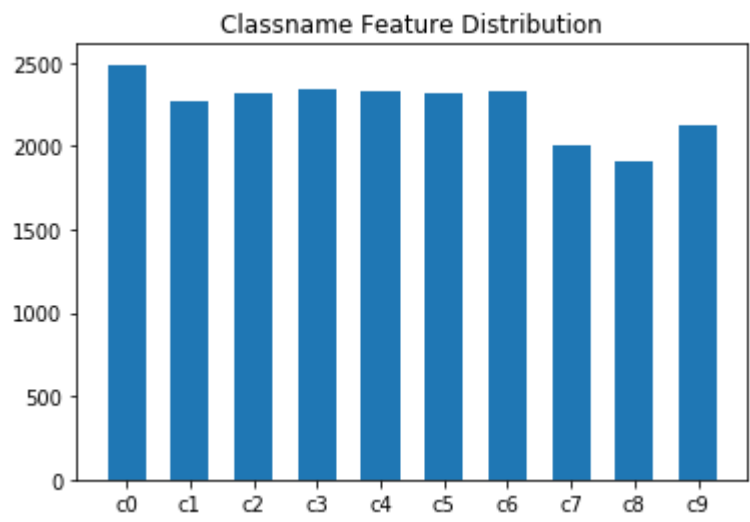


图 3 标签-数据量分布图

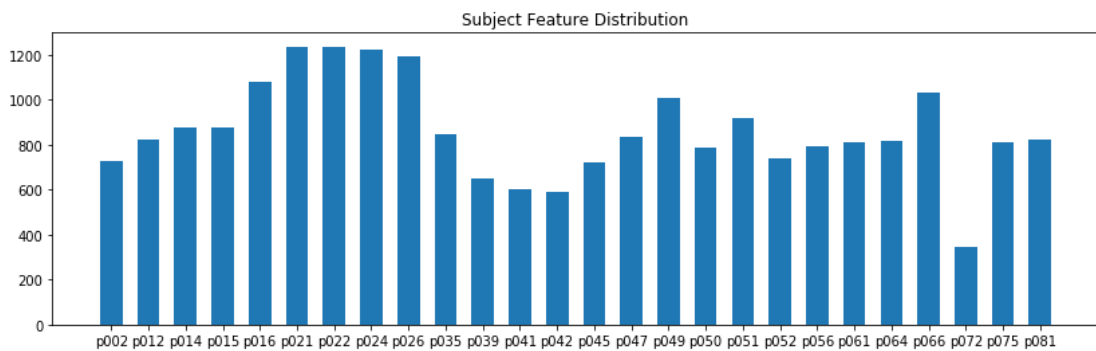


图 4 驾驶员 ID-数据量分布



图 5 每个驾驶员 ID 的标签-数据量分布

## 2.3 算法和技术

项目目的是通过学习训练集的图片数据，可以对测试集的图片进行分类，是一个基于监督学习的多分类识别问题。对于图片分类问题，卷积神经网络（CNN）[\[3\]](#)可以说是目前公认的最有效的方法。从 2012 年 AlexNet 在 ImageNet 的 LSVRC

（大规模视觉识别挑战赛）中以远超第二名的成绩夺得冠军，CNN 开始重新回到大众视野，此后的每年 LSVRC 大赛都是卷积神经网络的天下，并发展出更多的模型，如 VGGNet (2014.09)、GoogLeNet (2014.09)、ResNet (2015.12)、Inception v3 (2015.12)、InceptionResNetV2 (2016.02)、Xception (2016.10)、ResNeXt (2016.11) 和 NASNet (2017.07)。

一般这些模型已经在大数据集 Imagenet 上进行了预训练，可以使用这些预训练过的模型作为初始模型或者特征提取器，进行迁移学习。该项目可以使用 opencv 进行图片预处理；scikit-learn 做交叉验证和评价指标的计算；keras 用来创建神经网络模型，是对 tensorflow 的高级封装。选取三个模型 VGG16[4]、ResNet50[5]和 InceptionResNetV2[6]分别进行 finetune。

