

侦测走神司机

机器学习纳米学位毕业项目

张博 2018年3月16日

1. 问题的定义

1.1 项目概述

这个项目名为侦测走神司机，项目来自于数据分析竞赛网站Kaggle，是一个两年前的老项目，全称为State Farm Distracted Driver Detection^[1]。是由美国最大的车险公司State Farm发起的。据汽车安全部门统计，有五分之一的车祸是因为司机分心引起的。分心有可能是因为打电话，发短信等行为引起的，这些很小的细节对交通安全有着很大的影响。该项目的愿景是希望通过摄像头来自动检测驾驶员的行为，从而改善司机的驾驶状况，减小安全隐患并更好地为他们提供保险。



图1 走神司机示例照片

该项目从汽车保险这一市场的实际需求出发，在目前流行的大数据环境下，利用计算机视觉和机器学习算法等技术，搭建驾驶员行为检测模型，从而实现一定的商业价值。

1.2 问题陈述

该项目为我们提供了10种类型的照片，照片的类型如下：

- 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

示例图片如下：



图2 司机分分类示例照片

我们要解决的问题是建立一个机器学习模型，通过已有的标注照片进行训练，然后能够准确的判断出测试照片的类型。

这是一个监督学习领域的计算机视觉问题，属于分类问题。我们可以利用深度学习方法来建立预测模型。

1.3 评价指标

该项目的评估指标我采用Kaggle比赛中的Private Leaderboard得分。该得分是利用服务器上测试集的真实标签，对69%比例的测试集计算得到的多类别对数损失，公式如下：

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

式中， N 为样本数量， M 为类别数量， y_{ij} 为样本 i 是否实际属于类别 j 的0-1指示函数， p_{ij} 为样本 i 在类别 j 上的预测概率。

2. 分析

2.1 数据的探索

该项目的输入数据可以从Kaggle比赛界面下载。输入数据分为训练照片和测试照片，所有照片的尺寸均为640*480。训练照片共有22424张，并且已经标注好类别和司机的ID。测试照片共有79726张。

通过观察测试照片我们可以发现司机与训练照片不尽相同且测试照片远多于训练照片，所以为了保证训练模型可以在测试照片中能够有更好的泛化能力，我们在训练时需要按照不同的司机来分离训练集和验证集。

