机器学习工程师纳米学位毕业项目
基于深度学习的走神司机探测

陈亮

2018 年 7 月 7 日

目录

1. 定义

1.1. 项目概述

随着生活水平的提高，车辆日益剧增，交通事故数量逐年增加，造成了巨大人员伤亡和财产损失。根据中华网汽车安全部门的数据，五分之一的车祸是由一名分心的司机引起的，这意味着每年有 425,000 人受伤，3000 人因分心驾驶而死亡。

那么造成这样现象的原因是什么，主要有因为司机疲劳驾驶，或者走神去做其他事情，想象身边的例子，开车时候犯困，开始时候打电话，发短信，喝水，拿后面东西，整理化妆的都有。这对道路安全和行车效率形成了极大的影响。

为了改善这一令人吃惊的数据， State Farm 希望通过仪表板摄像头能够自动检测驾驶员分散注意力的行为，并更好地为客户提供保险。给定 2D 仪表板相机图像的数据集，State Farm 正在挑战 Kagglers 对每个驾驶员的行为进行分类。 他们是否专心驾驶，系好安全带，还是在后座与朋友一起拍照？

1.2. 问题描述

我们要做的事情，就是根据车内摄像机的画面自动检测驾驶员走神的行为。如果是安全驾驶则一切正常，如果有走神行为，给予警报提醒。

根据驾驶员常见驾驶动作，将将驾驶员的状态归纳为以下十种:

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

这是一个有明确分类的监督学习多分类问题，可以采用卷积神经网络模型来求解。

1.3. 输入数据集

数据集来自 kaggle。数据集包含以下几部分：

https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

训练数据：训练数据根据驾驶员的不同状态，将数据从 c0 到 c9 分为 10 组，每组 2000 个左右，总共 22424 个图片数据，图片大小：640x480

测试数据：测试数据为 79726 张图片数据，同一个驾驶员只出现在训练数据或者测试数

据里。

driver_imgs_list.csv: 一个 CSV 文件，里面为所有训练图片编号，以及其对应的驾驶员、动作编号。

1.4. 评估指标

评估指标采用 Logloss 的评估方式：

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

N：样本数
M：类别数，比如本项目里的类别，M 就为 10
yij：第 i 个样本属于分类 j 时为为 1，否则为 0
pij：第 i 个样本被预测为第 j 类的概率

采用 logloss 的评估方式更能反映模型和算法的能力。如果模型预测结果正确了，$p_{ij} = 1$ => $\log(p_{ij}) = 0$，$p_{ij} = 0.999$ => log(Pij)=-0.001，最后增加的 log 差不多。但如果判断错误，如$p_{ij} = 0$ => log(Pij)= -无穷，$p_{ij} = 0.001$ => log(Pij)=-6.9 也就是判断错误一个，对得分的影响会非常大。log loss 旨在惩罚错误分类，对于完全正确的分类(预测概率为 1)显然其对 loss 的贡献为 0。

1.5. 项目目标

我的目标是： logloss < 0.26 ，并且世界排名在前 1/10。

2. 分析

2.1. 数据分析及可视化

训练数据根据驾驶员的不同状态，将数据从 c0 到 c9 分为 10 组，每组 2000 个左右，总共 22424 个图片数据，图片大小都是 640x480。

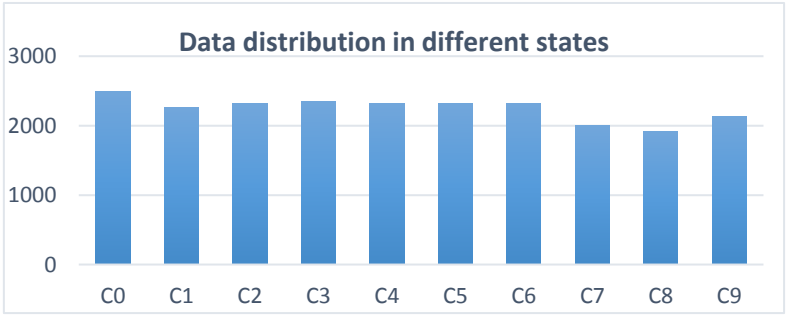

图 1：训练数据中不同状态下的数据分布

从统计结果上来看，从 c0 到 c9 这 10 种状态，每组数量分布还算比较均匀，所以并不需要更改数据分布。

分析训练数据中的司机分布，发现训练数据集一共有 26 个司机，测试数据集有司机若干，且和训练集中的司机并不相同。每个司机所对应的数据大小如图 2.

