侦测走神司机

机器学习工程师纳米学位毕业项目

肖映彩

2018 年 6 月 1 日

目录

1. 定义

1.1. 项目背景

在驾驶过程中，如果司机的注意力不集中而分心去做其他的事情（比如打电话，看手机，与其他人交谈，伸手去拿后面的东西等），对于驾驶过程中出现的状况就不能做出及时反应，很可能就会引发交通事故。根据美国国家统计与分析中心 2015 年的报告[1]，美国有 15%的严重交通事故是由于司机分心去做其他事情而造成的，这些事故共造成 3477 人死亡、约 391000 人受伤。所以通过一个辅助驾驶装置去检测司机的驾驶状态，当发现司机有分心做其他事情的行为时能及时提醒司机，这是非常有意义而且有必要的。

本项目是由 State Farm 在 Kaggle 上发起的一个竞赛项目，目的是能通过算法去识别司机的行为，判断司机是否在安全驾驶状态。这样，当司机做出非安全驾驶的行为时，就可以去提醒司机专心驾驶，从而在一定程度上减少交通事故。

1.2. 问题描述

项目的要求是给定司机驾驶状态的 2 维图像，去识别图像中司机是处于哪种驾驶状态，这实际上是一个分类问题：用给定的带标签的训练数据（图像）训练模型，然后用测试数据集验证模型。需要识别的驾驶状态一共有如下 10 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

1.3. 评估指标

根据项目要求，模型的评估指标为对数损失函数（multi-class logarithmic loss）：

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

其中 N 是测试集中图片的数量，M 是图片类别的数量，log 是自然对数，如果输入的图片 i 属于类别 j 则 y_{ij} 为 1 否则为 0，p_{ij} 则是输入图片 i 属于类别 j 的概率。在测试集上模型的对数损失函数的数值越低，说明模型的表现越好。

2. 分析

2.1. 数据集

项目提供分辨率为 640x480、格式为 JPG 的图片，分为训练数据和测试数据，其中训练数据中的图片按照标签 c0~c9 分别存放在对应的文件夹中。图片是从视频中导出的，由摄像机从司机的右侧进行拍摄，这样可以清楚地拍摄到司机动作。这些图片大部分拍的比较清晰，

有一小部分稍微有点模糊，光线大部分比较亮，有些则暗一点。每个司机在 c0~c9 中都由一定数量的图片，这些司机有男的也有女的，有年轻的也有年老的，身材有胖的也有瘦的。

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

图 2.1 训练集图片示例

测试数据一共包含 79726 张图片，训练数据一共包含 22424 张图片，其中各类别包含的图片数量如下图所示：

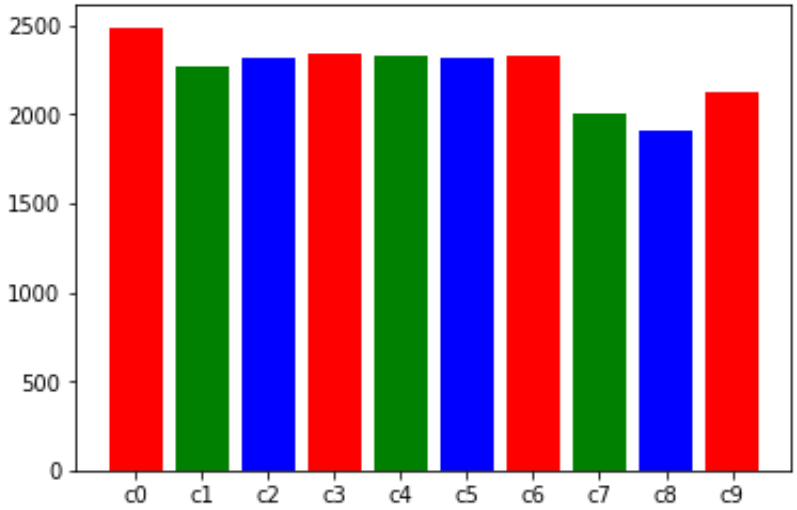
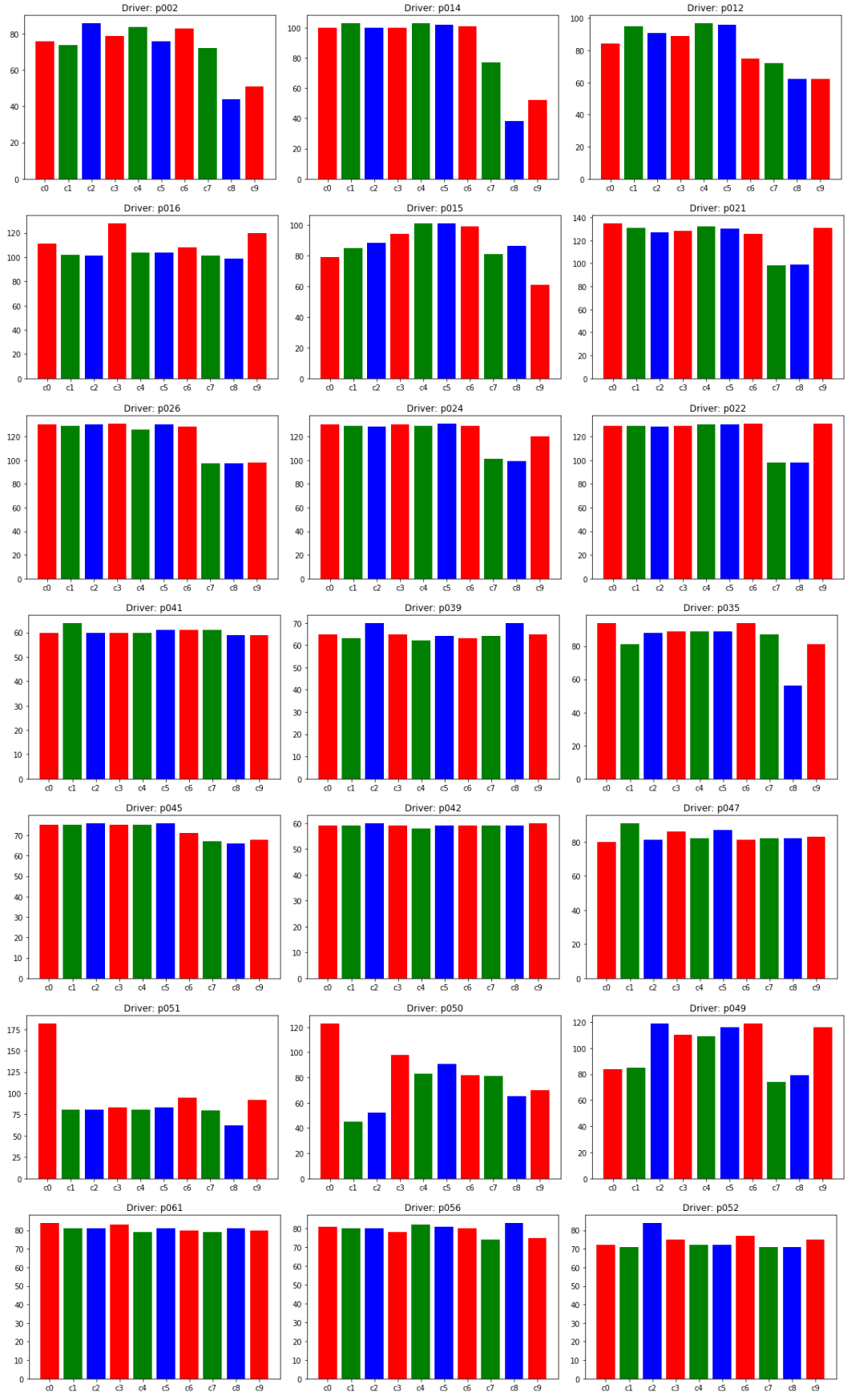