VGG16 是由 AlexNet 演化而来，分成 5 层（组），每层有 2~3 个 convolution 层，每层之间用 max pooling 层分开，最后再加三层 fully connected 层，filter 的大小为 3x3，默认输入尺寸是 224 x 224。ResNet 即深度残差网络，其出现是为了解决网络深度增加性能下降的问题，主要特色是跨层连接，这里选用 ResNet50 深度为 50 层，默认输入为 224x224。InceptionResNetV2 由 Inception v3 模型演化而来，同时结合了残差网络的思想，默认输入尺寸为 299x299。

本项目使用了 AWS EC2 上的 p2.xlarge 和 p3.2xlarge 实例，选用的 AMI（Amazon 系统映像）为 Deep Learning AMI (Ubuntu) Version7.0。p2.xlarge 为 4 个 CPU，型号为 Intel Xeon E5-2686v4，2.7GHz 主频，61GB 内存，1 个 Nvidia K80 GPU，显存 12GB；p3.2xlarge 为 4 个 CPU，型号为 Intel Xeon E5-2686v4，2.7GHz 主频，61GB 内存，1 个 Nvidia V100 GPU，显存 16GB。

## 2.4 基准模型

kaggle 上有两个榜单，Public Leaderboard 使用 31%的数据集计算 log loss，Private Leaderboard 使用剩下的 69%计算 logloss。计算结果越小排名越高，最终排名以 Private Leaderboard 为准。kaggle 上有 1440 只队伍提交了有效结果，本项目的目标是进入 kaggle 排行榜前 10%，即第 144 名，对应的 Private Leaderboard 上的 logloss 为 0.25634。

## 3 方法

### 3.1 数据预处理

由于以上选择的深度学习模型需要输入的数据格式为 numpy 数组，图片数据均应经过预处理再传入模型。

首先用 opencv 读取图片数据，并转换成 RGB 格式存储；

然后将图片数据缩放至要求的尺寸，VGG16 和 ResNet50 要求输入尺寸为 224\*224，Inception\_Resnet\_V2 要求尺寸为 299\*299；

再根据不同的模型应用不同的数据预处理方法，VGG16 和 ResNet50 是将数据根据 Imagenet 数据集上的均值进行零均值化处理，而 Inception\_Resnet\_V2 是将输入数据缩放到 $[-1, 1]$ 之间。Keras 中已有对应的“preprocess\_input”函数可以直接调用。

最后对提取完的数据进行随机打乱。

数据提取的过程需要注意标签与图片的对应关系，标签应进行独热编码处理。为避免显存不够用，可以在数据读取的时候保存为 np.uint8 格式，构建模型的时候加入 lambda 层进行数据预处理。

## 3.2 执行过程

为实现该项目，共构建了 3 个模型进行处理，分别是 VGG16、ResNet50 和 InceptionResnet V2。

### 3.2.1 VGG16

VGGNet 是由牛津大学计算机视觉组 (Visual Geometry Group) 和 Google DeepMind 公司一起开发的深度卷积神经网络。通过反复堆叠  $3 \times 3$  的小型卷积核和  $2 \times 2$  的最大池化层，成功构建了 16~19 层深的卷积神经网络，并取得了 ILSVRC 2014 定位项目的第 1 名。由于 VGGNet 结构简洁，拓展性强，经常被用来提取图像特征和对数据集测试。

本项目先选取 Keras 中预训练的 VGG16 进行训练和预测。weights 选用“imagenet”上预训练好的，去掉顶层全连接层，对最后一层卷积层应用 global average pooling，添加全连接层和分类层，全连接层后加入 dropout 层避免过拟合，分类层激活函数为 softmax。

模型训练分为两个阶段：第一步，锁住除全连接层以外的所有层，仅训练新添加的层，这样可以使新添加的层的 weights 快速收敛到最佳位置附近。因为预训练模型的其他层 weights 本来在最佳位置附近，而新添加的层 weights 是随机初始化的，如果一起训练，会由于新层梯度过大导致预训练模型的 weights 大幅度更新，偏离最佳位置。

第二步，等全连接层趋近收敛之后，放开感兴趣的层，再进行微调。

### 3.2.2 ResNet50

ResNet 在 2015 年被提出，在 ILSVRC 的分类项目上获得第一名。ResNet 使用了一种叫“shortcut connection”的连接方式（如图 6），解决了“梯度消失”的问题，使得网络性能不会随深度增加而降低。ResNet50 是由 18 个 block 加 1 个卷积层和 1 个全连接分类层组成，每个 block 包括 2 个  $1 \times 1$  卷积核和 1 个  $3 \times 3$  卷积核。

