Distracted Driver Detection

Yuxiang Machine learning Nanodegree

2017/6/22

I. 问题的定义

一、项目概述

本项目将训练一个能够识别驾车司机是否分心驾驶的模型，这个模型可以应用到车载摄像头中，实时显示当前司机驾驶的状态。

我们都有过类似的经历：绿灯亮的时候，在你前面的车辆在前面不让步导致你不能快速通过； 或者一个不起眼的车辆突然在你前面减速并且左右摇摆。

当你从这些令你厌恶的车辆旁经过时，你会发现这些司机没用专心驾驶。当你看到车里的司机正在发短信，或者似乎沉浸在社交媒体中，又或者正在拿着手机通话时，你一定不会感觉到太惊讶。

根据 CDC 机动车安全部的数据显示，20%的车祸是由司机分心所致。这意味着每年因此导致 425000 人受伤，3000 人死于分心驾驶。

State Farm 保险公司改善这些令人震惊的统计数据，通过仪表盘上的摄像头自动检测司机是否从事了分心驾驶的行为，保证客户安全。 State Farm 保险公司提供了 2D 仪表盘相机记录下来的数据集，希望能够对驾驶员的行为进行分类。

本项目将会通过构建卷积神经网络模型，解决这个分心司机识别的分类问题。

二、问题陈述

我们拥有驾驶员的图像信息，这些图像信息记录的驾驶员在车中正在做的动作（发短信、吃东西、打电话、化妆、转身谈话等等）。我们的目标是预测每一个图片的司机在做什么的概率。

10 种预测结果：

- c0: safe driving
- c1: texting - right
- c2: talking on the phone - right
- c3: texting - left
- c4: talking on the phone - left
- c5: operating the radio
- c6: drinking
- c7: reaching behind
- c8: hair and makeup
- c9: talking to passenger

项目中的训练集和测试集均由 State Farm 公司提供。其中训练集 22424 张图片，测试集 79729 张图片。

三、评价指标

结果使用“multi-class logarithmic loss”来评估。每一个图片已经被分为其中一个类。 对于每一个图片，将会有每一种类别的预测概率。 评估公式如下：

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

其中 N 是测试集中的图片数量，M 是图片的分类标签的数量，log 是自然对数，如果观测到的 i 属于类别 j，y_{ij}是 1，否则为 0。p_{ij} 是观测到的 i 属于 j 的概论。

其中一个给定的图片的概率和不需要等于 1，因为它们在评分前被重新调整过（每一行被除以行的和）。

II. 分析

一、数据的探索与可视化

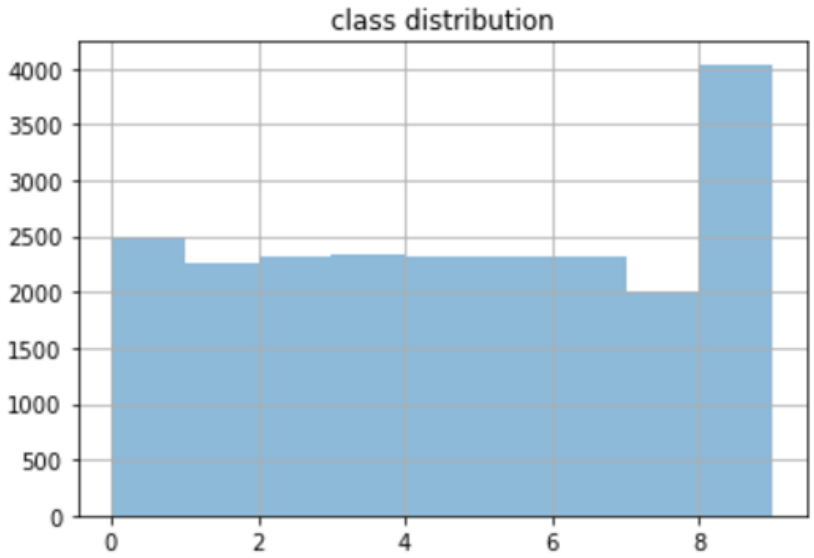
数据集来源于 State Farm 保险公司，并发布于 kaggle 数据分析比赛网。数据可以从如下网址下载: https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

文件描述：

- imgs.zip - zipped folder of all (train/test) images
- sample_submission.csv - a sample submission file in the correct format
- driver_imgs_list.csv - a list of training images, their subject (driver) id, and class id

其中训练集有 22424 张，测试集有 79729 张图，每张照片都是 640x480x3。其中收集了 26 个司机的 10 种不同的驾驶状态。每一种的状态样例图如下图：