图 2.2 训练数据集各类别图片数量

训练数据提供 26 个司机的图片，每个司机的图片都有 c0~c9 共 10 种类别，相应的统计分布图如下：


Driver: p002
Driver: p014
Driver: p012
Driver: p016
Driver: p015
Driver: p021
Driver: p026
Driver: p024
Driver: p022
Driver: p041
Driver: p039
Driver: p035
Driver: p045
Driver: p042
Driver: p047
Driver: p051
Driver: p050
Driver: p049
Driver: p061
Driver: p056
Driver: p052
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9

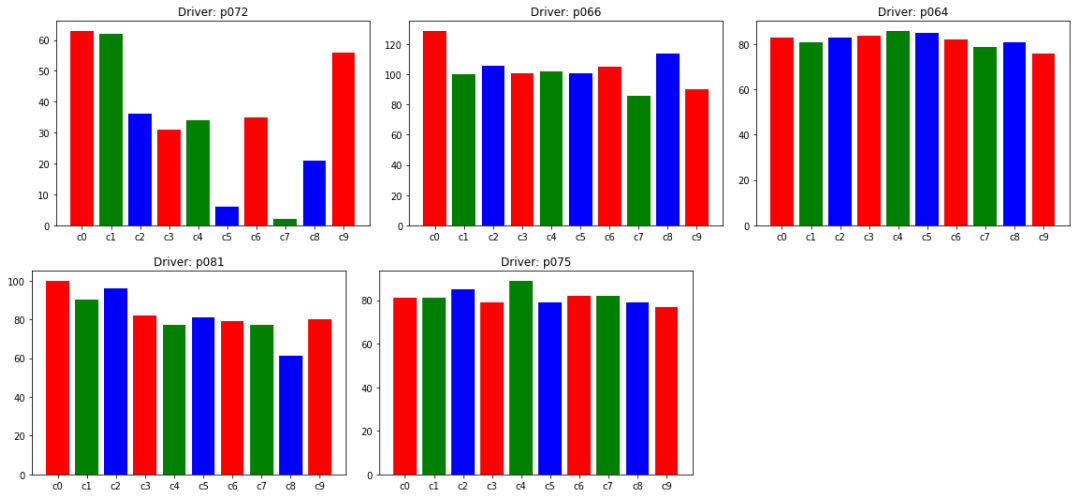


图 2.3 每个司机对应的驾驶行为类别分布

在项目提供的 driver_imgs_list.csv 文件中，把每个司机都分配了一个 ID，统计了每个司机所有图片的文件名，以及这些图片的类别。这样，在做数据预处理时我们就可以根据 driver_imgs_list.csv 文件提供的信息，对训练数据集进行划分（分为训练集和验证集）。划分训练数据集时应该按照司机来划分，同一个司机的图片不能同时出现在训练集和验证集中，这是因为同一个类别中包含了同一个司机的若干张照片，这些照片是不能同时出现在训练集和验证集中的。

2.2. 方法与技术

2.2.1. 卷积神经网络

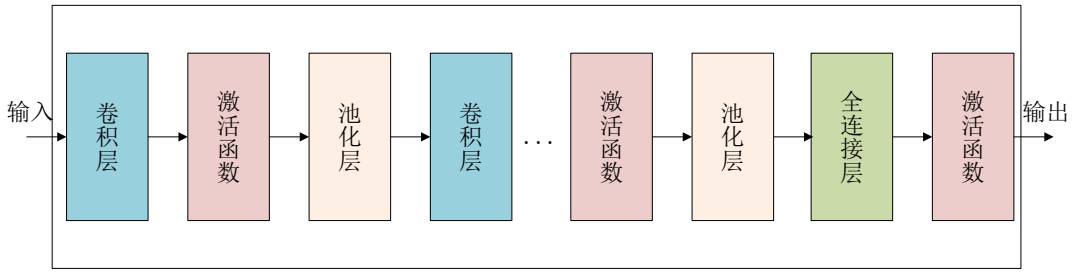


图 2.4 典型的卷积神经网络结构

一个典型的卷积神经网络模型的结构如图 2.4 所示，整个模型的前面由若干卷积层和池化层“堆积”而成，而在这些层的后面是全连

接层。卷积层用来从输入数据中提取特征，池化层抽取卷积后的结果实现对特征的降维操作，后面的全连接层实现目标任务的输出。

与传统机器学习方法不同的是，卷积神经网络不需要人工提取特征，提取特征的过程可由卷积层和池化层来实现。卷积运算可以看作是一个函数从另一个函数上“滑过”时产生的响应。在数字图像处理中，我们用到的是二维离散卷积。假设有图 2.5 中的卷积核和输入图像，从图像中的第一个像素开始，用卷积核中的参数与图像中对应位置的像素值相乘然后求累加和是一次卷积操作。类似的操作，用卷积核以一定的步长在输入图像上从左到右从上而下地做卷积，这样就得到输入图像对于该卷积核的响应。如果某卷积核的功能是提取图像中的线段，那么经过卷积操作后，输入图像中的线段就会被提取出来。卷积层就是通过多个不同功能的卷积核，把图像中的特征提取出来，前面的卷积层提取的特征比较简单，越往后的卷积层提取的特征就复杂。在卷积层之后，再通过池化层对卷积层的输出进行抽取，池化操作一般用最大池化法，选取一个局部最大值作为输出。池化操作虽然会损失一部分信息，但是可以大大减少网络的参数，防止模型过拟合。通过卷积层和池化层逐层地不断对图像的特征进行抽象，再结合网络的后续操作，就可以让模型可以学习高层次的概念。