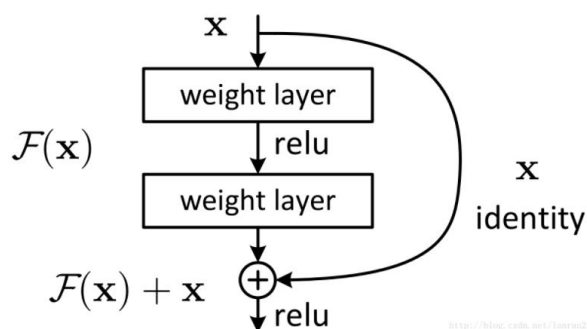


图6 ResNet 的 shortcut connection

与 VGG16 类似，选取 Keras 中预训练的 ResNet50 进行训练和预测。weights 选用“imagenet”上预训练好的 weights，去掉顶层全连接层，对最后一层卷积层应用 global average pooling，添加深度为 10 的分类层，激活函数为 softmax。

模型训练中首先锁住所有卷积层，只训练新添加的分类层，分类层收敛到一定程度后，微调其他感兴趣的层。

### 3.2.3 InceptionResnetV2

InceptionResnetV2 由 Google 团队于 2016 年发布，由早期的 Inception V3 模型[7]变化而来，使用了残差连接，既能使网络更深，也能简化 Inception 块。典型的 Inception 块如图 7 所示。

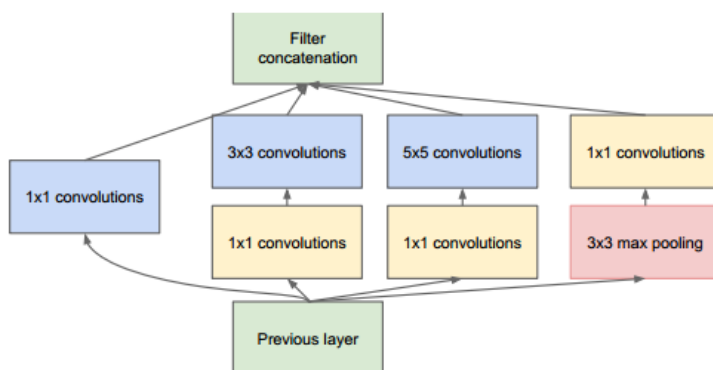


图7 Inception 模块示意图

与上述模型类似，选取 Keras 中预训练的 InceptionResnetV2 进行训练和预测。weights 选用“imagenet”上预训练好的 weights，去掉顶层全连接层，对最后一层卷积层应用 global average pooling，添加深度为 10 的分类层，激活函数为 softmax。

模型训练中首先锁住所有卷积层，只训练新添加的分类层，分类层收敛到一定程度后，微调其他感兴趣的层。

### 3.2.4 可视化

为了检查最终训练的模型是否正确学习到了用于分类的特征，本项目使用 Bolei Zhou 提出的 Class Activation Mapping 方法[8]对预测结果进行可视化。



CAM 方法可以对任一个 CNN 分类网络生成热力图，这个热力图可以高亮图片中跟最终预测结果相关的区域，即可以解释模型到底是基于什么区域进行的预测。具体如图 8 所示。