这十种状态在训练集中的分布情况：



从图中可以看出与乘客谈话的分布数量最多，有 4000 多张图片。

二、算法和技术

根据以上的分析得知，分心司机识别本质上是一个机器学习中的分类问题，我们的训练集有标签（c0, c1, c2.....），所以又同时属于监督学习的范围。用于分类的算法有不少，例如决策树，支持向量机（SVM），朴素贝叶斯分类器，神经网络等。这些算法没有绝对的优劣之分，本题选用神经网络来进行分类。

卷积神经网络

神经网络的灵感来源于大脑中的神经元间的信号传递过程。神经元从多个树突中获取输入信号，然后通过轴突输出，传入下一个或者多个神经元作为下级的输入信号之一。

神经网络中的神经元本质上是一个非线性函数。所有的输入首先根据神经元中的权重进行对应相乘，相乘后求和。这些权重也就是神经元的参数。进行加权求和后，输入到整流线性单元（ReLU）得到输出。

神经网络通过反向传播算法更新权重，使用交叉熵作为损失函数，利用神经网络的预测值与真实值的差别计算损失函数。通过损失函数对每一个权重求导，利用梯度下降法更新权重，减小损失，让神经网络的结果逼近真实值。

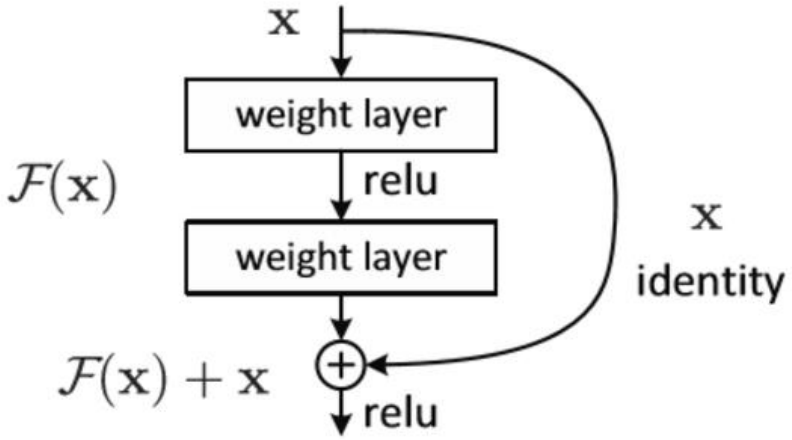
卷积神经网络是神经网络的一种，它受到了相邻神经元的感受野相互叠加的启发，后层的神经网络的输入只与前一层局部的输出有关。这一部分区域也就是感受野。因此卷积神经网络在卷积过程中，只响应了局部的输入，在后续的隐藏层中，这些隐藏层计算结果到输出层，逐步具有全局性，从而在输出层做出判断。

VGG16

VGGnet 是 Oxford 的 Visual Geometry Group 在 ILSVRC 2014 上的相关工作，主要工作是证明了增加网络的深度能够在一定程度上影响网络最终的性能，如下图，文章通过逐步增加网络深度来提高性能，虽然看起来有一点小暴力，没有特别多取巧的，但是确实有效，很多预训练的方法就是使用 VGG 的模型（主要是 16 和 19），VGG 相对其他的方法，参数空间很大，最终的 model 有 500 多 MB，所以训练一个 vgg 模型通常要花费更长的时间，所幸有公开的预训练模型让我们很方便的使用。

Resnet50

Resnet50 是一个有 50 层的深度残差网络。这个网络由微软研究院的 Kaiming He 等人在 Deep Residual Learning for Image Recognition 一文中提出来的。与传统的卷积神经网络相比，Resnet 使用了一种残差单元，将前层的输出与前层的输入相加，即 f(x)+x，作为后层的输入。一个残差块的结构如下图：



引入这个残差单元可以解决由于网络深度过大导致的梯度消失问题，即解决了深层网络与浅层相比，准确率反而下降的问题。深度残差网络能够充分发挥了卷积神经网络的深度优势。这个网络在去年各类 DCNN 比赛中获得了优异的成绩。

Tensorflow 与 Keras

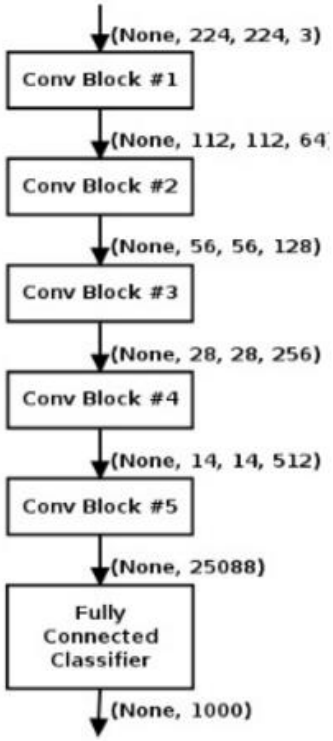
项目使用谷歌开源的深度学习框架 Tensorflow，并使用 Keras 作为核心库，配合其他众多协助计算的库，例如 numpy, cv2, sklearn 等。同时我们利用 Tensorflow 的 GPU 版本，使用 GPU 为计算加速。