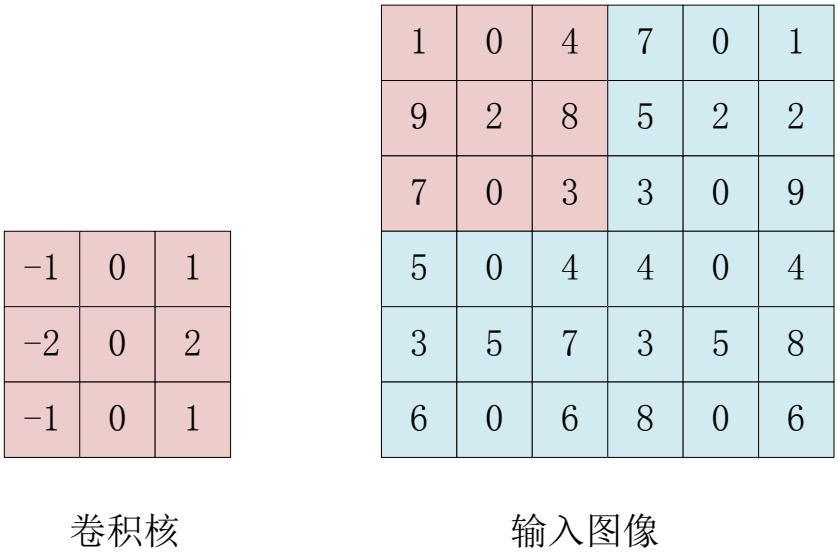


图 2.5 二维卷积核和图像

卷积神经网络的几个重要特性如下：

■ 局部连接

每个神经元与输入神经元的一个局部区域连接，这个局部区域叫做感受野(receptive field)。在图像中，一个局部区域中的像素是有很强的关联性的，这种局部连接特性在经过卷积操作后产生最强的响应，从而提取出有用的特征，如边缘、角点、纹理等。

■ 参数共享

用卷积核与输入图像做卷积操作的时候，并不是输入图像中的每个像素都有一个权重参数，而是整张图像共享这个卷积核中的参数。这样，对于一个提取特定特征的卷积核来说，不管这个特征在图像中的什么位置，经过卷积操作后这个特征都能被提取出来。参数共享的特性使得网络中的参数大大减少，有利于训练较大的卷积神经网络。

2.2.2. 迁移学习

近年来，卷积神经网络（CNN）在图像领域取得了巨大的成功，在大规模视觉识别竞赛（LSVRC）上，出现了一些非常经典的 CNN 模型，比如 AlexNet[2]、VGGNet[3]、GoogLeNet[4]、ResNet[5]等。这些模型在 LSVRC 上 取得的成功，说明 CNN 非常适合用来做图像的分类和识别。但是如果自己从头搭建 CNN 模型训练的话，需要大量的训练数据才能训练好模型，本项目提供的 10 个类别的训练图片一共只有 22424 张，这是远远不够的。AlexNet、VGGNet、GoogLeNet、ResNet 这些经典的模型都是在 LSVRC 的数据集上得到充分训练的，我们可以利用这些训练好的模型，通过迁移学习的方式来解决本项目的问题。VGG16 模型结构简单，容易理解，可以先基于该模型来做迁移学习。

2.2.3. 优化器

训练深度学习模型的过程，就是求取目标函数极值的过程，这涉及许多优化问题，需要用到一些优化算法来求解。下面介绍几种常用的优化算法：

- 随机梯度下降

  考虑目标函数

$$f(x)=\frac{1}{n}\sum_{i=1}^{n}f_i(x)$$

其中 $f_i(x)$ 是索引为 i 的训练样本的损失函数， n 是训练样本数。计算

梯度

$$\nabla f(x)=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x)$$

可以看到，当训练数据集很大时，梯度下降算法的计算开销很大。在实际应用中一般使用随机梯度下降，对于给定的学习率η，每次迭代时随机均匀地对数据集采样一个小批量样本集并计算$\nabla f_i(x)$来迭代x：

$$x \leftarrow x-\eta\nabla f_i(x)$$

- Adadelta

Adadelta 算法是一种自适应学习率的优化算法。该算法使用小批量随机梯度按元素平方的指数加权移动平均变量s，每个元素初始化为 0。给定超参数$\rho(0\le\rho\le1)$，在每次迭代中，首先计算小批量随机梯度g，然后对该梯度按元素平方项$g\odot g$做指数加权移动平均，

$$s \leftarrow \rho s+(1-\rho)g\odot g$$

然后计算当前需要迭代的目标函数自变量的变化量

$$g' \leftarrow \frac{\sqrt{\Delta x+\varepsilon}}{\sqrt{s+\varepsilon}}\odot g$$

上式中，ε为常数(10E-5)，Δx初始化为零张量，目标函数自变量中每个元素都分别拥有自己的学习率。记录g'按元素平方的指数加权移动平均

$$\Delta x \leftarrow \rho\Delta x+(1-\rho)g'\odot g'$$

最后，自变量的迭代步骤如下

$$x \leftarrow x-g'$$

- Adam

Adam 算法也是一种自适应学习率的优化算法，它通过计算梯度的一阶矩估计和二阶矩估计为不同的参数动态地调整学习率。Adam 算法有如下超参数：

alpha: 学习率或称步长因子，控制权重的更新比率，建议 0.001。

beta1: 一阶矩估计的指数衰减率，建议 0.9。

beta2: 二阶矩估计的指数衰减率，建议 0.999.

epsilon: 非常小的数，防止在实现中出现除以零，建议 10E-8。

Adam 算法使用了动量变量v和小批量随机梯度按元素平方的指数加权移动平均变量s，每个元素初始化为 0。在每次迭代中，时刻t的小批量随机梯度记作g_t。将小批量随机梯度的指数加权移动平均记作动量变量v，并将它在时刻t的值记作v_t

$$v_t \leftarrow \beta_1 v_{t-1} + (1-\beta_1) g_t$$

将小批量随机梯度按元素平方后做指数加权移动平均得到s，并将它在时刻t的值记作s_t

$$s_t \leftarrow \beta_2 s_{t-1} + (1-\beta_2) g_t \odot g_t$$

对变量v和s均作偏差修正

$$\hat{v}_t \leftarrow \frac{v_t}{1-\beta_1^t}$$

$$\hat{s}_t \leftarrow \frac{s_t}{1-\beta_2^t}$$

使用以上偏差修正后的变量$\hat{v}_t$和$\hat{s}_t$，将模型参数中每个元素的学习率通过按元素运算重新调整

$$g_t' \leftarrow \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t + \varepsilon}}$$

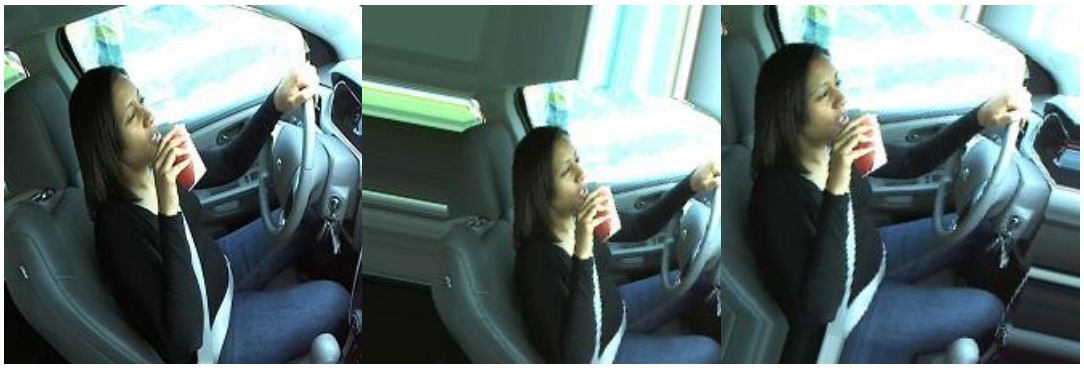
和 Adadelta 算法一样，目标函数自变量中每个元素都分别拥有自己的学习率。最后，时刻t的自变量x_t的迭代步骤与小批量随机梯度下降类似：

$$x_t \leftarrow x_{t-1} - g_t'$$

Adam 算法很容易实现，具有很高的计算效率和较低的内存需求。

2.2.4. 数据增强

为了训练泛化能力强的模型，最好的方法是使用大量的数据进行训练，这样模型就能从样本的差异中得到很好的泛化能力，避免过拟合。但是在实际中，训练样本总是有限的，本项目的训练数据集只有 2 万多张图片，用来训练模型是远远不够的。为了解决这个问题，一种有效的方法是对数据集进行数据增强操作。常用的图像数据增强操作有：旋转、剪切、缩放、水平移动、垂直移动、通道转换、水平翻转、垂直翻转等。在 Keras 中，可以通过生成器实时生成带数据增强的图像数据，下图是一个本文中所作数据增强操作的示例。



