机器学习纳米学位

毕业项目 侦测走神司机

蒋似尧

2018 年 4 月

I. 问题的定义

项目概述

每年都有很多交通事故源于司机走神。Statefarm，美国的一家交通保险公司，在 kaggle 上悬赏机器学习算法，用来自动识别开车时走神的司机。通过摄像机拍摄到的司机画面，分辨司机的不同状态，包括安全驾驶，打电话，和其他乘客说话等。这有助于为司机提供更好的保险服务，而且可以用于自动报警，提示司机安全驾驶

问题陈述

通过捕捉到的司机驾驶时的画面，通过深度学习来进行图像分类，辨识司机的状态。司机的状态大致分为 10 类，是一个多分类问题。我们需要构建一个分类模型，输入一张司机的图片，计算出图片中司机处于每种状态的概率。最后模型的评估采用 cross entropy loss，对比测试图片的预测分类概率和实际分类的误差。

评价指标

本项目是一个多分类问题，典型的评价指标可以用 cross entropy loss。模型的输出层激活函数是 softmax，也就是 10 类的概率 p_{ij}。图片的正确分类是 y_{ij}。i 代表图片数量一共 N 张。j 代表类别数一共 M=10 类。损失函数可以用如下公式表示：

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

事实上 kaggle 会先做结果的预处理再计算损失。首先会做一步归一化，再采取以下方式避免出现极端情况 p=0 或 1：

$$max(min(p, 1-10^{-15}), 10^{-15}).$$

具体说明引自这里：
https://www.kaggle.com/c/state-farm-distracted-driver-detection#evaluation

II. 分析

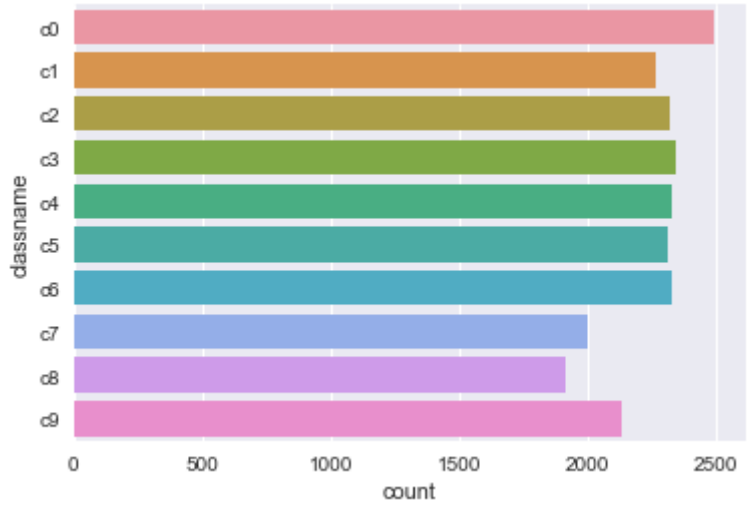
数据集探索

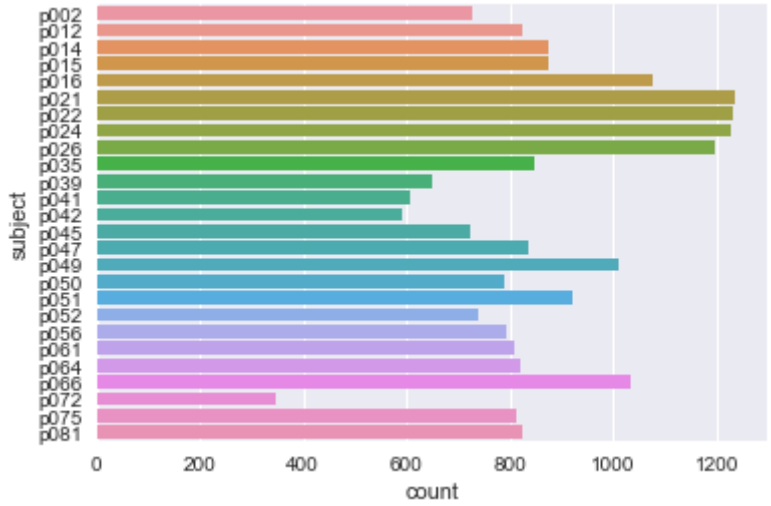
数据集包含测试和训练两部分。训练集 22424 张图片已经分类存放在 10 个文件夹内，文件夹名就是 label。一共分为 10 类。具体如下：

- 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

每类包含 2000 多张司机驾驶中的 RGB 图片。所有 10 类图片都是对 26 名不同的司机拍摄的，每名司机在每一类文件夹下都有若干张图片。司机的 id 号，对应的图片文件名，以及对应的类别在 driver_imgs_list.csv 文件中提供。司机 id 和 10 种分类的分布分别如下图，可见分别还是比较均匀的：