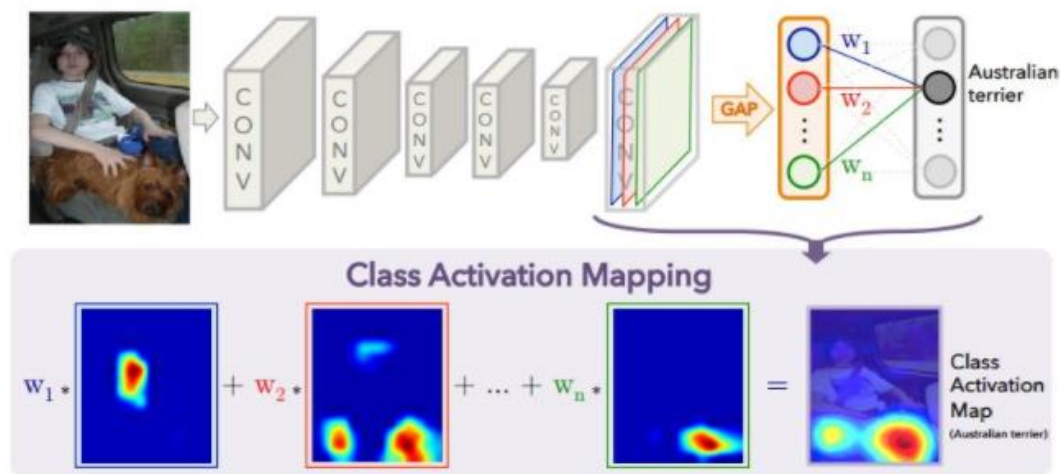


图 8 Class Activation Mapping 原理图

### 3.3 完善

初始模型使用的超参数如下：

- 1) 第一阶段 epoch1=5，第二阶段 epoch2=10；
- 2) 第二阶段 VGG16 和 ResNet50 均放开后 11 层（只算卷积层和全连接层）；
- 3) KFold=5；
- 4) batch\_size=32；
- 5) random\_state=2018；
- 6) 优化器使用 SGD，学习率 0.0001。

训练过程中发现该参数存在很多问题：

- 1) 第一阶段训练过程中 loss 上下跳动，第二阶段训练完后 loss 仍然较高；
- 2) 对于 VGG16 和 ResNet50，没有防止过拟合的措施，对于 InceptionResNetV2 训练完 10epoch 之后仍然欠拟合；

针对以上问题，提出如下改进方案：

- 1) 将第一阶段 epoch1 设为 1。

实际训练过程中发现在 1 个 epoch 周期内训练集 loss 先下降后上升，说明在锁住其他层的情况下，1 个 epoch 就造成了过拟合；

- 2) 增加 EarlyStopping 防止过拟合，同时加入 ModelCheckpoint 保存最佳模型；

由于训练 CNN 时并不知道具体应该训练多少 epoch 才能达到最好的拟合效果，因此可以引入 EarlyStopping 回调函数，在模型开始发生过拟合时终止训练。为避免在训练初期抖动时停止训练，设置监视“val\_loss”的 patience 为 4，表



示经过 4 个 epoch 后 val\_loss 仍不下降则停止训练。

为了记录最佳拟合状态下的参数，回调函数中还应该加入 ModelCheckpoint，将 save\_best\_only 值设置为“True”，只记录最好的模型参数。否则最后保存的模型参数是 EarlyStopping 最终停止时的参数。

3) 优化器改为 Adadelta，加快收敛速度；

Adadelta 对学习率进行自适应约束，训练速度很快。

4) 第二阶段放开所有的层。

由于本项目的输入数据大部分区域是类似的，而且与 ImageNet 的数据相差非常大，预训练模型学习到的底层特征并不太适合本项目，因此应该放开所有的层进行微调。

## 4 结果

### 4.1 模型的评价与验证

1) VGG16

VGG16 模型经过 fine-tune 之后，上传至 kaggle，得到的 private score 为 0.44076，排名 286/1440。相较于优化前的 1.18945 得分，有较大提升。

VGG16 使用 SGD 作为优化器，学习率为 0.0001，并未使用 EarlyStopping。

2) ResNet50

ResNet50 预测结果上传至 kaggle 后，得到的 private score 为 0.66034，将第二阶段的 epoch 从 10 增加至 20 之后，loss 降低至 0.63863，排名 388/1440。

ResNet50 使用 SGD 作为优化器，学习率为 0.0001，并未使用 EarlyStopping。

3) merge VGG16 & ResNet50

通过对比 VGG16 和 ResNet50 的预测结果，发现两个模型预测错误的图片是不一样的，而且通常都是因为有几个结果预测概率相近导致的错误，因此考虑将两个模型最终预测的结果进行综合，可能得到更准确的预测结果。

直接对两个模型的预测结果取平均值，得到融合后的结果，上传至 kaggle 得到的 private score 为 0.43975，排名 287/1440，融合后的模型得分略微提高，但是效果几乎可以忽略不计。

4) InceptionResnetV2

运用上文提到的所有优化策略，使用 Adadelta 作为优化器，加入 EarlyStopping 防止过拟合，将最终的预测结果上传 kaggle，得到的 private score 为 0.18683，排名 49/1440，进入 Top 5%，达到了本项目的预期目标。