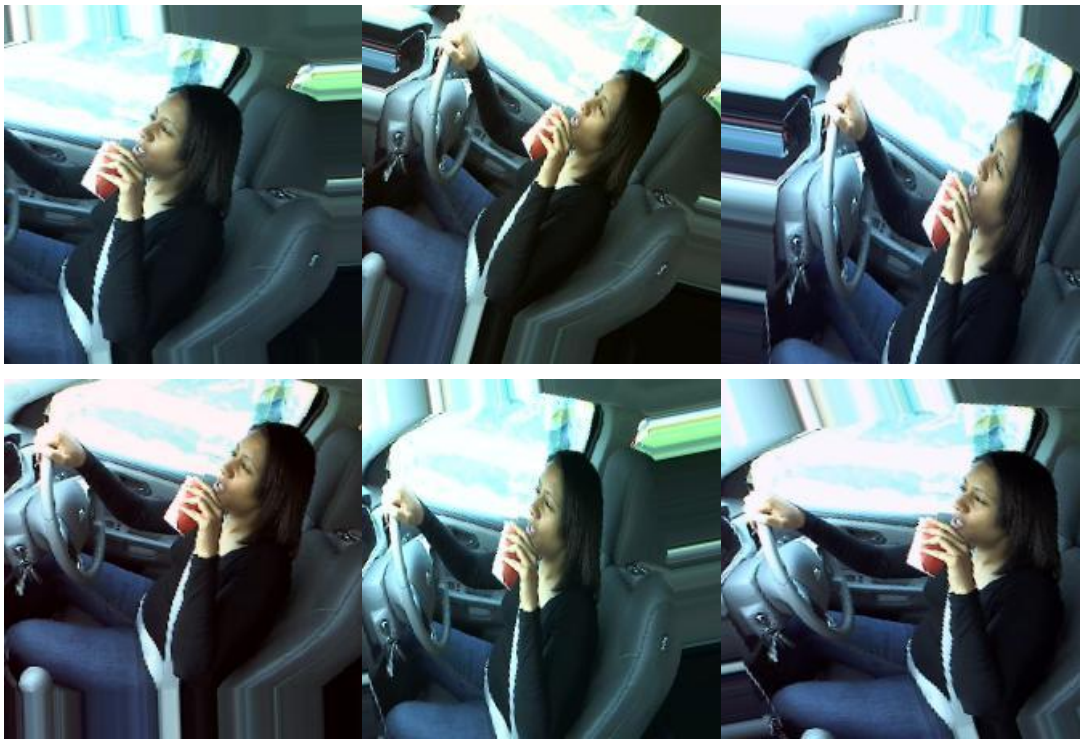


图 2.6 数据增强效果示例

上图中第一张图片是原图，其他 8 张图片是做了增强操作后的效果。数据增强的基本前提是不会改变样本所属的类别，而且要根据实际情况选择合理的操作，例如在本项目中，垂直翻转就是不合理的操作。本项目对驾驶行为的分类是从驾驶员的肢体动作判断的，所以数据增强操作不能影响驾驶员的肢体动作。从上图可以看出，所做的数据增强操作很好地保留了这些信息。

2.3. 基准指标

在 Kaggle 上，该项目的 Public Leaderboard 是用大约 31%的测试数据去测试模型，Private Leaderboard 则是用大约 69%的测试数据去测试模型，评估指标 logloss 的值越低，说明模型的表现越好，在 Leaderboard 上的排名也就越靠前。该项目的 Leaderboard 有 1440 组参赛者，做本项目的目标是进入 Private Leaderboard 的 top 10%，也

就是 Private Leaderboard 上的 logloss 要低于 0.25634。

3. 具体方法

3.1. 数据集划分

按照前面的分析，划分训练集和验证集时是根据 driver_imgs_list.csv 文件中司机的 ID 去划分的，也就是同一司机的图片不能同时出现在训练集和验证集中。本文中对数据集的划分方法为：取 driver_imgs_list.csv 文件中前 3 个司机的图片作为验证集，放在 validation 文件夹中，剩下 23 个司机的图片作为训练集放在 test 文件夹中，这样训练集有 20000 张图片，验证集有 2424 张图片。

3.2. CNN 模型

3.2.1. 简单的 CNN 模型

在使用复杂的模型之前，先从一个简单的 CNN 模型开始。构建的简单 CNN 模型的结构如下图所示：

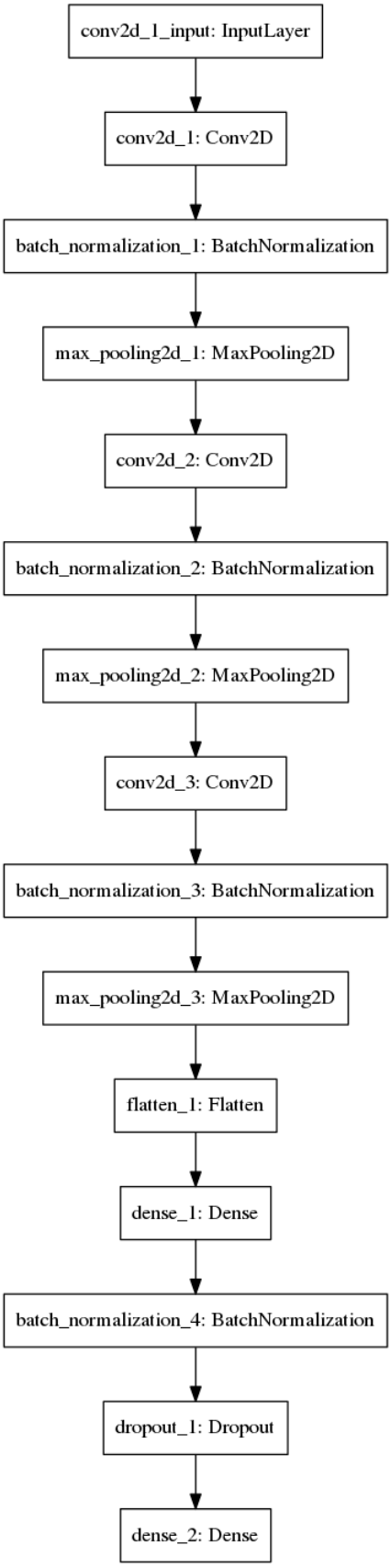


图 3.1 简单的 CNN 模型

该模型包括 3 个卷积层、3 个最大池化层、1 个全连接层和 1 个

Softmax 输出层，在每个卷积层和全连接层后面，还有 1 个 BatchNormalization 层，BatchNormalization 层可以加快模型的收敛速度，而且起到正则化的作用，可以降低模型过拟合的风险[6]。