从统计结果上来看，这 26 个司机数量分布略有差异，但整体还算比较均匀，所以当采用司机进行训练集合验证集的时候，并不需要对数据进行特殊预处理。

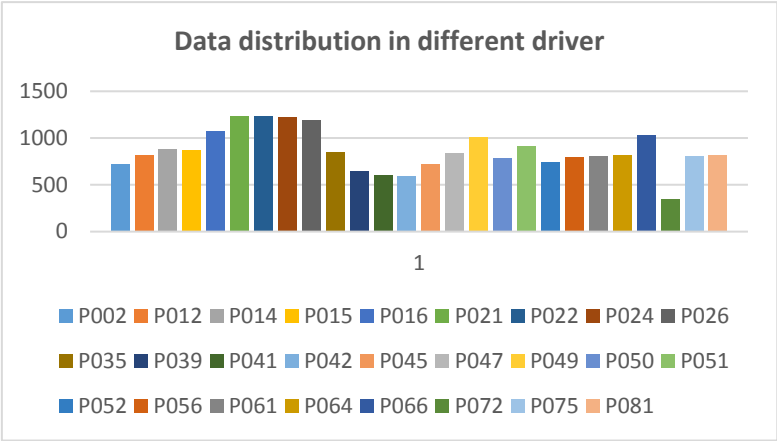

图 2：训练数据中不同司机下的数据分布

测试数据总共有共有图片 79726 张，查看图中图片，司机和训练数据中的并不同，司机更多，场景更多。

2.1.1. 数据抽样并列出

随机抽取数据集中的某种状态展示，这里分别抽取了训练数和测试数据。训练数据中抽取了 26 位司机喝饮料的状态，展示如图 3 所示。测试数据中，随意抽取了一些图片，展示如图 4 所示。