三、基准模型

使用 VGG16 即有 16 层的深度卷积神经网络，并使用 imagenet 预训练初始化权重。
VGG16 的结构如下：

Keras VGG-16 Model

```
( 0, 'input_6',      (None, 224, 224, 3))
( 1, 'block1_conv1', (None, 224, 224, 64))
( 2, 'block1_conv2', (None, 224, 224, 64))
( 3, 'block1_pool',  (None, 112, 112, 64))
( 4, 'block2_conv1', (None, 112, 112, 128))
( 5, 'block2_conv2', (None, 112, 112, 128))
( 6, 'block2_pool',  (None, 56, 56, 128))
( 7, 'block3_conv1', (None, 56, 56, 256))
( 8, 'block3_conv2', (None, 56, 56, 256))
( 9, 'block3_conv3', (None, 56, 56, 256))
(10, 'block3_pool',  (None, 28, 28, 256))
(11, 'block4_conv1', (None, 28, 28, 512))
(12, 'block4_conv2', (None, 28, 28, 512))
(13, 'block4_conv3', (None, 28, 28, 512))
(14, 'block4_pool',  (None, 14, 14, 512))
(15, 'block5_conv1', (None, 14, 14, 512))
(16, 'block5_conv2', (None, 14, 14, 512))
(17, 'block5_conv3', (None, 14, 14, 512))
(18, 'block5_pool',  (None, 7, 7, 512))
(19, 'flatten',      (None, 25088))
(20, 'fc1',          (None, 4096))
(21, 'fc2',          (None, 4096))
(22, 'predictions',  (None, 1000))
```



这个模型使用预训练权重训练，将训练集（22424 张）分割成训练集 17920 张和验证集 4485 张，并设置 batchs size 为 32，对所有训练集图片训练 20 次时只有训练集 98%/验证集 60%的准确率。

III. 方法

一、数据预处理

图片像素 680x460 所记载的信息量很大，而实际上进行分类所需要的的信息不需要如此大的图片信息，受限于计算机硬件条件的限制，计算机处理高像素的图片速度很低，而且读取图片所占内存很大，所以我们需要将图片进行线性压缩。

由于我们使用预训练模型来初始化权重，并采用了"imagenet" 作为预训练集，根据 imagenet 预训练的描述，输入的图片应该以像素均值作为中心。而中心像素值以 BGR 排序应为：[103.939, 116.779, 123.68]。所以所有图片都需要减去这个中心像素值。

数据集中含有驾驶司机的名单列表，其中包括了每名司机所有的图像名称。分心司机识别不需要对驾驶员进行识别，与驾驶员的名字没有关系，所以训练时舍弃这一特征，仅作为显示提示作用。

二、执行过程

导入数据

定义了一个“load_train”函数，按顺序从十个文件夹中依次读取，并缩小图片尺寸为 224x224x3 并导入内存。使用 opencv 的读取函数于尺寸更改函数，默认的图像通道顺序是 BGR。

创建训练集

将图像矩阵转换为 numpy 矩阵的 uint8 格式，创建标签。将数据中的像素格式转化为 16 位浮点型。然后进行训练集的标准化，将训练集数据按照 BGR 通道顺序依次减去均值[103.939, 116.779, 123.68]。