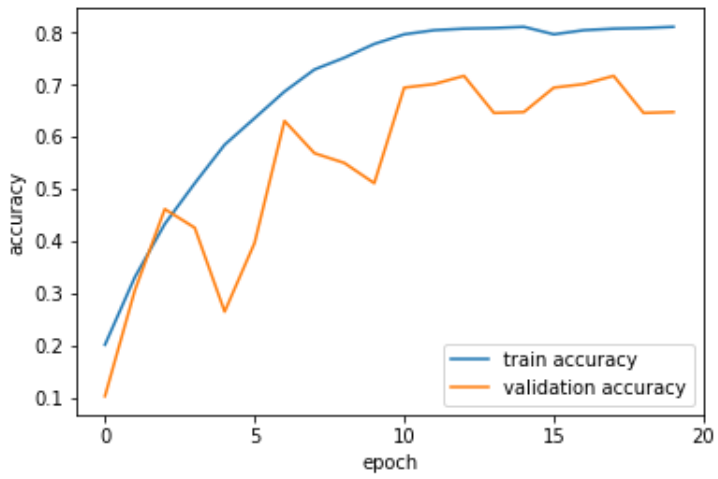
使用优化器 Adadelta 对模型进行训练，用图片生成器 ImageDataGenerator 把图片的灰度值缩放到[0，1]范围内，不做其他处理。得到的结果如下：

```
Epoch 1/10
156/156 [==============================] - 102s 655ms/step - loss: 0.3214 - acc: 0.9124 - val_loss: 2.5789 - val_acc: 0.2
556
Epoch 2/10
156/156 [==============================] - 107s 686ms/step - loss: 0.0180 - acc: 0.9965 - val_loss: 3.7387 - val_acc: 0.1
890
Epoch 3/10
156/156 [==============================] - 107s 685ms/step - loss: 0.0050 - acc: 0.9994 - val_loss: 4.3296 - val_acc: 0.1
903
Epoch 4/10
156/156 [==============================] - 107s 686ms/step - loss: 0.0017 - acc: 0.9999 - val_loss: 3.9433 - val_acc: 0.1
947
```

可以看到，训练 2 代以后模型就出现了严重的过拟合。

为了避免过拟合，可以对训练集进行一定程度的数据增强操作。设置训练集生成器的参数为：rescale=1./255，zoom_range=0.2，rotation_range=15，height_shift_range=0.2，width_shift_range=0.2，channel_shift_range=10, horizontal_flip=True。

对训练集做数据增强后，重新构建模型进行训练，训练 20 代后得到的结果如下：



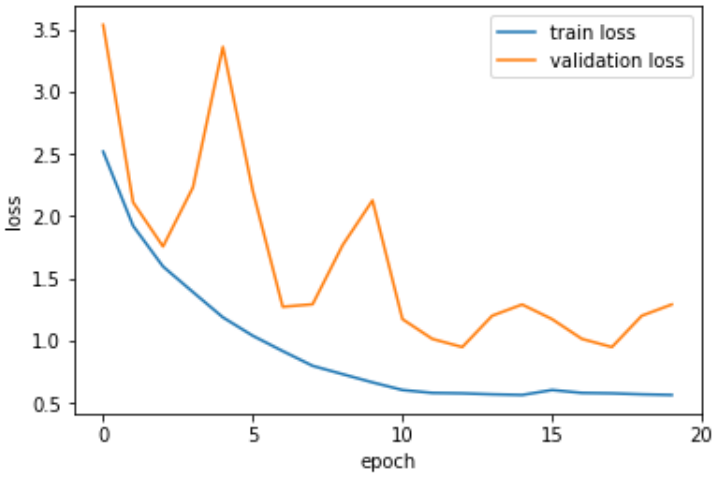


图 3.2 训练简单 CNN 模型的 accuracy 和 loss 曲线

可以看到，数据增强后的效果比较明显，val_acc 达到了 0.65 左右，这对于一个简单的 CNN 模型已经是非常不错了。接下来，可以尝试一下复杂的模型了。

3.2.2. VGG 模型

在上一节中，构建了一个简单的 CNN 模型进行训练，得到 val_acc 的 val_acc 在 0.65 左右，val_loss 在 1.1 左右，这结果对于本项目所要达到的要求是远远不够的。如果要提升图像分类精度，我们需要更加复杂的网络，最直接的方法是加深网络的层数。但是随着网络层数的增加，网络参数也随之增多，模型就越容易过拟合，要想训练出一个好的模型就需要更多的数据。本项目提供的训练数据比较少，要自己构建并训练一个全新的 CNN 模型基本上是不可能的，但是可以基于已有的预训练模型来训练我们所需要的模型。

VGG16 模型的结构简单，前面由一系列卷积层和池化层堆叠而成，后面接两个 4096 的全连接层，最后通过 Softmax-1000 层输出结果。在本项目的实现中，我们使用 Keras 自带的 VGG16 模型，只加

载前面的 5 个 block，采用"imagenet"预训练权重，后面接两层 128 全连接层，每个全连接层后接一个 BatchNormalization 和 Dropout(0.5)层，分类输出层改为本项目对应的 Softmax-10，整个模型的结构如下：

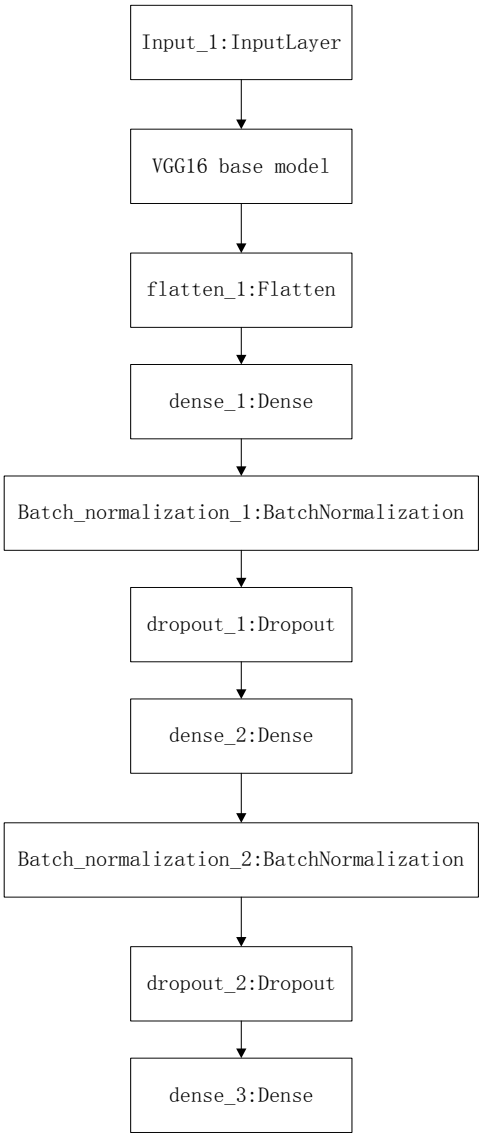


图 3.3 VGG16_FC 模型结构框图

该模型在下文中用 VGG16_FC 指代。

由于本项目非常容易过拟合，所以对数据增强是很有必要的，训练 VGG16_FC 模型所做的数据增强操作：shear_range=0.2，zoom_range=0.2，rotation_range=15，height_shift_range=0.2，width_shift_range = 0.2，channel_shift_range=10，horizontal_flip=True。