2.2 探索性可视化

我们对数据进行了可视化分析。详细代码见 `data_distribution.ipynb`。

训练照片按类别分布如下图所示：

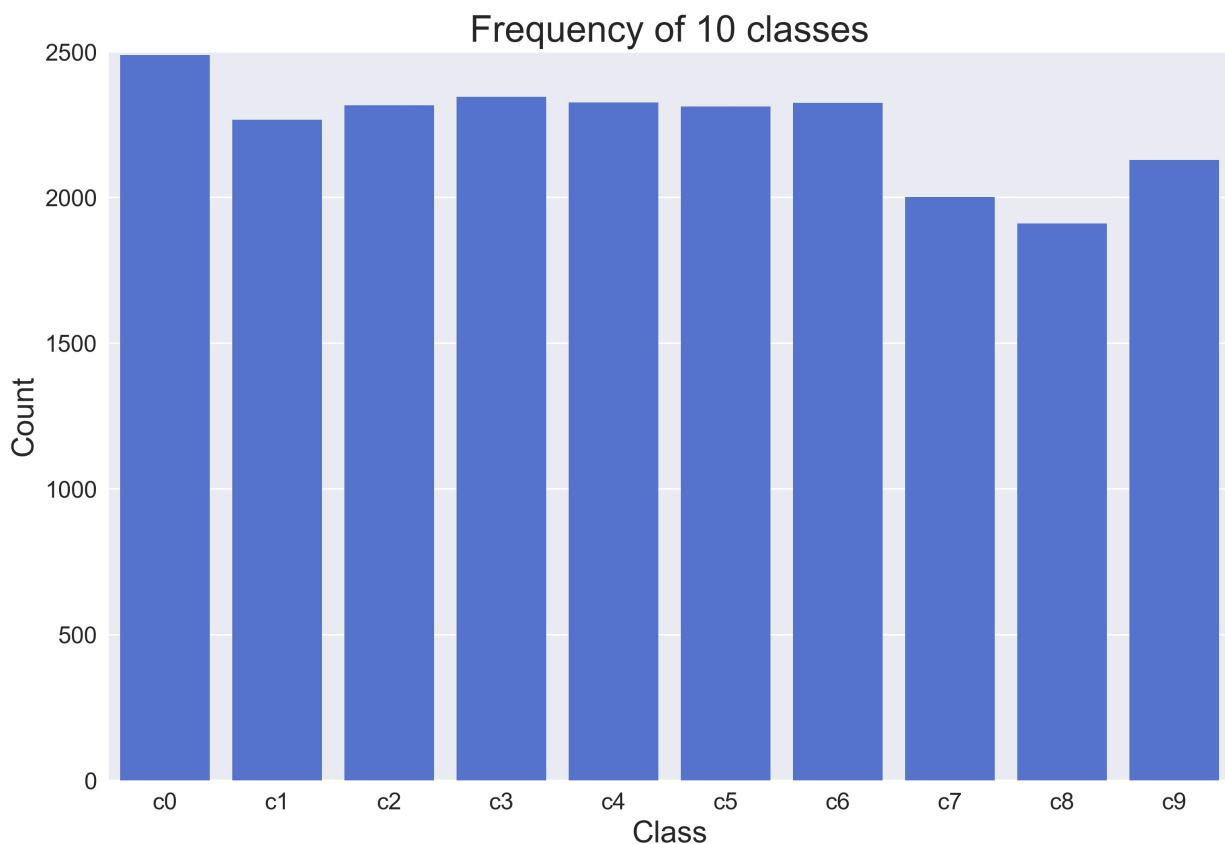


图3 司机类别分布图

从图中可以看出，10个类别的图片分布比较均匀，大多集中在2000-2500之间。

训练数据中共有26位司机，我们按不同司机做了类别分布图，如下：

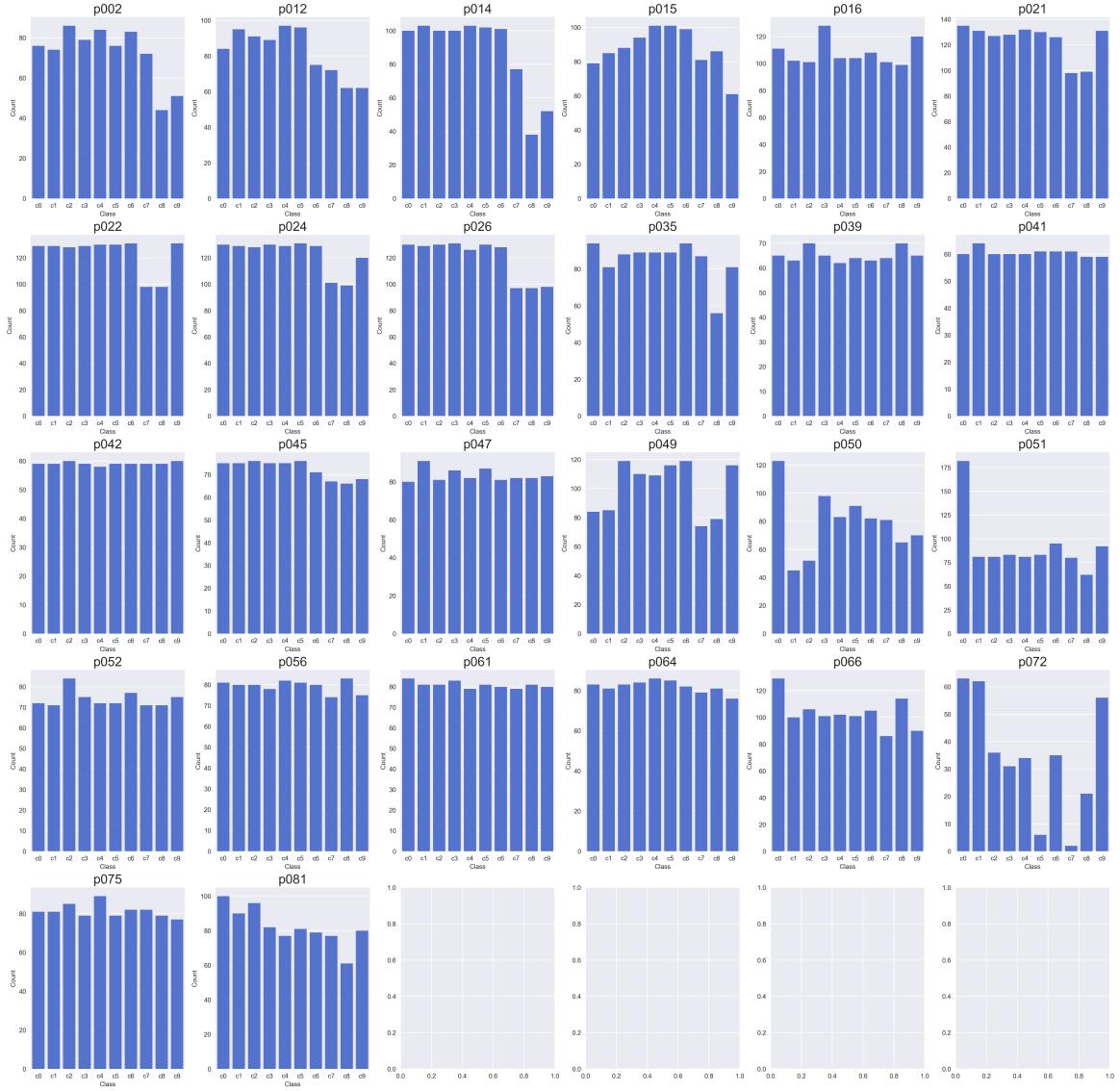


图4 单个司机类别分布图

从图中可以看出，绝大部分的司机的类别分布都比较平均，但个别司机比如p050, p051, p072的类别分布差异比较大，所以在考虑按司机划分训练集和验证集时，需要进行多次划分以保证训练出稳定的模型。

2.3 算法和技术

该项目属于机器视觉领域，通过训练集建立识别人类行为的分类模型，属于机器学习的监督学习范畴。虽然传统的监督学习分类算法有很多，但对于图片来说，由于状态空间巨大且状态之间存在局域的关联，所以传统的监督学习分类算法并不适用。

就目前来说，最有效的图片分类算法是卷积神经网络。卷积神经网络最大的特点和优势在于其层与层之间通过滤波器实现局部连接和权值共享，从而减少网络的参数，有利于建立深层的网络模型来提取更加抽象的特征。

近几年来随着计算能力的提升，涌现出很多卷积神经网络模型，他们在各种图像识别竞赛上都取得了突破性的表现。包括 LSVRC-14 大赛上提出的 VGG16 模型^[2] 和 GoogeLenet 模型^[3]、LSVRC-15 大赛的冠军 ResNet 模型^[4] 等等。由于这些模型已经在大型数据集上进行了训练并且得到了很好的结果，我们利用这些已有的训练好的模型，稍加改动使用在其他的类似的分类任务上通常也会取得不错的结果，这便是我们在后面主要使用的迁移学习的思想^[5]。

在项目中我们所使用的预训练模型主要是以下三个：ResNet50，InceptionV3，Xception。

该项目中我们主要使用 Scikit-learn 和 Keras 进行数据的处理和模型的构建与训练。所使用的硬件环境是 AWS 的 AWS EC2 p2.xlarge 实例^[6]。

2.4 基准模型

该项目在 Kaggle 上已经结束了，Leaderboard 上一共有 1440 组参赛者，我的目标是分类结果能排在 Private Leaderboard 的 Top5%，也就是 loss 要低于 0.20805，争取能进入前 50 名。