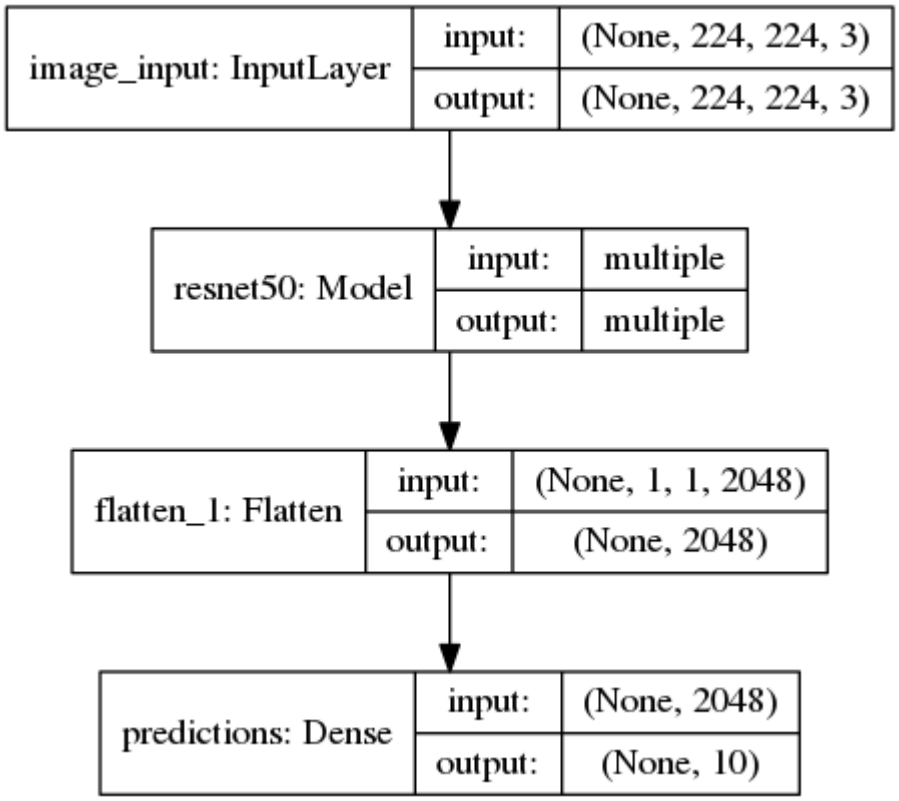
定义模型

使用 keras 库，导入 Resnet50 模型，去掉全连接层，使用 Dense 函数定义输出，softmax 作为激活函数。使用 adam 优化器，学习率设置为 0.001，使用交叉熵作为损失函数。

模型可视化

可视化模型的结构，并显示每一层输入输出的空间大小。

从下图中可看出，模型输入的是 224x224x3 的图片，经过 Resnet50 传入 flatten 层将多维数据一维化，为输入 Dense 全连接层过渡。输出层也是全连接层，输出十种结果。



训练模型

使用 K 折交叉验证的方法，同时对数据进行清洗，打乱顺序，随机将训练集按照 1：0.2 的比例分为训练集和验证集。使用打乱的 5 个不同的训练集分 5 次训练模型。由于时间的缘故，每一轮训练的最大次数设置为 40 次。Batch size 为 32，即在每一轮训练中，每次取出 32 张图片进行训练。

设置早期停止。神经网络的训练不是无休止的，我们需要确定训练应该在何时停止。使用 keras 库中的 early stopping 可以解决这个问题。最初的训练由手动的训练次数决定，我设置了 20 次。但是有些情况下模型的表现并不是特别好，所以我将训练次数提升到了 40 次。而且训练过多也会导致模型过拟合的可能性增加。所以我设置了早期停止，根据每一次训练的验证集损失量(loss)，决定是否停止训练，当超过 10 次没有减小时，停止训练。

设置 Model Checkpoint，这个函数可以让整个训练过程只保存损失量最低的模型。这样做可以避免了大量的权重参数数据存储工作，提高运行效率，节省时间。

预测测试集

我们使用了 5 折训练后的 5 个模型。在预测前将测试集的图片进行 resize 和减均值的处理。然后这 5 个模型分别对测试集进行预测，5 个预测结果取平均值作为最终的预测结果。

可视化预测

使用 5 折的其中一个模型对测试集中的两个样本进行可视化预测。

三、完善

VGG16

项目一开始我采用了 VGG16 模型，对图像进行 resize 和标准化，标准化的过程是所有像素除以 255 进行归一化，在不加入 K 折交叉验证的情况下训练，这个模型每一轮训练的时间大约在 360 秒左右，在测试集中的测试得到的 log_loss 结果是 0.65810。

然后我采用了 5 折交叉验证，即分 5 次训练模型，在训练完后得到了 5 个模型权重。在预测的时候，将图片分别输入这 5 个不同权重的模型，得到的预测概率相加后除以 5 取平均值作为最终的预测结果。这个结果在测试集中的得到的 log_loss 结果是 0.42475。与不加 k 折交叉验证相比，提升了 0.22 左右。

当我使用了 imagenet 图片的标准化方式后，我的模型预测成绩再提升到了 0.40071，这个成绩提升得并不是很多。

Resnet50

VGG16 如果想继续提高成绩，则必须加入更多次数的训练，由于机器和时间的限制，继续使用 VGG16 模型是不划算的。而且 VGG16 模型相比 Resnet50 而言，是一个比较老旧的模型，它的层数不是特别深，对数据的的抽象提取能力不如 Resnet50。所以我改用 Resnet50 继续进行优化。

使用 Resnet50，重复以上的过程得到了更好的结果。不使用 K 折交叉验证的情况下，Resnet50 的得分为 0.46433。而使用了 5 折交叉验证的情况下，得分为 0.27105.