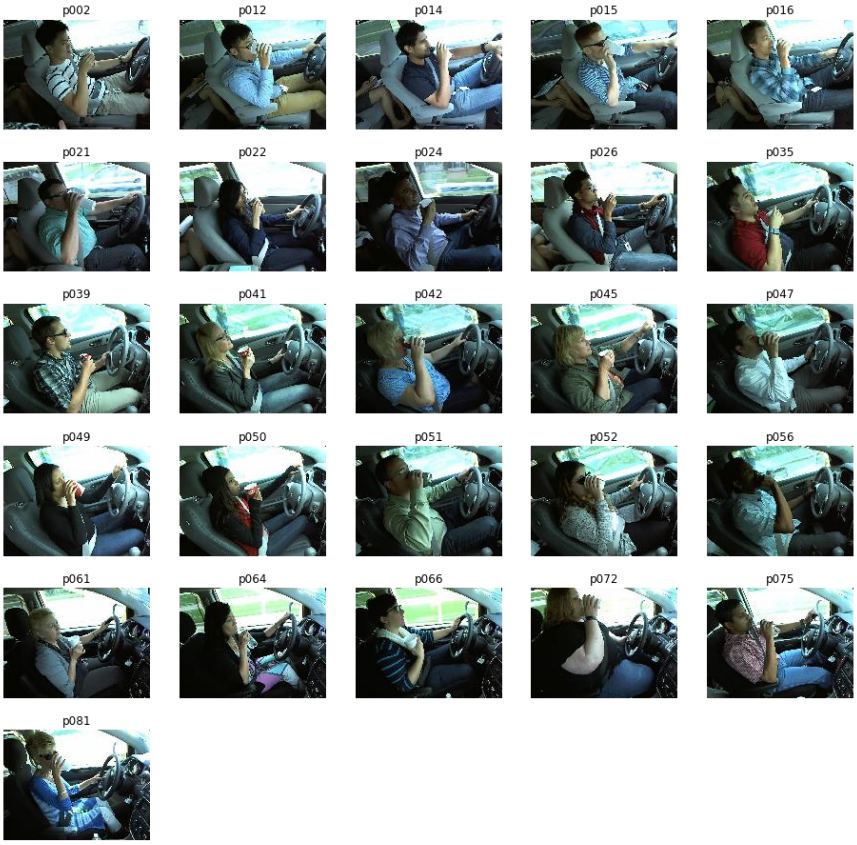

图 3：26 位司机喝饮料状态抽样展示

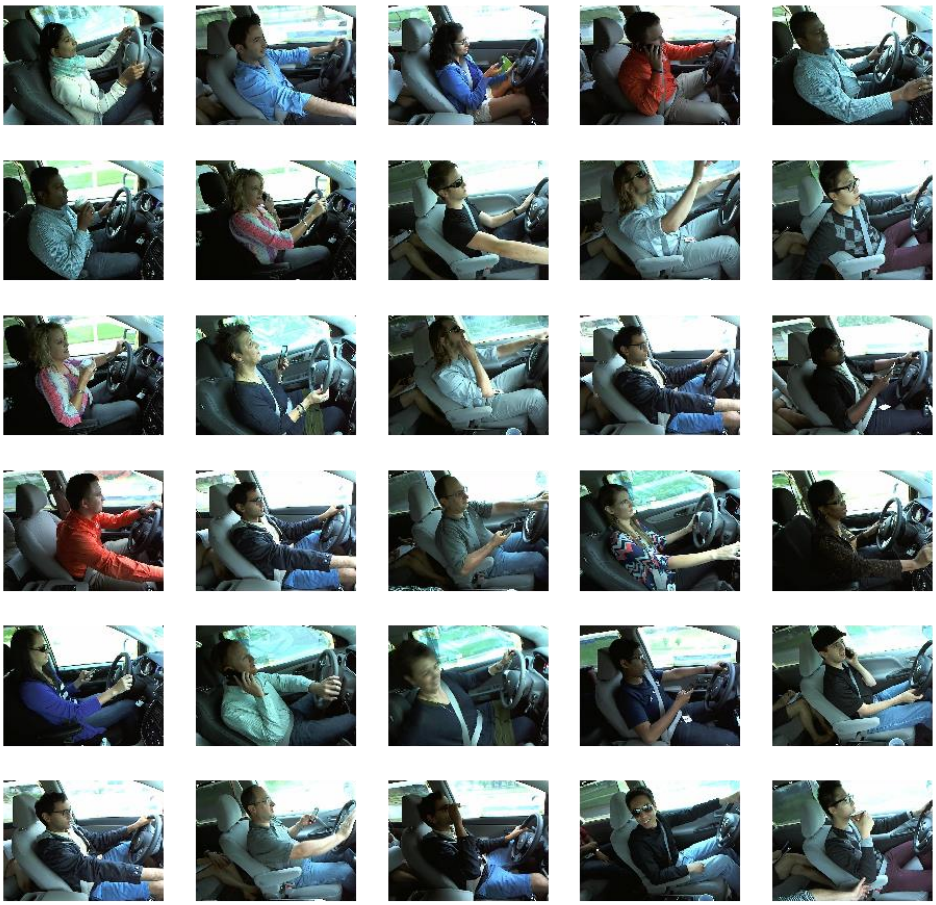

图 4：测试数据随机展示

2.1.2. 数据总结

基于以上数据分析，可以看出以下特点：

1. 训练集和测试集司机并不相同，车应该是一个车或者同样款型的车，摄像头的位置，角度，差不多，但是略有差异。所以后续数据扩增，例如旋转、缩放等可能会有帮助。

2. 数据集是从连续视频里截取出来的，有些截取时机并不妥当，会给训练数据带来干扰，如果能够很好的排出干扰数据，会更有帮助。

3. 训练数据总共只有 26 位司机，虽然每位司机有不同的动作分类，但是同一位司机的外貌、着装相同，很容易导致模型只认司机而忽略动作。所以按照司机进行划分训练集和验证集可能会很有必要。

4. 相对于训练数据，测试数据的图片更多，司机更多，场景更多。这样看来得充分运用训练数据，尽量提高其泛化能力。

2.2. 算法与技术

识别驾驶员到底处于 c0 到 c9 中哪个状态，其实本质上是一个分类问题。一些常用的机器学习分类算法，例如感知机、逻辑回归、支持向量机、决策树，在图片分类上面并不能好好的应用。所以这里采用卷积神经网络，输入是每幅图片的像素所组成的特征向量，输出为状态类别。

2.2.1. 神经网络

神经网络模型工作方式参考人类大脑思维模式， 它由输入层和输出层、 一个或者多个隐藏层构成神经元结构，神经元之间的连有相应的权重， 训练学习算法就是在在迭代的过程中不断的去调整这些权重，从而使得预测误差尽量最小化。

感知机构是最简单的神经元，神经网络是在它的基础上建立起来的。它有若干输入和一个输出。

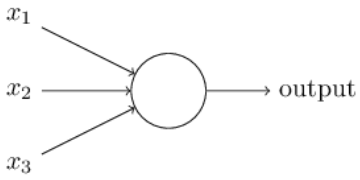

图 5：感知机模型

在感知机的模型上进行扩展，加入一层或多层隐藏层，就形成了神经网络，如下图 6 所示。当然增加了这么多隐藏层模型的复杂度也增加了好多，输出层的神经元也可以不止一个输出，可以有多个输出，这样模型就可以扩展运用到分类回归，以及其他的机器学习领域比如降维和聚类等。

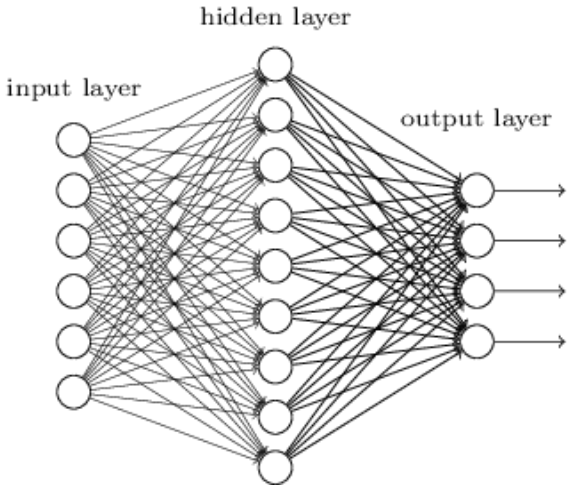

图 6：神经网络模型

神经网络是从感知机的扩展而来，而深度神经网络 DNN 可以理解为有很多隐藏层的神经网络。深度神经网络 DNN 内部不同层，根据其位置可以分为三类，输入层，隐藏层和输出层，如下图 7 所示，一般来说第一层是输入层，最后一层是输出层，而中间的层数都是隐藏层。

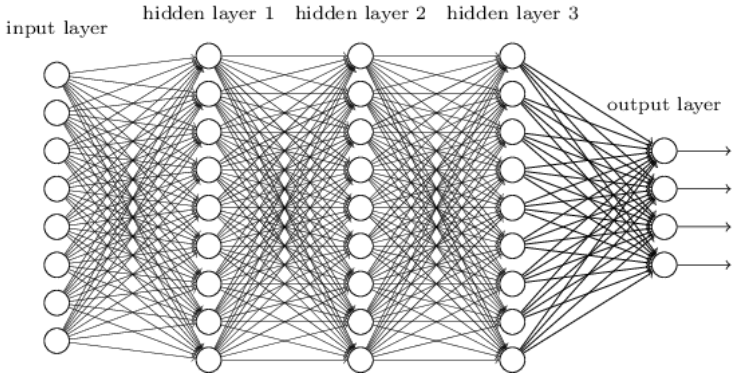

图 7：深度神经网络模型 DNN

2.2.2. 卷积神经网络 CNN

卷积神经网络与普通神经网络的区别在于，卷积神经网络包含了一个由卷积层和子采样层构成的特征抽取器。在 CNN 的一个卷积层中，一般包含若干个特征平面(featureMap)，它们由一些矩形排列的的神经元组成，同一特征平面的神经元共享权值，也就是卷积核。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对卷积核进行更新调整，从而得到合理的权值。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。子采样也叫做池化，通常有均值子采样和最大值子采样两种形式。卷积和子采样大大简化了模型复杂度、参数。卷积神经网络的基本结构如图 8 所示：