3. 方法

3.1 数据预处理

在建立机器学习模型之前，我们需要对司机的图片进行预处理以满足我们的训练要求。

训练集的图片尺寸均为 640*480，而我们所选择的三个预训练模型默认的输入尺寸分别是 224*224 (ResNet50)，299*299 (InceptionV3 和 Xception)。综合考虑保持原图尺寸比例和减轻计算压力，我们选择 ResNet50 模型的输入图片尺寸为 240*320，InceptionV3 和 Xception 模型的输入图片尺寸为 330*440。所有模型均使用 Keras 内各个模型自带的 preprocess_input 函数进行预处理。

3.2 执行过程

在图片输入到训练模型之前，考虑到训练图片数量相对较少，且任务需要泛化到其他未见过的司机图片上，所以这里采取两个策略来提升训练和泛化效果：数据增强和以司机 id 为依据的交叉验证。

数据增强

在这里仅使用较为简单的数据增强，采用 Keras 自带的图片预训练函数 ImageDataGenerator 进行数据增强，具体的参数如下表所示：

参数	含义	取值
----	----	----

rotation_range	随机旋转度数	10
width_shift_range	随机水平移动	0.1
height_shift_range	随机垂直移动	0.1
shear_range	剪切强度	0.1
zoom_range	随机缩放范围	0.1
channel_shift_range	随机通道转换	10

交叉验证

为了提升模型的泛化能力，我们将训练数据按司机 id 分为训练集和验证集。为保证训练集的图片数足够多，验证集的司机总数选为3，其余23位司机图片归为训练集。考虑到验证集选取的特殊性，我们将图片进行五次训练集验证集划分，每次划分均训练一个模型。每次划分的验证集司机 id 如下：

划分序号	验证集司机id
1	p002, p012, p014
2	p021, p022, p024
3	p039, p041, p042
4	p049, p050, p051
5	p061, p064, p066

预训练模型

该项目我们选择三个预训练模型进行训练。

ResNet50。ResNet 模型由 He 等人首先提出。ResNet 模型赢得了 ILSVRC 2015 竞赛，其测试结果达到了 3.57% Top-5 error，历史上第一次超过了 5% Top-5 error。它具有很深的网络结构，其中最显著的特征便是使用了 Residual Block，将输出跨层传播，从而使得输出变为对原有输入的一个微小改动，进而提升优化效果，有利于训练非常深的网络。

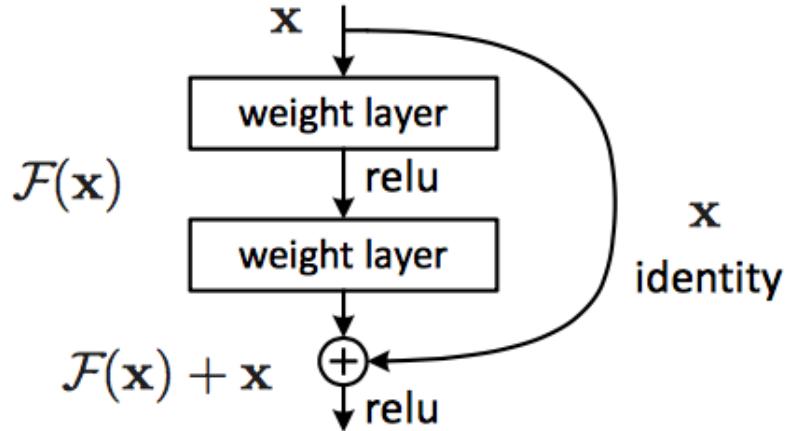


图5 Residual Block 示意图

InceptionV3。Inception V3 模型来源于 Szegedy 等人在 2014 年提出的 Inception 模块。该模块使用多种不同的 filter 提取特征然后融合在一起，作为一个输出。InceptionV3 模型是对 Inception 模块的更新，进一步提高了分类的效果。