下图 9 所示，是 3 个模型在交叉验证过程中第 2/5Folds 的 accuracy 和 loss 走势，说明采用 K 折交叉验证，对 K 个模型预测值取平均作为最终预测结果，在测试集上表现的效果更好。Adadelta 相对于 SGD 可以在更短的 epoch 上达到最

佳拟合状态。

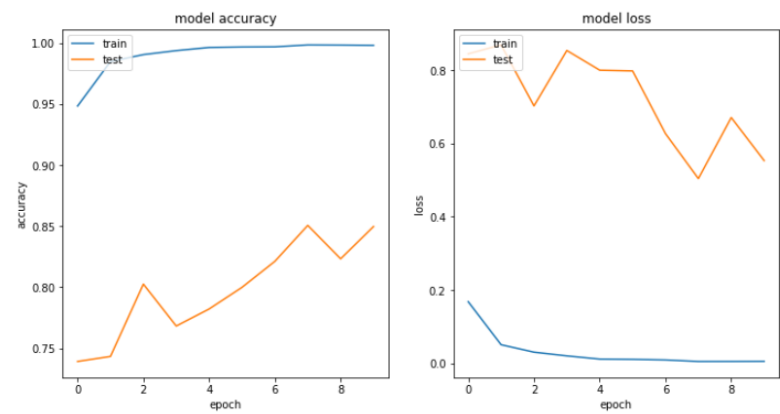


图 9-1 VGG16 训练结果

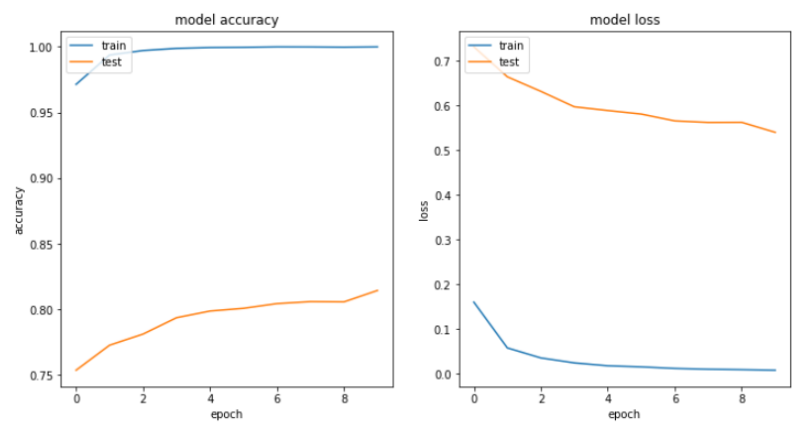


图 9-2 ResNet50 训练结果

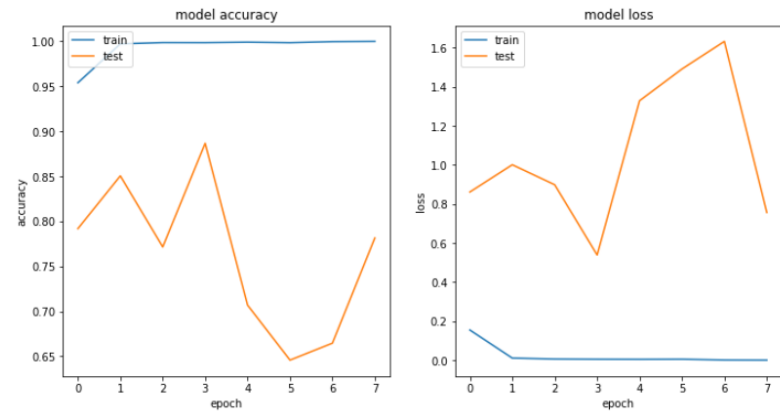


图 9-3 InceptionResnetV2 训练结果

4.2 合理性分析

下表 1 是所有训练模型在 kaggle leaderboard 上的得分结果。表 2 是模型在 1 个 epoch 上的训练和预测时间统计，测试的机器为 AWS EC2 的 p3.2xlarge。从得分上看，InceptionResnetV2 表现最好，但是训练时间是其他两个模型的 3 倍，结合本项目的实际应用场景，更关注的是模型的准确率和预测的速度，而且通常情况下，模型的训练是离线进行的，考虑到本项目中 InceptionResnetV2 预测单