训练的时候，首先锁住预训练模型的 5 个 block 的层，用优化器 Adam(lr=1e-3)训练后面的全连接层和 Softmax-10 层，这样不致于在训练新加入层的时候，因为大的梯度变化破坏预训练模型的权重。训练几代后，再放开 block5 的层，冻结前面 4 个 block 的层，继续训练。block5 训练几代后，再放开 block4 的层训练。按照同样的方法，逐个放开 block 的层进行训练，训练每个 block 的时候都设置 EarlyStopping，当 val_loss 超过 patience=3 代不下降就停止训练。这样得到的整个训练过程的训练集和验证集的 accuracy 和 loss 的曲线如下所示：

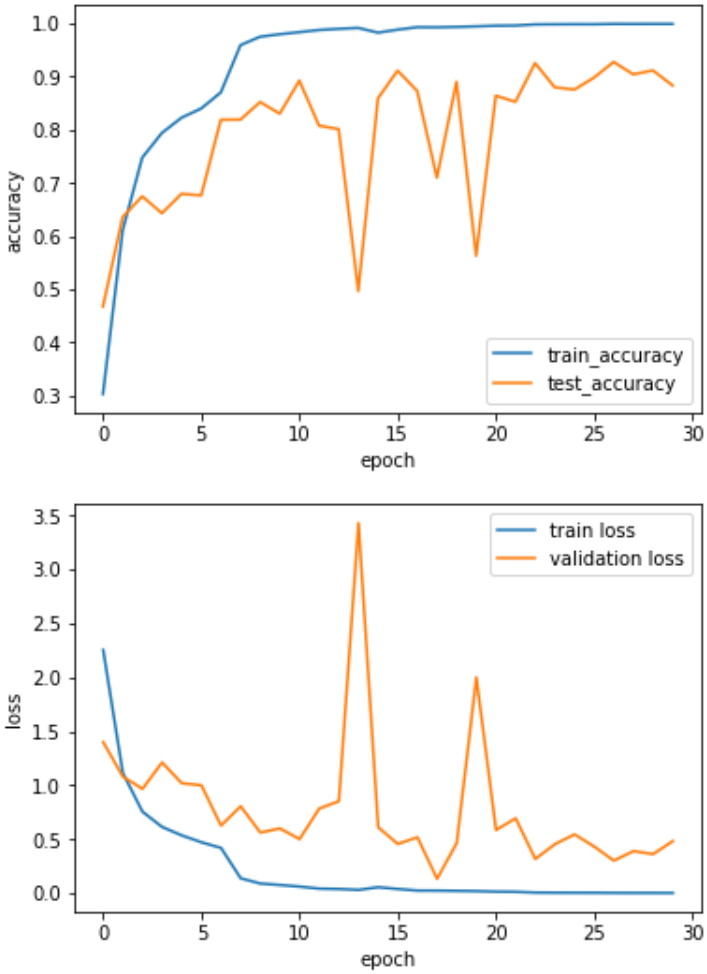


图 3.4 训练 VGG16_FC 的 accuracy 和 loss 曲线

VGG 模型除了 VGG16 外，还有 VGG19。基于预训练的 VGG19

模型构建一个新的模型，后面的全连接层和 Softmax 层跟 VGG16_FC 相同，在下文中该模型用 VGG19_FC 指代。

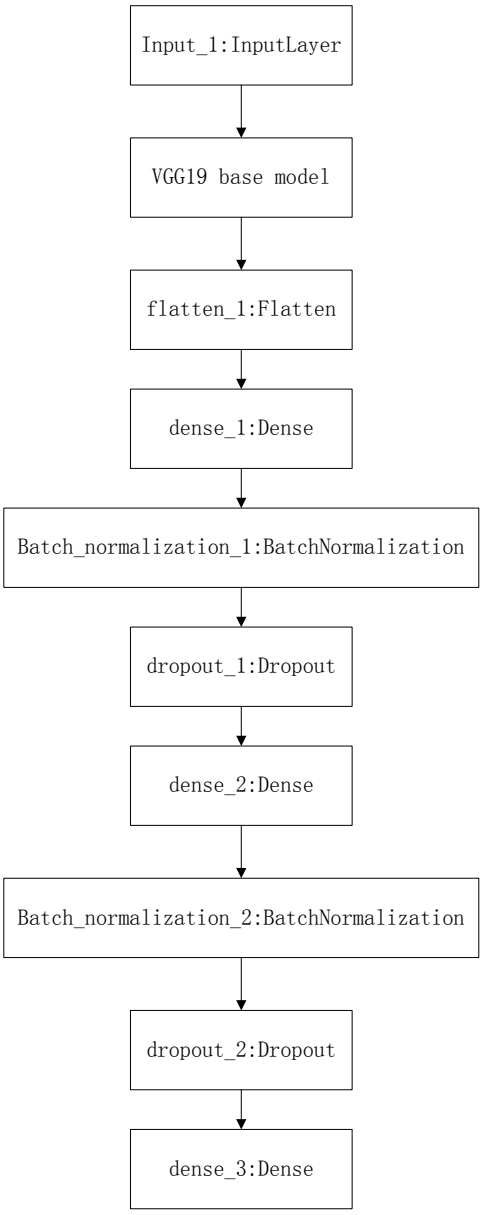


图 3.5 VGG19_FC 模型结构框图

训练该模型的方法与 VGG16_FC 一样，得到的 accuracy 和 loss 曲线如下：

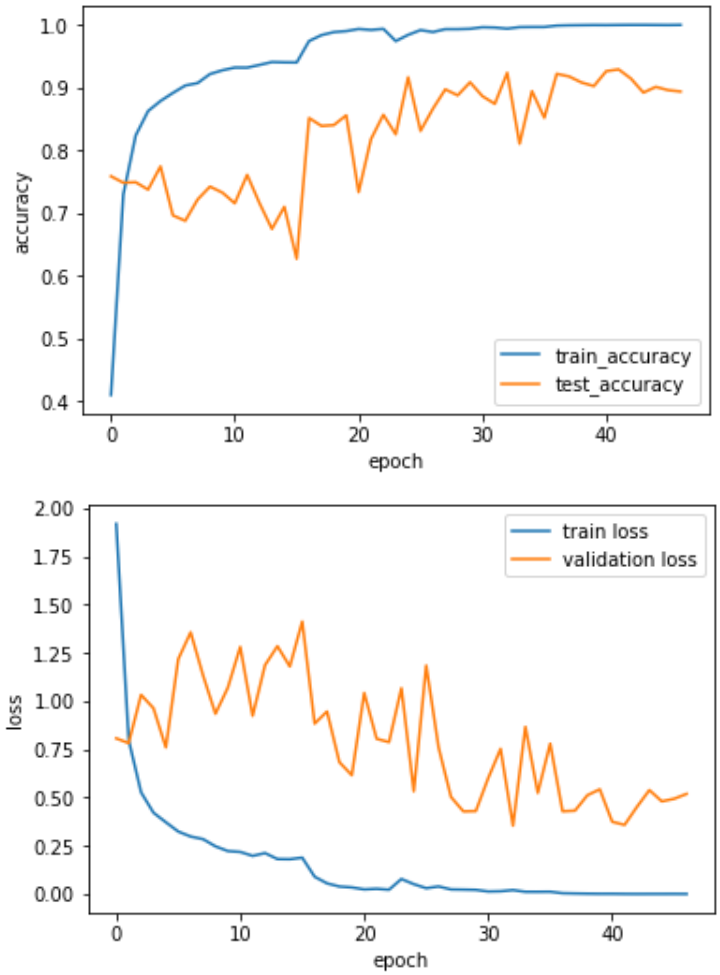