根据以上的不断完善，最终选用 Resnet50 作为最后的模型，数据预处理使用减去 BGR 均值的方法，并使用 K-fold 交叉验证产生的 5 个模型对测试集进行预测后取平均值作为最后的预测结果。

IV. 结果

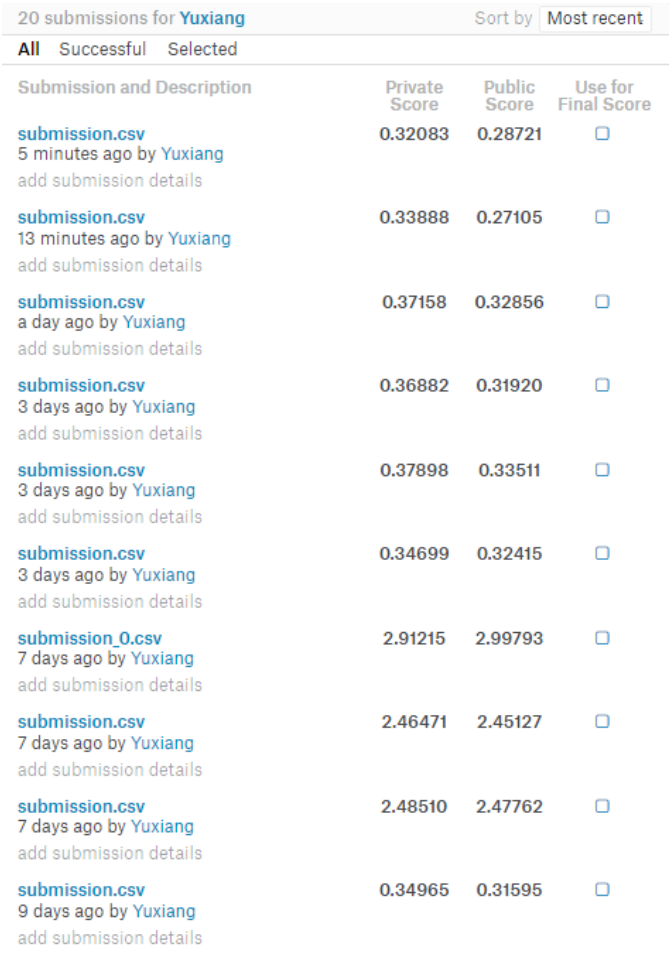
模型的评价与验证

最终我选用的模型是 Resnet50。最终模型实验过的模型对比如下图：

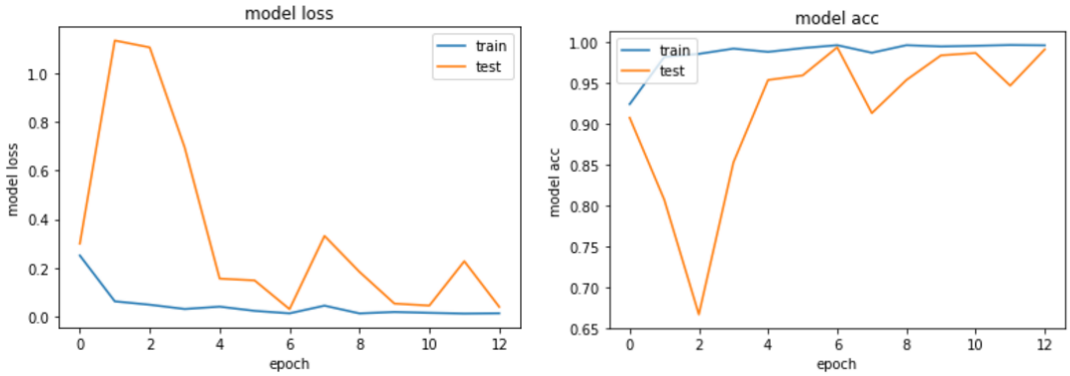
方案	训练集	验证集	测试集
VGG16 归一化	acc:0.9992 loss:0.0031	acc:0.9955 loss:0.1930	Public loss:0.64133 Private loss:0.67315
VGG16 5-FOLD 归一化	acc:1.0000 loss:0.0001	acc:0.9958 loss:0.0225	Public loss:0.42475 Private loss:0.48220
VGG16 5-FOLD 减均值	acc:1.0000 loss:0.0001	acc:0.9958 loss:0.0233	Public loss:0.40071 Private loss:0.48220