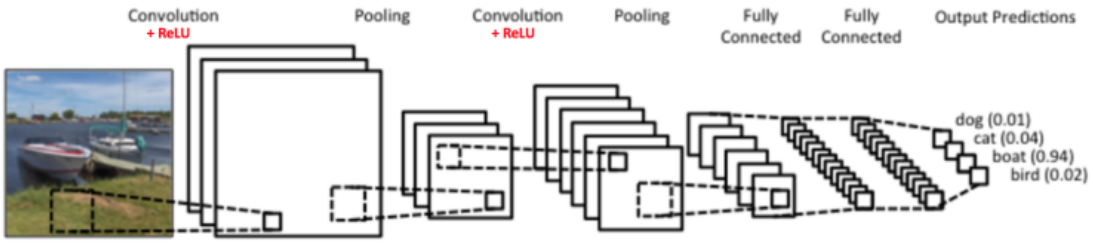

图 8：卷积神经网络基本结构

图中是一个图形识别的 CNN 模型。可以看出最左边的船的图像就是我们的输入层，它为若干个矩阵，接着是卷积层，在卷积层后面是池化层，卷积层+池化层的组合可以在隐藏层出现很多次，在若干卷积层+池化层后面是全连接层，全连接层会采用激活函数来做图像识别的分类，一般采用 softmax。

卷积其实就是对输入的图像的不同局部的矩阵和卷积核矩阵各个位置的元素相乘，然后相加得到。

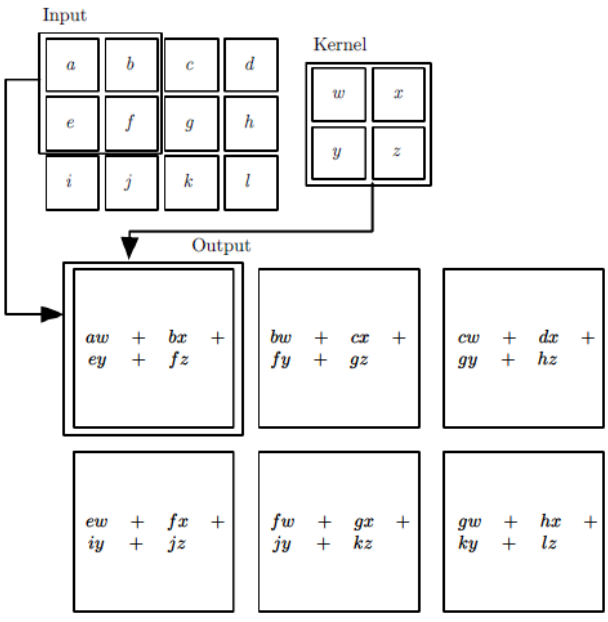

图 9：卷积的算法

图中的输入是一个二维的 3x4 的矩阵，而卷积核是一个 2x2 的矩阵。这里我们假设卷积是一次移动一个像素来卷积的，那么首先我们对输入的左上角 2x2 局部和卷积核卷积，即各

个位置的元素相乘再相加，得到的输出矩阵相应的元素。接着我们将输入的局部向右平移一个像素，同样的方法，我们可以得到输出矩阵的元素。

相比卷积层的复杂，池化层则要简单的多，所谓的池化，个人理解就是对输入张量的各个子矩阵进行压缩。假如是 2x2 的池化，那么就将子矩阵的每 2x2 个元素变成一个元素，如果是 3x3 的池化，那么就将子矩阵的每 3x3 个元素变成一个元素，这样输入矩阵的维度就变小了。常见的池化标准有 2 个，MAX 或者是 Average。即取对应区域的最大值或者平均值作为池化后的元素值。

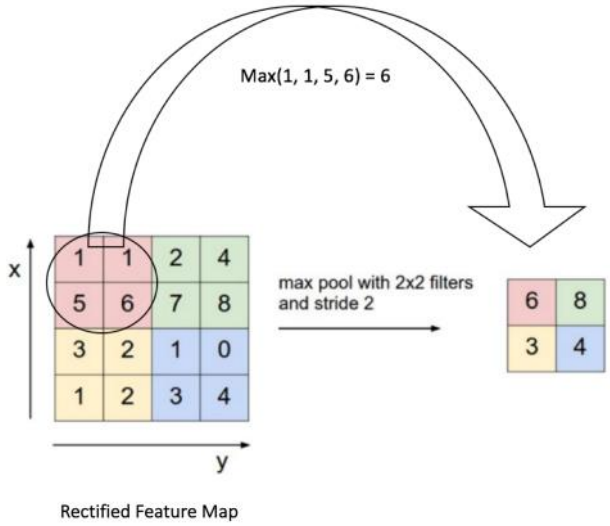

图 9：池化 max 的算法

上图的例子采用取最大值的池化方法。采用的是 2x2 的池化，步幅为 2。首先对红色 2x2 区域进行池化，由于此 2x2 区域的最大值为 6.那么对应的池化输出位置的值为 6，由于步幅为 2，此时移动到绿色的位置去进行池化，输出的最大值为 8.同样的方法，可以得到黄色区域和蓝色区域的输出值。最终，我们的输入 4x4 的矩阵在池化后变成了 2x2 的矩阵，进行了压缩。