图 3.6 训练 VGG19_FC 的 accuracy 和 loss 曲线

VGG 模型是 2014 年推出的 CNN 模型，在 Softmax 输出层前面接的是两层全连接层。由于全连接层参数较多，用全连接层容易造成过拟合，所以在后面几年推出的 CNN 模型摈弃了使用全连接层，转而使用全局平均池化层(Global Average Pooling Layer)。全局平均池化层由 Lin 等人[7]提出，主要思想是分别对每个特征图内部求平均，再把得到的特征向量送入 Softmax 层进行分类。全局平均池化层可以大大减少参数量，可以有效地防止过拟合。对上文中的 VGG16_FC 和 VGG19_FC 模型进行改进，用全局平均池化层取代两层全连接层，得到的模型分别用 VGG16_GAP 和 VGG19_GAP 指代，两个模型的框图如下：

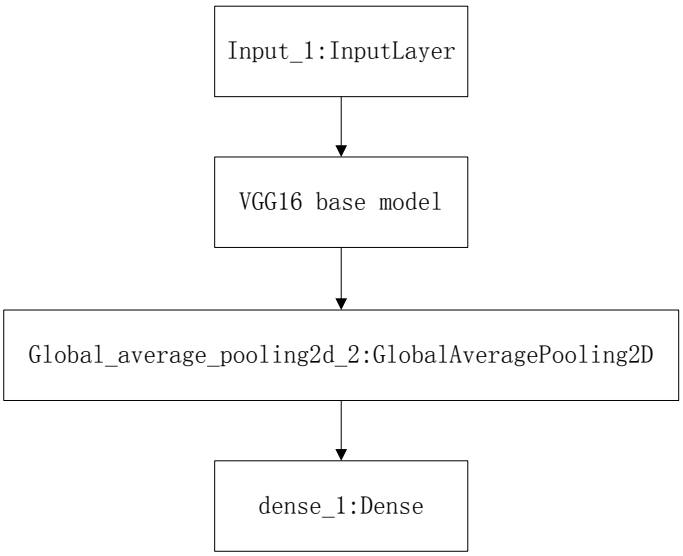


图 3.7 VGG16_GAP 模型结构框图

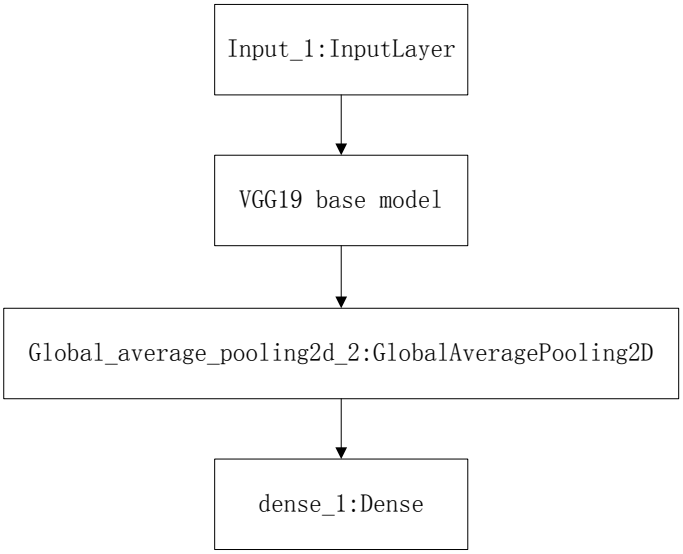


图 3.8 VGG19_GAP 模型结构框图

两个模型的训练方式与上文的 VGG16_FC 相同，得到的 accuracy 和 loss 曲线如下：

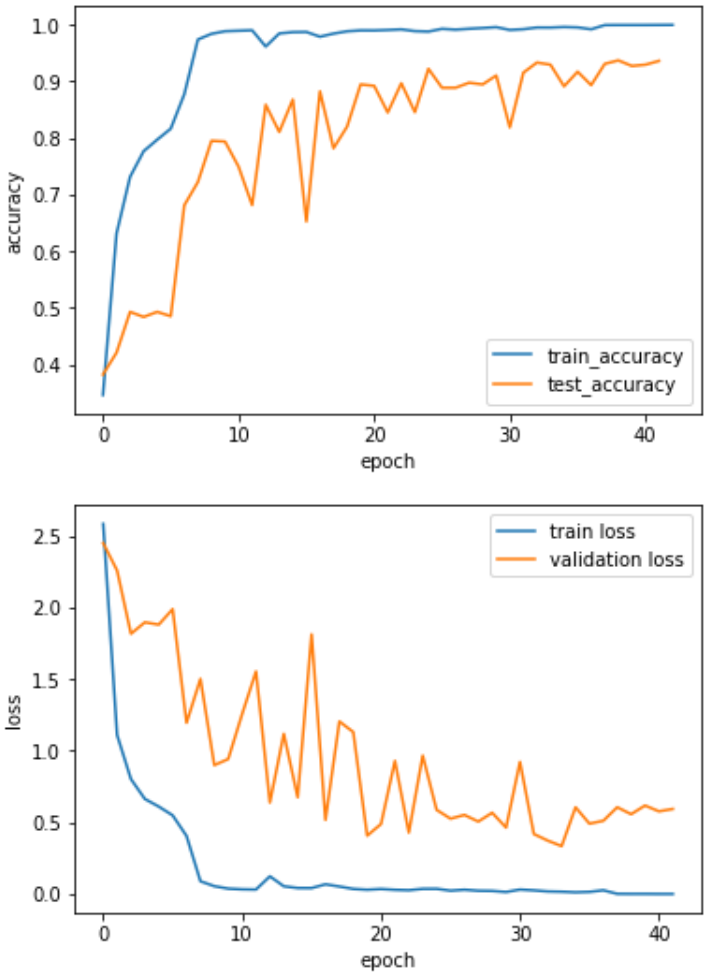
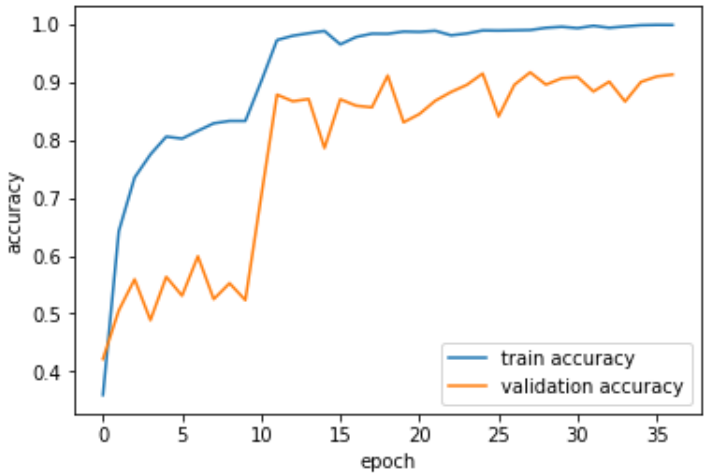