RESNET50 减均值	acc:0.9964 loss:0.0129	acc:0.9938 loss:0.0301	Public loss:0.46433 Private loss:0.57844
RESNET50 5-FOLD 减均值	acc:1.0000 loss:0.0000	acc:0.9978 loss:0.0133	Public loss:0.27105 Private loss:0.33888

在 Kaggle 网站的部分提交情况如下图所示：

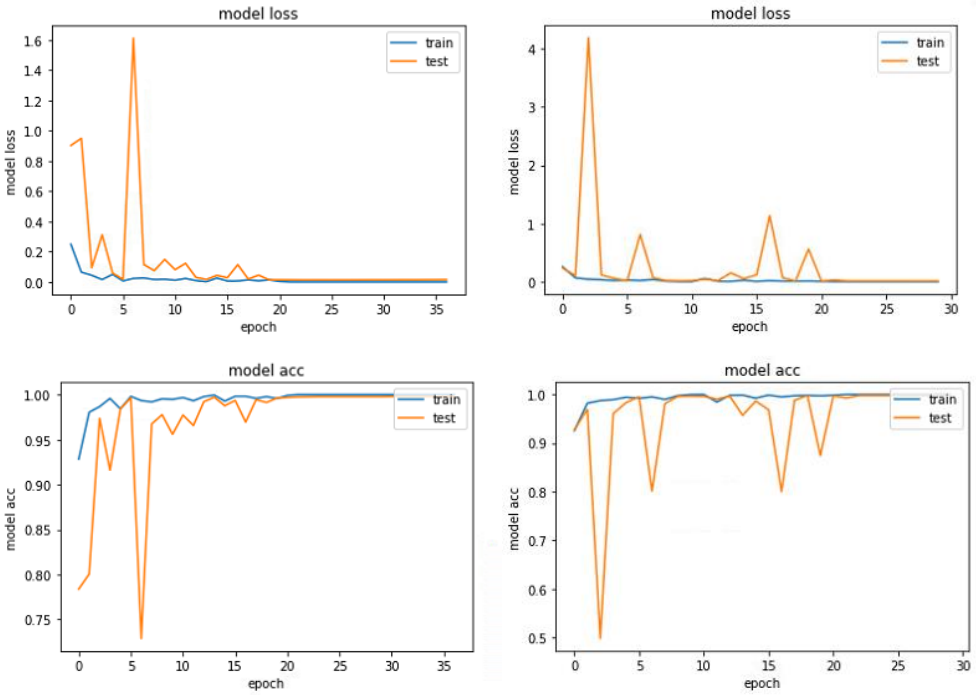


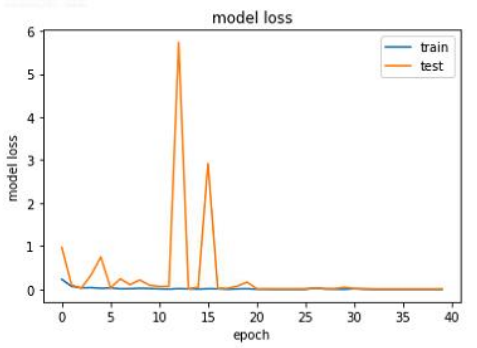
5 折交叉验证中的其中一个模型在训练时的 loss 与 accuracy 如下面两张图所示：

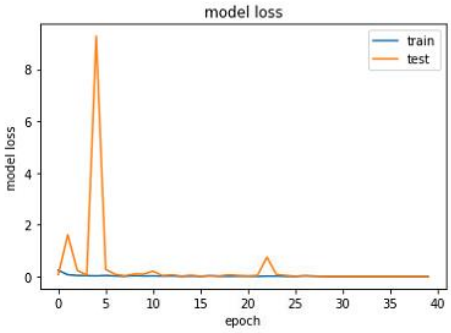


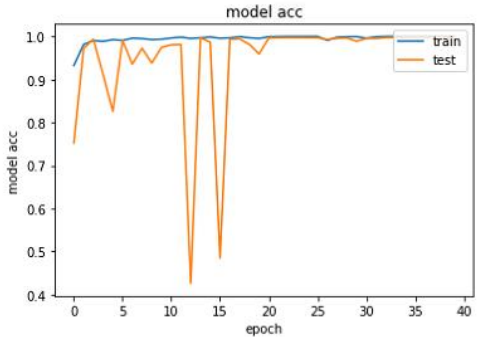
从图中我们可以看出，这个模型的训练在第十二轮的时候停止了，可以看出早起停止在其中起到的作用。训练集的训练很早就趋于收敛了，但是验证集的波动还是比较明显，但是也有明显的收敛趋势。