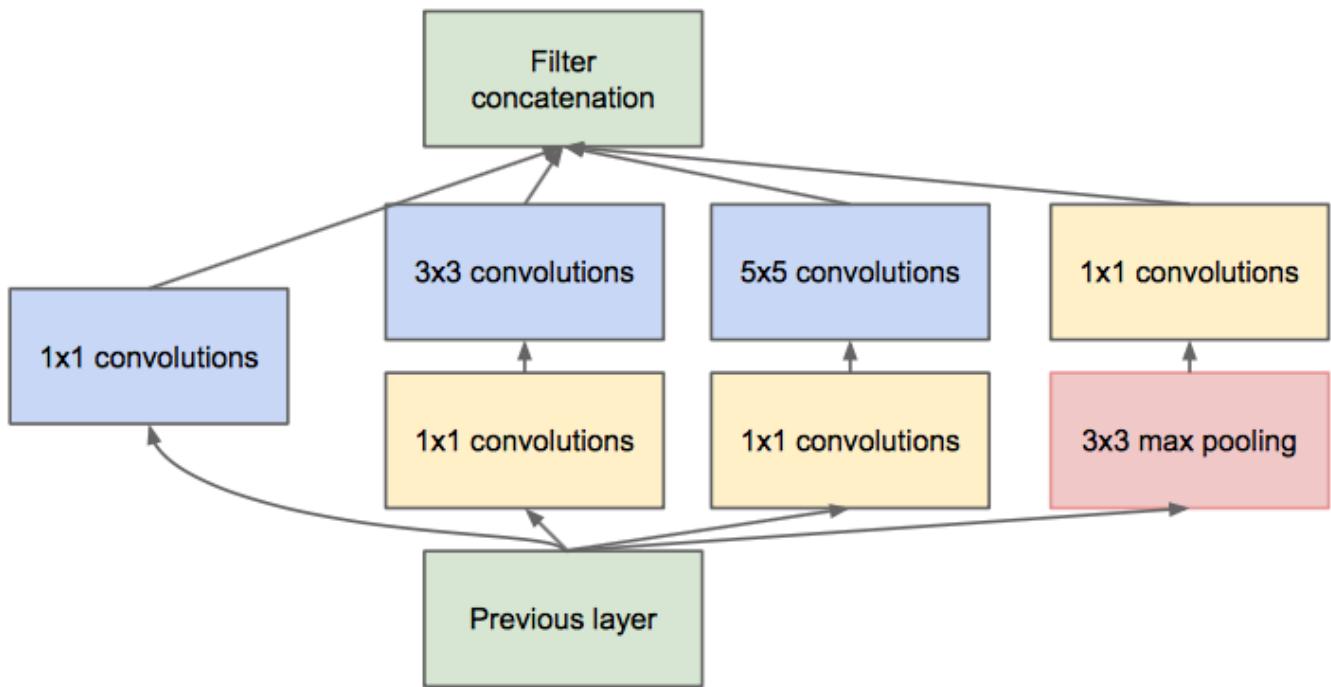


图6 Inception module 示意图

Xception^[7]。Xception 是由 François Chollet 提出的。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。

在实现过程中，所有模型均使用 Keras 自带的模型，模型的 weight 使用 'imagenet' 预训练 weights，全部去除末端的全连接层，然后添加全局平均池化层，Dropout 层(0.5)，Dense 层(10 个节点)，设置 softmax 激活函数作为 10 类的分类概率。

模型搭建起来后，便可以使用训练集数据进行训练。训练过程当中，需要考虑四个参数的选择：batch size, epoch, fine-tune的层数和optimizer。其中 batch size 会根据训练模型的深浅来决定。一般越大越好，除非内存不足。Optimizer 这里选择最常用也是效果最好的 Adam，其中优化器参数除学习率外保持默认值，学习率根据训练效果进行不断的尝试和调节。

在训练过程当中，我们首先对 fine-tune 的层数进行了尝试。根据初步的训练结果，我们发现 fine-tune 的层数越深，模型越容易训练到很高的精度，虽然训练时间稍长，但最终验证集的表现都还不错，所以在训练最终模型时，我们首先固定下来的就是 fine-tune 层数，利用预训练的权重从头开始训练模型。

考虑到从头训练模型的时间花费比较长，所以在训练时为了让模型能够在尽可能短的时间里收敛到不错的效果，我们在训练过程中尝试改变优化器的学习率，既不同的 epoch 采用不同的学习率，学习率越来越小以便模型更快收敛。