图 3.9 训练 VGG16_GAP 的 accuracy 和 loss 曲线



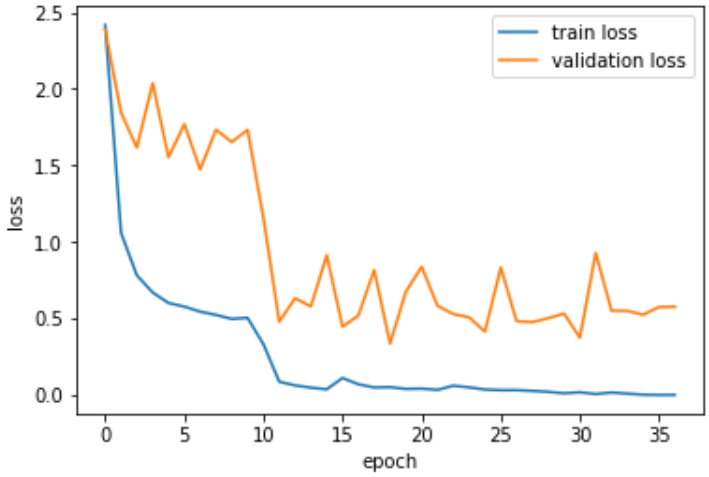


图 3.10 训练 VGG19_GAP 的 accuracy 和 loss 曲线

3.2.3. 模型融合

在上一节中，一共训练了 VGG16_FC、VGG19_FC、VGG16_GAP 和 VGG19_GAP 这四个模型。为了得到更好的结果，可以把这四个模型做融合。一种简单的融合方法是把四个模型的输出做平均，作为最终模型的输出，本文采用就是采用这种融合方法。融合后模型的结构如下图所示：

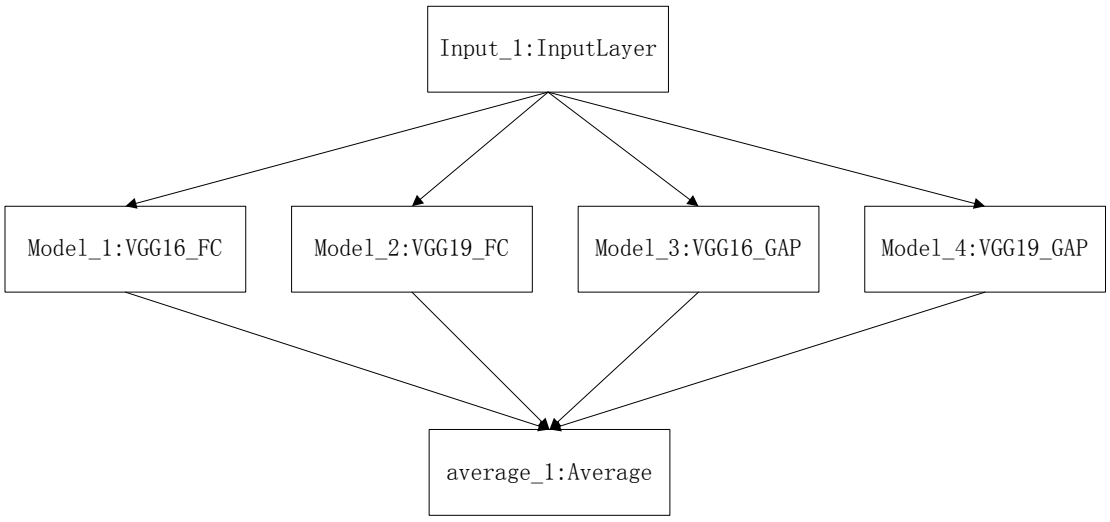


图 3.11 融合模型结构框图

4. 结果

将训练的模型提交到 Kaggle，得到的结果如下：

模型	Private Score	Public Score
简单 CNN 模型	1.67948	1.45456
VGG16_FC + VGG19_FC	0.31061	0.28310
VGG16_GAP + VGG19_GAP	0.26045	0.27306
4 个 VGG 模型	0.24568	0.24612

集成 4 个 VGG 模型后，在 Kaggle 上的 Private Score 为 0.24568，排名 130/1440，已经达到本项目的预期目标。

5. 结论

本项目的训练集图片是从视频中提取的，不同类别的图像相似性非常高，而且训练集总共只有 2 万多张图片，所以训练过程中非常容易出现过拟合。针对过拟合的问题，本文中使用了数据增强、Dropout、Batch Normalization、EarlyStopping 等方法，训练了 VGG16_FC、VGG19_FC、VGG16_GAP 和 VGG19_GAP 这 4 个模型。集成这 4 个模型，把它们输出结果的平均值作为最终输出结果，提交到 Kaggle 的结果达到预期目标。

通过这个项目，学习了像 VGG 这样的经典 CNN 模型，并基于这些预训练的模型来构建自己的图像分类应用，在这个过程中学习了训

练 CNN 模型的基本方法，这还是有很大收获的。

本文中只使用了 VGG 模型，其实还有很多优秀的 CNN 模型可以使用，后续可以多做尝试。改进的方法可以从以下几个方面考虑：

1) 本文中只使用了 VGG 模型，整个网络只有十几层，为了提升模型性能，可以尝试 ResNet101，ResNet169，Inception-ResNet-V2 等大型网络。
2) 使用衰减的学习率，每隔几代将学习率减少。
3) 除了对训练集做数据增强外，还可以在预测的时候进行数据增强(Test Time Augmentation)，也就是先对图片做数据增强再进行预测，然后把得到预测结果进行平均作为最终的输出。
4) 可以尝试近两年提出的新模型，这些模型的性能更好，比如 DenseNet[8]和 SENet[9]。

6. 参考文献

[1]https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/812_381_distracteddriving2015.pdf

[2] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.

[3] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

[4] C. Szegedy *et al.*, "Going deeper with convolutions" *2015 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition" *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, United States, 2016, pp. 770-778.

[6] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. 2015:448-456.

[7] Lin M, Chen Q, Yan S. Network In Network[J]. Computer Science, 2013.

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, et al. "Densely Connected Convolutional Networks". 2017 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Jie Hu and Li Shen and Gang Sun. "Squeeze-and-Excitation Networks". 2017.