2.2.3. 技术

使用 AWS 所提供的 EC2 计算，它不仅具有强大的计算能力，所提供的 AMI 也预装了深度学习的基础包，使用起来方便且高效。项目中采用了 p2.xlarge 和 p3.2xlarge。

p2.xlarge 硬件配置：12G Tesla K80 显卡，4 核 CPU，60G 内存，500M 的网速。

p3.2xlarge 硬件配置：16G Tesla V100 显卡，8 核 CPU，60G 内存，10Gbps 的网速。

系统映像采用 AWS 配置好的深度学习映像 Deep Learning AMI (Ubuntu) Version 10.0。

2.3. 基准指标

在 2016 年 kaggle 比赛中，State Farm Distracted Driver Detection 中有 1440 个队伍参加，Public Leaderboard 第一名得分 0.08689，前十分之一处，第 288 名得分 0.26。本项目的目标就是能够进入前十分之一，也就是 muticlass loss 目标 0.26.

3. 具体方法

3.1. 数据预备处理

由于本项目以 VGG16 为基础模型，首先将图片大小从 640x480 转换为 224x224，以适应模型。然后将图片进行规范化处理，即每层减去平均像素。由于读取过程是按顺序读取的，所以对其进行随机打乱。训练数据经过随机打乱以后，采用 K 折交叉验证来衡量模型分类的性能。

训练数据共有 10 种分类，22424 个样本，采用 5 折交叉验证，也就是说将数据平分成 5 份，每次采用其中 4 份作为训练集，1 份作为交叉验证集。

测试数据比训练数据大很多，总共有 79726 个样本，由于数据量比较大，考虑到存储，将其截成 5 部分进行缓存，然后分部分进行读取和预测，最后将其拼接起来。

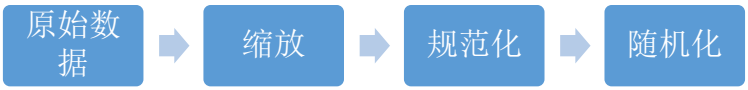

图 10：图片的预处理步骤

3.2. 实现

3.2.1. 模型构建

VGG16 采用 ImageNet 作为训练集，具有很强的分类能力，并且结构简单，所以本项目打算采用其作为基础模型。

此模型总共有 16 层，最后一层为 1000 分类的全连接层，由于本项目针对 10 种分类，所以将原模型中的最后一层去掉，加上 10 分类的全连接层。综合考虑计算时间，以及分类效果，本项目打算将前 15 层导入训练好的权重，只对最后一层进行训练。

优化器采用随机梯度下降算法，随机梯度下降算法学习率采用常见的 1e-3，其衰减系数设置为 1e-6。

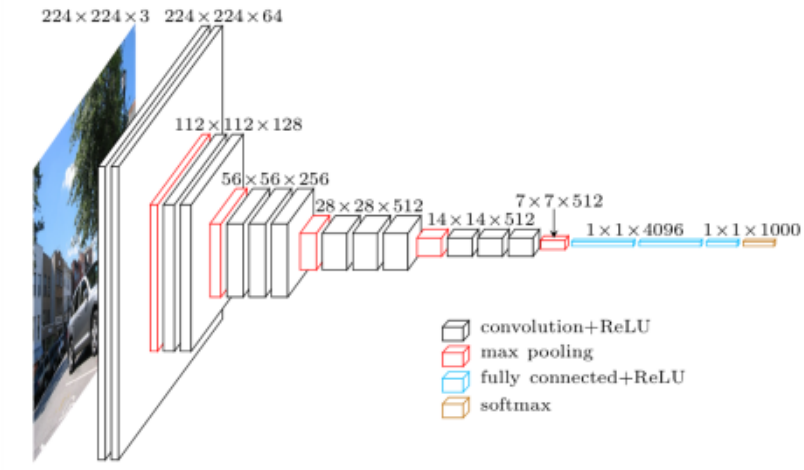

图 11：VGG16 模型结构

3.2.2. 模型训练

为了防止模型过于复杂而引起的过拟合，模型训练采用 K 折交叉验证。 这是一种统计学上将数据样本切割成较小子集的实用方法。于是可以先在一个子集上做分析， 而其它子

集则用来做后续对此分析的确认及验证。 一开始的子集被称为训练集。而其它的子集则被称为验证集或测试集。

这种方法可以充分的利用样本资源，可以把全部训练样本都包含进来，但是比较占用计算资源，需要训练 K 次，考虑到本算例样本较大，每轮训练时间较长，这里的 K 值取 5。

图 11：K 折交叉验证

当数据量过少，不足以对整个数据集进行分布估计，因此往往需要防止模型过拟合，提高模型泛化能力。而为了达到该目的的最常见方法便是 early stopping、数据集扩增（Data augmentation）、正则化（Regularization）、Dropout 等。这里采用了 Dropout 和 early stopping。

Dropout 是在没次迭代中，随机删除一些神经元，下次与上次不一样，做随机选择。这样一直进行瑕疵，直至训练结束，进而来防止过拟合。 这里 Dropout 取值 0.5。