实际的训练效果如下表所示，详细代码见 model_train.ipynb。

model		metric	epoch1	epoch2
ResNet50	1	train loss	0.3925	0.0458
		valid loss	0.6859	0.2576
	2	train loss	0.4368	0.0444
		valid loss	0.2395	0.2112
	3	train loss	0.373	0.0485
		valid loss	0.3289	0.1589
	4	train loss	0.3894	0.0396
		valid loss	0.4227	0.3352
	5	train loss	0.3802	0.0442
		valid loss	0.7274	0.3953
InceptionV3	1	train loss	0.4123	0.0364
		valid loss	0.5527	0.311
	2	train loss	0.4088	0.0471
		valid loss	0.6285	0.2401
	3	train loss	0.3948	0.0496
		valid loss	0.1638	0.1353
	4	train loss	0.4381	0.0424
		valid loss	0.3822	0.201
	5	train loss	0.4136	0.039
		valid loss	0.3901	0.3847
Xception	1	train loss	0.3527	0.0331
		valid loss	0.4078	0.291
	2	train loss	0.3693	0.044
		valid loss	0.1697	0.2011
	3	train loss	0.3482	0.0392
		valid loss	0.3003	0.1827
	4	train loss	0.3438	0.0359
		valid loss	0.2942	0.2049
	5	train loss	0.3459	0.0327
		valid loss	0.6162	0.4029

每个模型均使用五个不同的训练验证集进行训练，分别训练五个子模型。对于优化器 Adam，如果训练全新模型一般学习率选取 $1e-3$ ，由于我们采取了预训练模型的权重，所以在第一轮我们使用 $1e-4$ 以便模型尽快收敛。在训练初始时我们一个 epoch 训练一遍所有训练集图片，后来发现效果不好，模型一轮训练下来权重迭代的总次数偏大，模型训练误差很快降到很低但验证集误差很大，出现了过拟合。所以我们采取一轮训练一半图片，到第二轮，将学习率降至 $1e-5$ ，减慢学习速率来减缓过拟合，当第二轮训练完成后训练集误差降到很低且验证集误差也保持在比较优秀的水平，故没有再进行第三轮训练以防出现过拟合。

综合来看，所有模型在第二轮训练完毕后训练集误差均降到了很低的水平，准确率均在 99% 以上，且验证集误差也达到了很高的水平，绝大部分分布在 0.2-0.4 之间，同个模型不同训练验证集整体表现比较稳定，其中 Xception 表现稍好。

3.3 完善

如上所述，在训练模型的过程当中，通过尝试不同的训练层数，训练轮次数和优化器参数，最终平衡了训练效果和泛化效果，用相对较少的轮次达到了相对不错的训练效果。

通过观察训练结果我们发现使用不同的训练验证集结果也不尽相同，由于只将数据进行了一次切分，偶然性会比较大，可能会影响到最终测试的结果，所以这里我们采用集成的思想，将五个不同的训练验证集训练出的模型进行集成，来做最终的测试，集成的方法使用平均值，这样做集成可以避免单个模型只学习到了部分图片的特征的这一缺陷，使最终的整体表现更加稳定，泛化能力更强。

4. 结果

4.1 模型的评价与验证

15个模型的单个模型测试结果以及每5个模型的集成结果如下表所示：

model		private score	public score
ResNet50	1	0.26328	0.29176
	2	0.34414	0.42316
	3	0.31723	0.42185
	4	0.34448	0.41924
	5	0.32193	0.35739
	merge	0.21976	0.26484
InceptionV3	1	0.25724	0.26612
	2	0.27961	0.36347
	3	0.26533	0.33615
	4	0.25605	0.29532
	5	0.24867	0.31721
	merge	0.19977	0.23922
Xception	1	0.22163	0.22295
	2	0.22532	0.23877
	3	0.20865	0.23083
	4	0.23843	0.24352
	5	0.24001	0.30301
	merge	0.17946	0.18828