张图片的时间也仅为 6ms，几乎不会察觉到延时。综合考虑，InceptionResnetV2 最适合作为本项目的最终模型。

表 1 不同模型在 kaggle leaderboard 上的结果

模型	Private score	Public score
VGG16（原始）	1.18945	1.14537
VGG16（优化后）	0.44076	0.44148
ResNet50	0.63863	0.66786
merge VGG16+ResNet50	0.43975	0.44427
InceptionResnet V2	0.18683	0.18962

表 2 不同模型训练和预测耗时

模型	训练耗时		预测耗时
	1 epoch	1 picture	1 picture
VGG16	102s	6ms	2ms
ResNet50	89s	5ms	2ms
InceptionResnet V2	301s	17ms	6ms

5 项目结论

5.1 结果可视化

利用 CAM 技术，对 ResNet50 和 InceptionResnetV2 分别进行可视化分析。载入的模型是 K 折交叉验证中保存的第一个模型，分析结果如下图 10 所示，对于同一个图片（c0，正常驾驶），两个模型都是通过识别出双手放在方向盘上从而实现的正确分类，但是 InceptionResnetV2 判断为正常驾驶的概率为 0.9221，远高于 ResNet50，说明 InceptionResnetV2 模型确实表现更优。

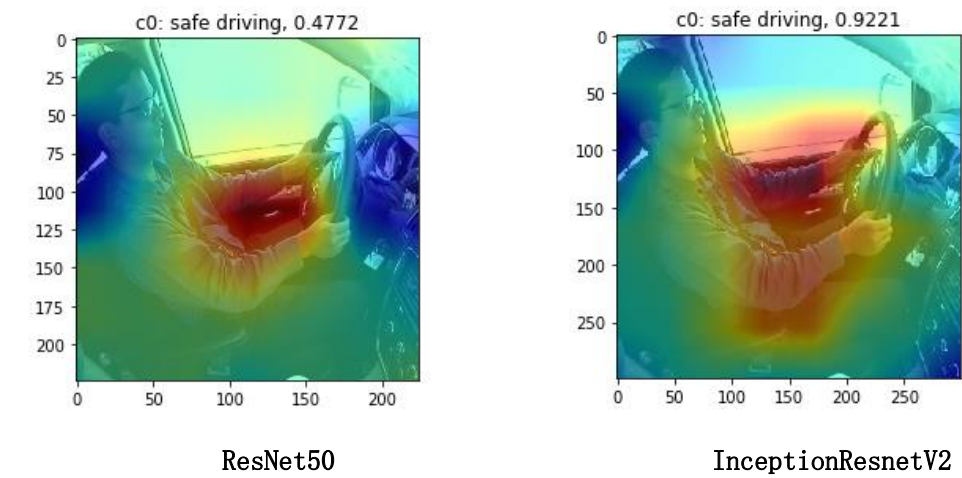


图 10 不同模型 CAM 可视化结果

下面检查一些预测正确的图片，如图 11 所示，模型非常准确的判断出了目标

特征，包括水杯、手机和调收音机的手。

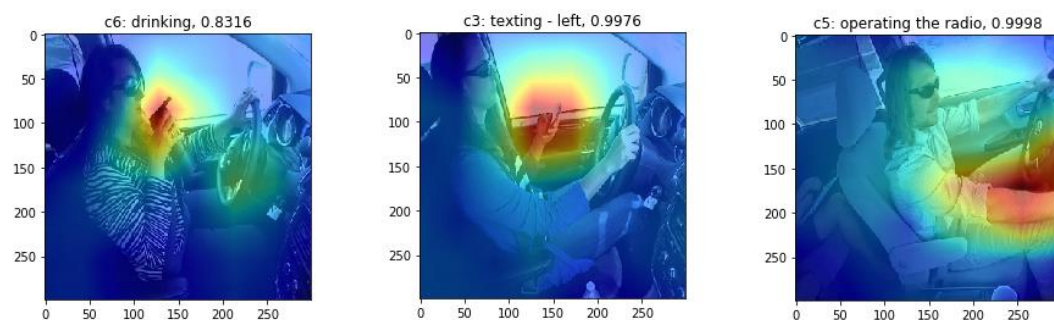


图 11 InceptionResnetV2 模型 CAM 可视化结果

对于分类错误的照片，如图 12 所示，正确分类应该是 c8(整理头发和化妆)，但是模型判断为 c4(左手打电话)。这张图比较有迷惑性，司机左手朝上，但是并没有拿着手机。模型检测到了左手臂朝上，但是其判断的主要依据是车门上的某处，说明可能是过拟合，模型学习到的是额外的、不应该学习的特征。