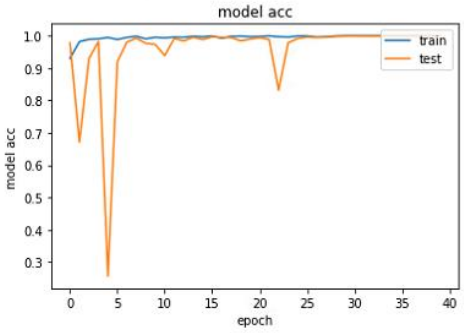
如果我们继续增加迭代次数，并设置早起停止为 logloss 不降低的 10 轮后停止。则我们可以得到明显收敛的结果。其中 5 折训练的 5 个模型 loss 和 accuracy 收敛图如下所示：

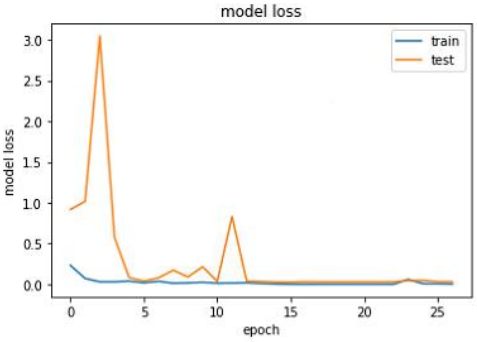


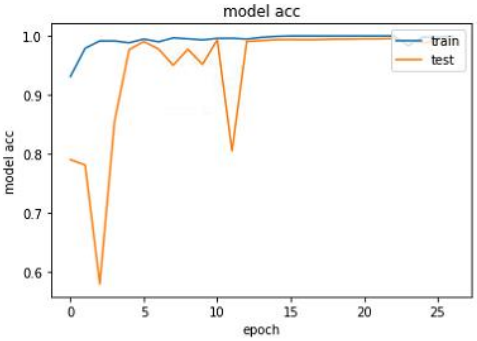












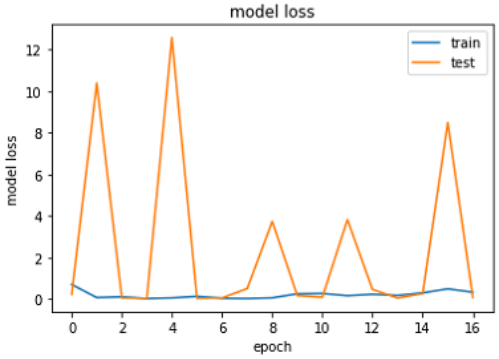
由于测试集的标签 kaggle 网站并没有向我们提供，所以我们无法使用测试集对结果进行验证。但是把预测结果提交给 kaggle 网站，网站就会根据你提交的预测结果计算并显示你的 multi-class log loss 作为评判标准。

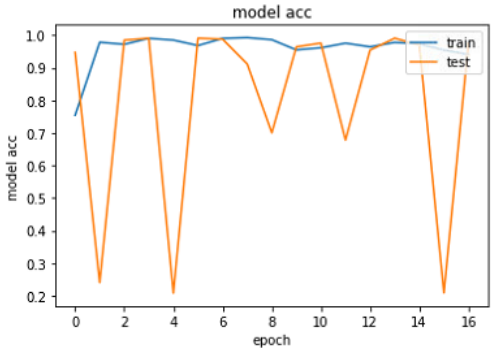
我的模型在 kaggle 中最好的得分是 0.27105，在公示的排名 leaderboard public 中，这个成绩能排到 166 名；在 private 排名中最好的成绩 0.32083，能排名到 213 名。

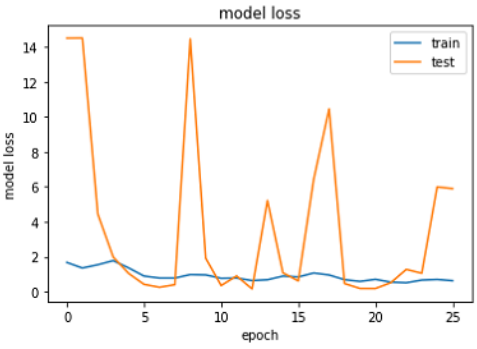
合理性分析

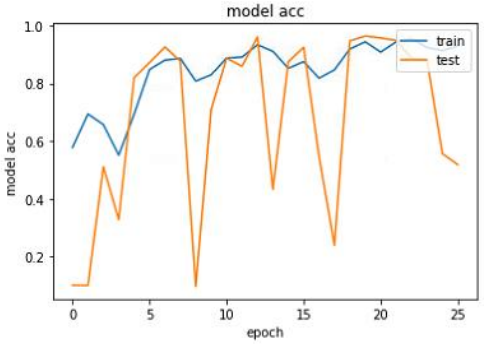
从训练集和测试集中的图片我们可以看到，驾驶员体型有胖有瘦，样貌迥异，所以模型的结果具有较好的鲁棒性。与基准模型 VGG16 相比（logloss=0.64133），使用 Resnet50 具有更低的 logloss(logloss=0.46433), 预测的准确度更高。在使用 K 折交叉验证并交叉测试后，有效的减少了模型的过拟合，进一步提高了准确率降低了 logloss(logloss=0.27105)。可以说我们最终的模型相比于基准模型准确率有了很大的提升，是解决分心司机识别问题的更优秀的模型。