Early stopping 是一种迭代次数截断的方法来防止过拟合的方法，即在模型对训练数据集迭代收敛之前停止迭代来防止过拟合。Early stopping 方法的具体做法是，在每一个 Epoch 结束时计算 validation data 的 accuracy、loss，当 accuracy 不再提高时，或者 loss 不在降低时就停止训练。这里 monitor 采用 val_loss，patience 取 5。既当 loss 连续 5 次没有达到历史最佳就停止训练。

3.3. 结果分析

3.3.1. 模型在训练数据集上的表现

K_num	epoch	Train_loss	Train_vcc	val_loss	val_acc
1	12	6.1923e-05	1	0.0125	0.9982
2	14	3.0759e-05	1	0.0188	0.9964
3	22	1.1027e-05	1	0.0181	0.9971
4	18	5.4773e-05	1	0.0256	0.9958
5	14	0.0018	0.9993	0.0862	0.9844

表 1：K 折交叉验证每折训练表现

从训练结果来看，模型具有很高的准确率，验证集几乎为 1。从 loss 上来看，每轮之间具有一定的波动，最小的为 0.0125，最大的达到了 0.0862,。从收敛轮数来看，最快 12 轮就收敛了，最慢的 22 轮收敛，没有达到模型设置的 25 轮。