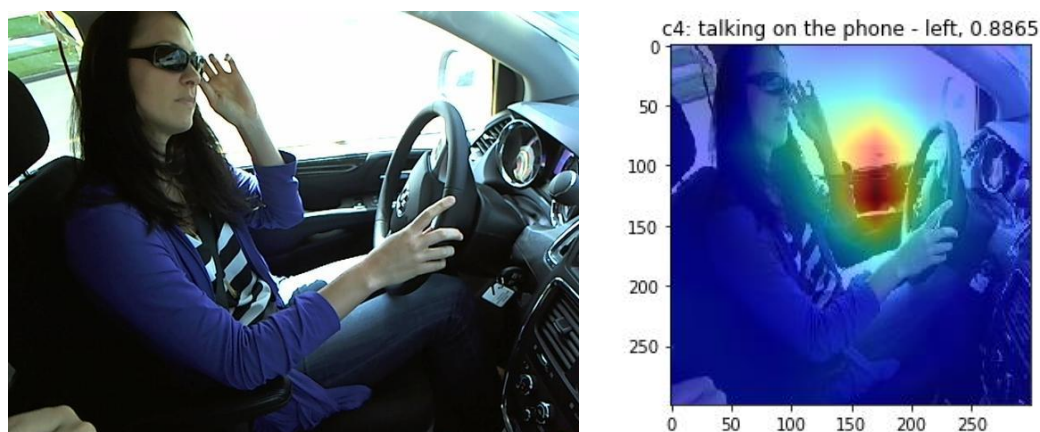


图 12 分类错误的 CAM 可视化结果

为检查模型的泛化性，随机从网上下载数据集中没有的图片进行检测，图片的尺寸也与数据集中不同，可视化结果如图 13。在拍摄角度和尺寸均与数据集有明显差异的情况下，模型均正确的进行了分类，说明本项目训练的 InceptionResnetV2 模型具有较好的泛化性。虽然预测的概率都不高，考虑到使用的模型是 K 折中的一个模型，如果使用 K 个模型进行综合，得到的结果会有较大的提高。

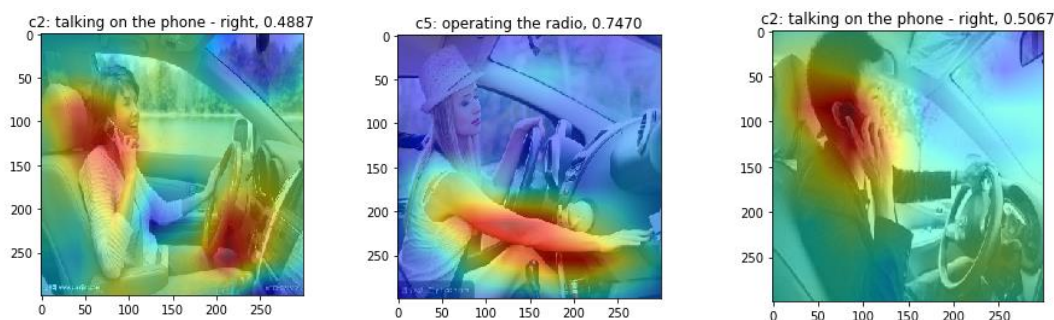


图 13 对外部图片预测的 CAM 可视化结果

## 5.2 对项目的思考

本项目首先对数据集进行分析，了解训练集中的数据分布状态，然后使用 VGG16 模型对数据进行测试，根据初步的结果，明确了模型优化的方向。然后使用优化后的方法，利用迁移学习训练出 VGG16、ResNet50 和 InceptionResnetV2 模型，最后运用 CAM 技术对结果进行可视化分析和检验。