过程中我还尝试了使用 Xception, InceptionV3, Resnet50 这三种模型进行学习并集成他们的结果。但是在训练 Xception 和 InceptionV3 的时候，我发现它们的收敛结果并没有 Resnet50 优秀。其中 Xception 和 InceptionV3 的训练结果分别如下图所示：









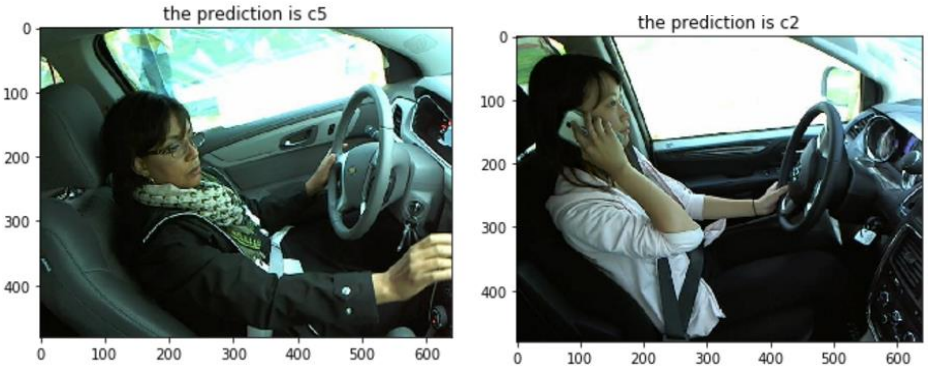
所以我对 Resnet50 的结果最满意，最终决定使用这个模型。

V. 项目结论

结果可视化

我们可以将测试集的图片输入模型来查看预测结果。虽然测试集中的图片都没有加标签，但是我们可以将预测结果与我们肉眼所看到结果相对比，检查预测结果是否可靠。

例如我们输入测试集中的两张图片“img_1.jpg”，“img_2.jpg”它的测试结果可视化如下图所示：



我们可以看到左图中的驾驶员正在用右手调试汽车的音响，而预测结果是“c5”代表“operating the radio”，所以预测结果正确。右图中的司机正在用右手打电话，预测结果是“c2”代表“taking on the phone - right”，预测结果正确。从这两个预测结果可以看出模型的预测符合我们期待的结果，模型表现优异。

对项目的思考

本项目目标是实现识别分心司机驾驶状况的 10 种不同的状态。首先我们构建了基准模型，使用预训练模型 VGG16 作为起始模型，但是预测结果不是很理想，过拟合的情况比较严重。于是对数据进行减均值的处理，并且使用 K 折交叉验证后的交叉测试，使用 5 个不同权重的模型进行预测取均值，预测结果有了明显的改善。然而 VGG16 的结果不是非常好，需要训练的神经元数目庞大，运行时间长。为了进一步提高成绩，采用预训练的 Resnet50 模型，取得了良好的效果。

在这个过程中，由于本地机器的限制产生了不少的问题，例如内存和显存的问题。最终选用了亚马逊云服务器来进行运算，解决了内存和显存的问题。

这个项目让我加深了对深度学习知识的理解，对 keras 库的使用更加熟练，对解决深度学习处理图像分类的问题有了实际操作并解决的能力，学习并理解了 VGG, Resnet 这两个经典深度学习网络。

需要作出的改进

在数据方面，数据集的图片均是以一个角度捕捉的驾驶员状态，如果变化了拍摄角度，模型的预测能力会下降。

本项目对于 VGG 和 Resnet 模型本身没有进行大改动，仅仅尝试了在全连接层加入 Dropout。如果需要进一步减小过拟合，提高模型的泛化能力，可以尝试考虑在内部加入 Dropout。还可以对数据集进行处理，采用切块的方法对图片中印象最终判断识别特征进行识别，类似 cascade classifier.

由于本项目提供的训练集比较少，如果想进一步提高模型的表现，需要进行数据增强处理，例如对训练集中的图片进行分割重组，或者旋转翻转，产生更多训练集图片。