图 12：第 1 折训练集和验证集 loss、accuracy 随 epoch 走势

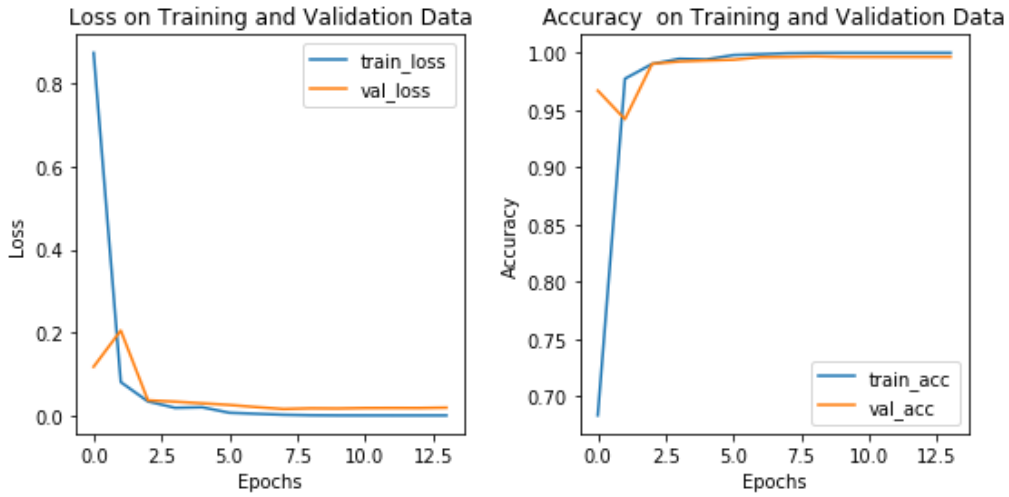

图 13：第 2 折训练集和验证集 loss、accuracy 随 epoch 走势

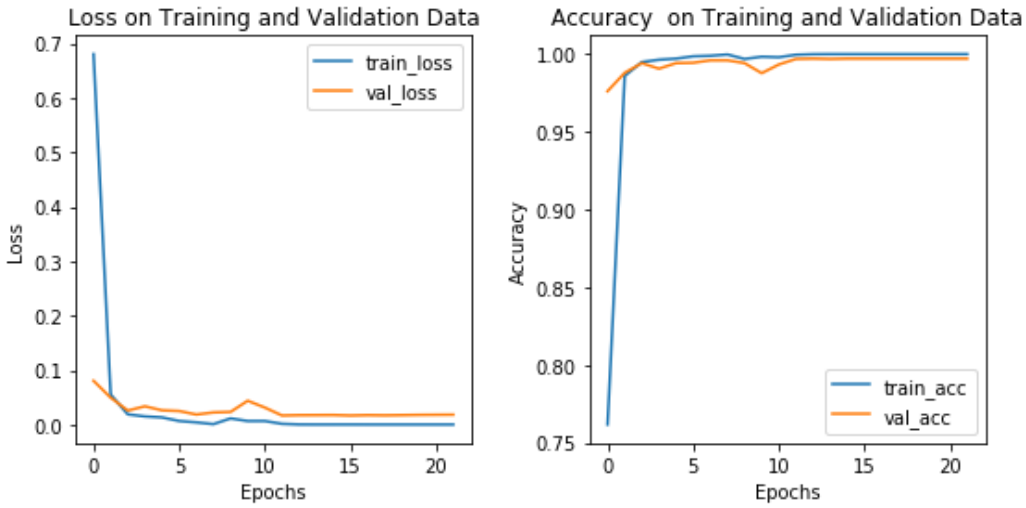

图 14：第 3 折训练集和验证集 loss、accuracy 随 epoch 走势

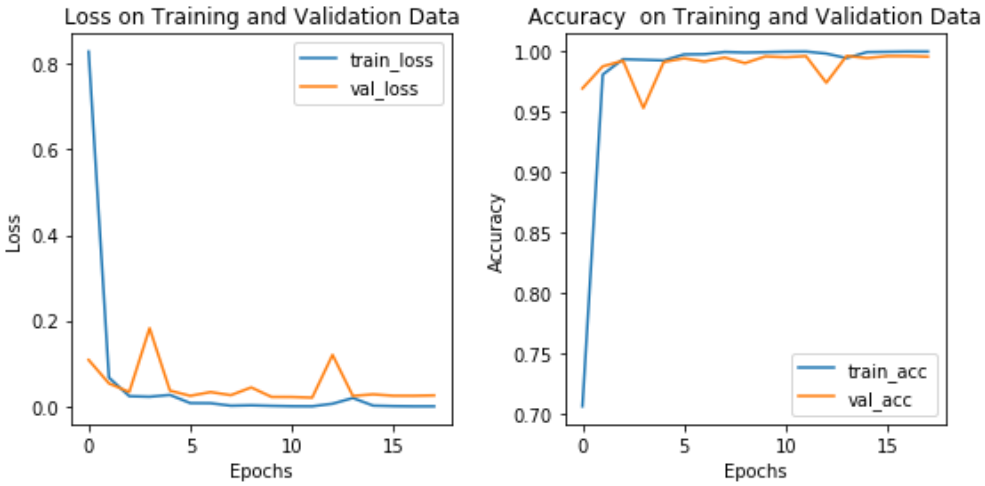

图 15：第 4 折训练集和验证集 loss、accuracy 随 epoch 走势

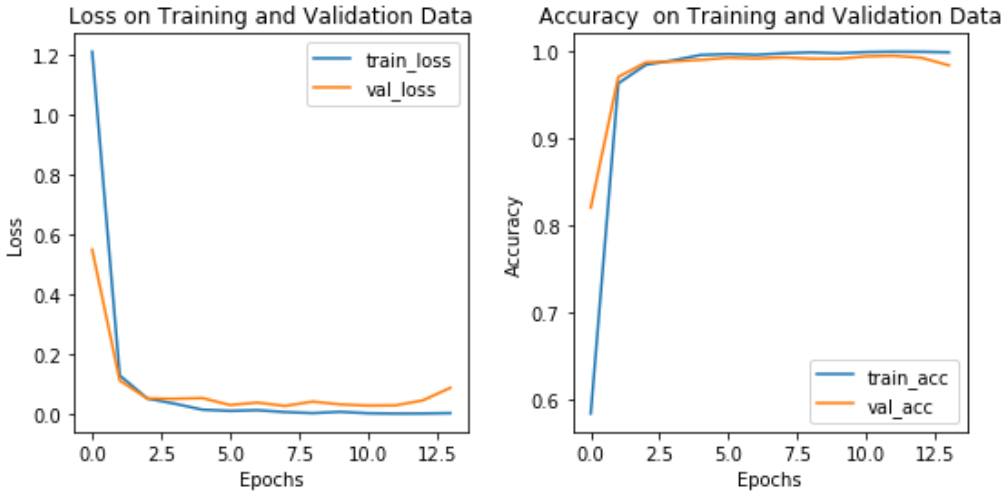

图 16：第 5 折训练集和验证集 loss、accuracy 随 epoch 走势

上面 5 组图为每折训练集和验证集 loss、accse 随 epoch 走势。从图中可以看出模型收敛速度很快，2 到 3 轮就基本上收敛了，后续收敛速度很慢，偶尔会有波动，到达第 10 轮时，基本上已经停止收敛，继续训练验证集 loss 反而会增加，说明开始出现过拟合了，所以 Early stopping 还是很有必要的。

3.3.2. 模型在测试数据集上的表现

由于采 5 折交叉验证进行训练，这里保存了 5 次训练的权重，然后将这 5 组权重分别导入到模型中对测试集进行预测。然后将 5 组预测结果进行取平均值，然后将此平均值保存为 CSV 文件提交文件进行提交，查看 Kaggle 得分只有 0.28831，没有达到 0.26 的目标要求。

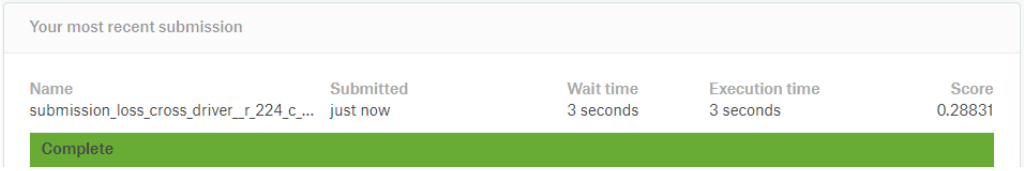

图 17:随机划分数据时 kaggle 得分

4. 改进

4.1. 改进方向

一、模型方面改进，采用多模型融合：每个模型往往都有自己的局限性，融合多个模型，就是充分运用不同分类算法各种的优势，取长补短，组合形成一个更强大的分类系统。

二、数据预处理方面改进。

从上面的结果可以看出，模型在训练集上有很好的表现，但是到测试集上，表现却大大降低了。分析训练数据集和测试数据集，测试数据集除了有更多的数据外，司机与训练数据集也完全不同，原因可能出现在这里。在不区分司机的情况下，训练集和验证集里会有相同的司机，而这可能导致过拟合，使得模型在训练数据上表现很好，但降低了泛化能力，使得其在测试数据上表现不佳。还就是训练数据如果能够扩增的话，这样也可以增加训练数据库，增强模型的泛化能力。所以数据预处理上主要着手以下两个方向的改进：