本项目最终使用 InceptionResnetV2 模型取得了 Kaggle Leaderboard 上 Top 5% 的成绩，主要原因是：1) 使用了迁移学习，快速构建模型，并获得一个较好的初始权重；2) 使用 EarlyStopping 和 ModelCheckpoint 技术，有效的避免了过拟合；3) 根据司机 ID 划分数据集，使用 K 折交叉验证方法，最终综合 K 个模型的预测结果进行预测，使得模型能够很好的泛化。

## 5.3 需要做出的改进

由于 InceptionResnetV2 模型参数量巨大，受制于训练时间和资源，本项目还有很多改进的空间。

- 1) 本项目只使用了 10 个 epoch，有可能在波动期就停止了训练，可以增加 epoch 继续训练；
- 2) 本项目只使用了 5 折交叉验证，可以增加至 10 折或更多；
- 3) Adadelata 在训练初中期有非常好的效果，但是后期可能陷入局部最小值，可以结合 SGD 优化器，将学习率调低，在 Adadelata 训练结束之后再进一步使用 SGD 作为优化器训练；
- 4) 采用数据增强的方式，对训练集数据进行旋转、平移、对称等变换，增加训练数据量；
- 5) 融合多个模型；

## 5.4 训练的记录

图 14 是在 AWS 上训练的时长，分别是 p2.xlarge 和 p3.2xlarge。


Amazon Elastic Compute Cloud running Linux/UNIX Spot Instances 		
p2.xlarge Linux/UNIX Spot Instance-hour in US East (Virginia) in VPC Zone #1		95.374 Hrs
p3.2xlarge Linux/UNIX Spot Instance-hour in US East (N. Virginia) in VPC Zone #1		21.467 Hours

图 14 AWS EC2 训练记录

图 15 是在 kaggle 上的提交和得分情况。

<a href="#">subm_loss_inception_resnet_v2_r_299_c_299_folds_5_ep_10_20180418_0...</a> 4 days ago by raywww inception-resnet-v2, earlystopping	0.18683	0.18962	<input type="checkbox"/>
<a href="#">vgg_resnet_20180416.csv</a> 6 days ago by raywww vgg16+resnet50, 5folds	0.43975	0.44427	<input type="checkbox"/>
<a href="#">subm_loss_resnet50_r_224_c_224_folds_5_ep_20_20180415_1918.csv</a> 7 days ago by raywww resnet50, 5folds, 20epochs	0.63863	0.66786	<input type="checkbox"/>
<a href="#">subm_loss_resnet50_r_224_c_224_folds_5_ep_10_20180415_0326.csv</a> 7 days ago by raywww resnet50, 5folds-cv, finetune-10epochs	0.66034	0.72573	<input type="checkbox"/>
<a href="#">subm_loss_vgg16_r_224_c_224_folds_5_ep_10_20180414_1522.csv</a> 7 days ago by raywww vgg16, 5fold cv, fine tune all layers-10epochs	0.44076	0.44148	<input type="checkbox"/>
<a href="#">subm_loss_vgg16_r_224_c_224_folds_1_ep_10_20180413_0535.csv</a> 9 days ago by raywww 1/5 KFold model, no merge, 10 epochs	1.75570	2.04126	<input type="checkbox"/>
<a href="#">subm_loss_vgg16_r_224_c_224_folds_5_ep_5_20180411_1009.1.csv</a> 11 days ago by raywww vgg16, kfold=5, fine-tune	1.18945	1.14537	<input type="checkbox"/>

图 15 kaggle 提交记录

## 6 参考资料

- [1] State Farm Distracted Driver Detection, <https://www.kaggle.com/c/state-farm-distracted-driver-detection>.
- [2] Distracted Driving, Centers for Disease Control and Prevention, [https://www.cdc.gov/motorvehiclesafety/distracted\\_driving/](https://www.cdc.gov/motorvehiclesafety/distracted_driving/).
- [3] Convolutional neural network, [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network).
- [4] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.
- [5] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C]// Computer Vision and Pattern Recognition. IEEE, 2016:770-778.
- [6] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning[J]. 2016.
- [7] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[J]. Computer Science, 2015:2818-2826.
- [8] Zhou B, Khosla A, Lapedriza A, et al. Learning Deep Features for Discriminative Localization[J]. 2015:2921-2929.