从结果可见，15个模型的 private score 表现都不错，基本在 0.2-0.3 之间，但把五个模型进行集成之后，表现更加突出，比任何单个模型的表现都要好出很多，尤其是 Xception 的集成模型，private score 得分为 0.17946，达到了 top 5% 的水平，在 leaderboard 中排名 40/1440，达到了预期的目标。由此可见，集成方法虽然很简单，但在提升模型的鲁棒性和泛化能力方面很有效果。

4.2 合理性分析

相比于初始设置的基准指标 (top 5%，争取进入前50) 最终的结果达到了这一目标，由此可见，利用迁移学习的优势，fine-tune 预训练模型对于识别司机行为可以达到很不错的效果。当然，模型的选取是成功的原因之一，在模型的调试和训练过程当中，训练层数、训练轮数、优化器参数参数的选择等等都对结果有很大的影响，尤其是最后的5个模型集成，对结果的提升帮助很大。

综合最终的 Kaggle 的结果，可以看到我们训练的 15 个模型都可以较为出色的完成项目任务，可以很好的解决项目中设定的问题。

5. 项目结论

5.1 结果可视化

为了更为直观的展示结果，我们引入了 Class Activation Mapping 的方法进行了可视化^[8]。CAM 可以对图片中的物体进行定位，利用物体定位我们可以看出我们的分类模型是否能正确检测到目标分类的图像。CAM 对物体定位的核心思想是，模型的高层卷积层输出的特征图往往记忆了目标分类的位置信息。我们将某深层卷积层的输出特征图，乘以这一层对应分类的权重，再映射回图像空间，这样就得到目标分类的可视化图像。

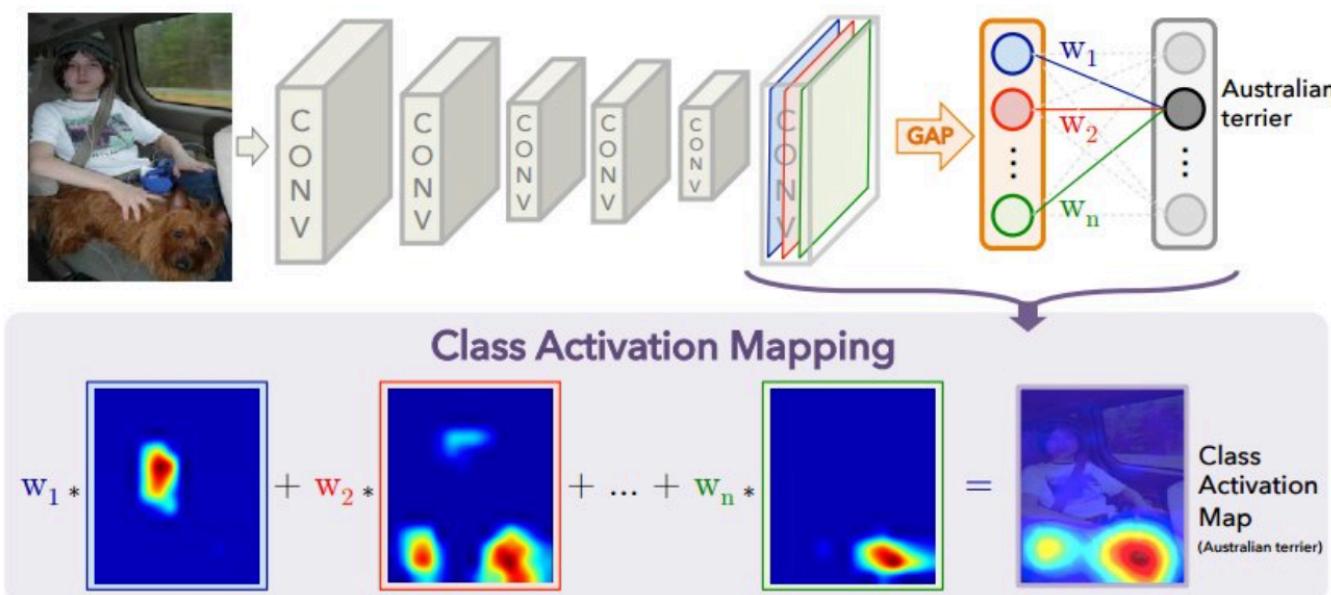


图7 CAM原理图^[8]