1、 所以按照不同的司机对训练数据进行划分，使得训练数据里的司机不出现在验证数据里。然后再训练模型，对测试集进行预测。

2、 数据扩增，对训练图片进行随机旋转，水平平移，垂直平移，错切变换，图像缩放等，侧面丰富训练数据，达到增大训练集的作用。

4.2. 改进结果

这里尝试了四种单独模型，InceptionV3、ResNet50、Xception、InceptionResNetV2。尝试不同的层数，分别对他们进行 fing_tune，然后将表现最好的 weight 保存下来，用于后续模型融合。

单模型 名称	模型 总层数	Fine_tune 起始层	验证集 val_loss	验证集 val_acc	Kaggle public score
InceptionV3	314	130	0.56	0.85	0.90
InceptionV3	314	168	0.30	0.91	0.42
InceptionV3	314	17	0.47	0.85	NA
Xception	135	76	0.33	0.90	0.64
Xception	135	81	0.29	0.89	0.63
Xception	135	106	0.64	0.82	NA
ResNet50	176	78	0.53	0.86	0.61
ResNet50	176	90	0.56	0.86	0.98
ResNet50	176	120	0.35	0.88	0.53
ResNet50	176	110	0.43	0.88	1.0
InceptionResNetV2	783	301	0.24	0.89	0.29

表 2：不同单模型的表现

现在用单模型提取特征向量，然后进行模型融合，不同的模型融合之后效果并不一样，单模型表现好的融合起来未必对总体有提升作用。现采用不同组合，其表现效果如下表 3 所示。参入标示为√，不参入标示为×。

模型 编号	InceptionV3	Xception	ResNet50	InceptionResNetV2	Kaggle public score
1	√	√	√	√	0.27
2	√	√	√	×	0.26
3	√	×	√	√	0.27
4	×	√	√	√	0.27
5	×	×	√	√	0.27
6	√	×	√	×	0.27

表 3：模型融合后的表现

从结果来看，模型融合之后效果非常显著，kaggle 得分有了很大的降低。融合前三个模型表现最好，达到了项目要求的 0.26 的目标。同时也发现，未必模型融合的越多越好，融合四个模型结果和融合两个模型差不多，都为 0.27。单模型表现不错，比如 InceptionResNetV2，融合起来未必帮助最为明显。

下图 18 为模型融合后（编号 2），训练集和验证集 loss、accuracy 随 epoch 走势，从图中可以看出随着 epoch 增加，loss 和 accuracy 在前 10 轮快速收敛，30 轮之后训练集和验证集都非常好的收敛了。

图 18: 模型融合之后训练集和验证集 loss、accuracy 随 epoch 走势

5. 结论

5.1. 结果

当采用 VGG16 模型，对训练数据集进行随机划分的时候，kaggle 得分为 0.28，没有达到项目要求前 10%，0.26 的目标。

改进之后，采用多模型融合，按照不同司机对训练数据进行划分、采用数据增强的时候，

kaggle 得分 0.26, 达到了项目要求前 10%的目标。

图 19：模型融合后提交 kaggle 得分（编号 2）

5.2. 总结

通过这次毕业项目的实践，我学到了很多东西，也增强了处理实际问题的信心。首先我对处理整个项目的整体步骤有了更加清晰的认识，如何数据分析、预处理，如何构建模型，选择什么样的评估方式等等。还就是对几大经典模型有了更加清楚的了解，知道如何运用这些训练好的基础模型，如何进行迁移学习和 fine_tune。如何吸取各大模型的长处，进行模型融合。还就是对调参也认识更清楚了，需要很细致，很要耐心。

自己动手通过实际项目的演练，学到的东西方能更加扎实，才能更好的将课程学到的东西转化为自己的东西。所以今后学习过程中，要多动手实践，最好多接触一些实际项目。

5.3. 后续改进

清理训练数据中标签异常值，由于种种原因训练数据中往往会有一些异常值，这些异常值会对我们的模型有不利影响。可以想办法对训练数据进行梳理，这样可以一定程度上提升模型的性能。

数据预处理，探索更好的数据预处理方式，对数据进行更好的预处理，减少不必要的干扰。

探索更多的模型融合，每个模型往往都有自己的局限性，融合多个模型，就是充分运用不同分类算法各种的优势，取长补短，组合形成一个更强大的分类系统。

参考文献

Karen Simonyan and Andrew Zisserman. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. Computer Vision and Pattern Recognition (cs.CV)

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. In Proc. BMVC., 2014

Cimpoi, M. Maji, S. and Vedaldi, A. Deep convolutional filter banks for texture recognition and segmentation. CoRR, abs/1411.6836, 2014.

Kaiming He. Xiangyu Zhang .Shaoqing Ren. Jian Sun. Deep Residual Learning for Image Recognition.2015.

杨培文.手把手教你如何在 Kaggle 猫狗大战冲到 Top2%.
https://zhuanlan.zhihu.com/p/25978105

刘建平. 卷积神经网络(CNN)模型结构. http://www.cnblogs.com/pinard/p/6483207.html

ZSYGOOOD. 从过拟合与欠拟合到偏差-方差分解.
https://blog.csdn.net/BitCs_zt/article/details/79076500

SunnyMarkLiu. 误差模型：过拟合，交叉验证,偏差-方差权衡.
https://blog.csdn.net/Mark_LQ/article/details/51482860

Evillist. 交叉验证，K 折交叉验证的偏差和方差分析.
https://blog.csdn.net/evillist/article/details/76009632