下面我们将对之前的分类结果进行 CAM 可视化，并分析其结果。详细代码见 CAM.ipynb。



图8 预测正确示例图

上图是15个模型中表现最好的 Xception 3号模型的可视化结果，三张图片是验证集预测正确的分

类示意图。从图中可以看出，预测正确的图片都可以检测出关键的位置或者物品，比如第一张图片检测出双手放在方向盘上，第二张图片右手拿着手机，第三张图片右手拿着手机放在耳边。



图9 预测错误示例图

上图是三张验证集预测错误分类的示意图。从图中我们可以看到，预测错误的图片均有一定的误导性。第一张图司机的左手在方向盘靠下的位置，难于检测，所以模型认为司机的左手在拿着手机靠近耳边打电话，因为这种情况下司机的左手也很难被检测到。第二张图司机的双手均放在方向盘上，头部向右扭动，如果让我们用肉眼分类，也不确定是在说话还是在看后视镜，模型检测到了驾驶盘上的双手，更倾向于认为是安全驾驶。第三张图司机手里拿着口红在抹嘴唇，模型检测到了关键位置和物品，却错认为是在用右手手机打电话。

可以看到，三张分类错误的图片均有一定程度的迷惑性，这从模型的预测分类概率也可以看出，三个分类的概率为 39.99%，46.59%，50.59%，说明模型也不是很确定是否属于这一类别，这时候模型的集成便可以发挥作用。不同模型的预测能力各有千秋，综合在一起，会使得结果更加具有可信度。

5.2 对项目的思考

本项目我们选取了三个预训练模型 ResNet50, InceptionV3 和 Xception，采用预训练的权重来训练所有层。训练数据使用了数据增强和交叉验证，按司机 id 将数据分类，23 位司机为训练集，3 位司机为验证集，以保证模型有良好的泛化性能。并进行五次不同的划分，每次划分分别训练一个模型。

训练过程当中，为保证模型尽快收敛且不过拟合，每轮只训练一半的数据，且优化器学习率随着轮次的增加而减小，以减缓收敛速度抵抗过拟合。最终模型训练 2 轮就已经达到了不错的收敛水平和泛化水平，优化器 Adam 两轮的学习率分别是 $1e-4$ 和 $1e-5$ 。

最终训练出的 15 个模型表现良好，我们又使用集成的方法，将相同模型的 5 个子模型的预测结果进行集成，最终集成的结果均好于单个模型的预测结果。其中，Xception 模型的集成结果 private score 达到 0.17946，在 leaderboard 中排名 40/1440，进入 top 5%，达到了预期的目标。

5.3 需要做出的改进

由于模型是训练所有层，花费的时间较多，所以只是划分了5次训练验证集，如果再多划分几次然后进行集成，可能效果会更好。

同时，由于时间有限，对于数据增强的方式以及优化器的选择上并未进行更多的尝试，也许会有更好的训练效果。

最后，考虑到集成方法对结果的改善，我们尝试将所有的15个模型进行了集成。不同模型的5个子模型集成可以将数据集的特征学的更加全面，而三个集成模型的学习能力各有千秋，集成在一起结果可能会更好。最终结果如下，详细代码见 merge.ipynb。

merge model	private score	public score
ResNet50	0.21976	0.26484
InceptionV3	0.19977	0.23922
Xception	0.17946	0.18828
merge all	0.18029	0.20451

从结果来看，和 Xception 的集成结果很接近，所以三个模型的集成并未继续提升模型的表现。如果要继续提升模型表现，需要采取其他手段，如引入更加复杂的数据增强方式来扩充训练集数据。

参考文献

1. [State Farm Distracted Driver Detection](#)
2. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
3. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.
4. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.
5. Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng. 2010;22(10): 1345–59.
6. [AWS EC2 p2.xlarge](#)
7. François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv preprint arXiv:1610.02357, 2016.

8. B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In In Advances in Neural Information Processing Systems, 2014.