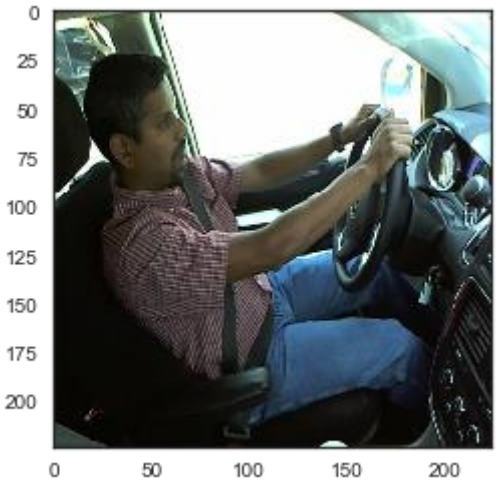




值得注意的是测试集图片中的司机和训练集中的不同，也就是 26 名用于训练的司机不会出现在测试集中。这是不同于其他图片分类项目比如猫狗大战的地方。

对于 1 名司机建立好的模型，必须能成功的预测另一名没见过的司机的分类。这也是符合实际需求的，在实际侦测中肯定要适用于陌生的司机。为此必须先按照司机 id 分类，在训练和验证中使用不同的司机，这样才能让模型学到在不同的司机身上都能有效分类，而不是只认识这 26 个司机。

另外 OpenCV 通道顺序读取时需要注意，把 BGR 转换成 RGB，才能读出原图如下，否则色彩是不对的：



在项目进行中，逐一观察训练图片后，发现 p081 号司机的所有类别图片中，都会带有头向副驾驶偏的倾向。例如下图是 c0 也就是正常驾驶状态的一张图片：

可以看出这位司机显然在和副驾驶说话，而非安全驾驶。经过观察其他类别的文件夹里这位司机的图片也有大量类似的错误标注，因此在项目中弃用了这个 id 的所有图片。

算法和技术

采用迁移学习的方法，导入在 ImageNet 上预训练好的模型如 ResNet，Inception 等，放开一些层进行 fine tune。训练若干个这样的模型，并采用类似 bagging 的思想进行模型融合。每个模型抽取一些司机 id 作为训练集，其他司机为验证集，进行训练。下一个模型换一批验证司机。这样训练得到多个模型后，对他们在测试集上的结果取平均值作为融合后的结果提交。

基准模型

这个问题最简单的基准模型就是等概率的预测图片状态属于 10 类中的一类，也就是每类的概率都是 0.1。这个模型没有对数据集进行任何学习，有 10%的正确率。Sample submission.csv 文件就是做了这样一个预测，提交到 kaggle 后损失函数为 2.3。

III. 方法

数据预处理

本项目利用 opencv for python 做图片的预处理和数据增强。具体预处理的方式根据选择的预训练模型决定。如果选择 ResNet，则需要把图片裁剪到 224*224，通道顺序保持为 opencv 读取的 BGR 顺序，像素减去每个通道的均值。如果选择 Inception，Xception，VGG 等模型，则需要把尺寸裁剪到 299*299，通道顺序转换为 RGB，像素归一化到-1 和 1 之间。

训练集中的图片每个类别约 1000 多张，相比测试集是比较少的，为了避免过拟合问题，采取以下数据增强技术，以增强模型对不同图片的泛化能力：

1. 对原始数据图片围绕中心点随机旋转-10 到 10 度。

2. 在水平方向上平移-50 到 50 个像素。

3. 随机选取同一分类里的两张图片，各自取其左右半边，然后拼接在一起。这样可能是模型注重于学习对分类有关键作用的点如手部，脸部朝向。而不是学习不同司机的样貌。

4. 色彩 PCA 转换。参考 AlexNet 的做法，分析出色彩的 principle component，并做随机转换，以改变图片的色彩。

5. 对图片在 HSV 空间随机变换，改变图片的亮度。

另外，对于 p081 司机的图片全部没有采用，以避免误标注对模型训练的影响。

因此训练集图片的总数降到 21601 张。

执行过程

这个项目的一大难点是训练司机和测试司机不一样，因此要训练模型能够识别不同的行为模式，而不是不同司机的脸。所以要按照司机的 id 来切分训练集和验证集。我在不同模型里切分出来的验证司机不一样，这是参考了 bagging 的思想，获得多样的模型，以待融合。具体执行时把 driver_imgs_list 分成两个列表。一个是训练列表，里面是所有训练集司机的图片文件名。另一个是验证列表，里面是所有验证集司机的图片文件名。然后把两个列表都随机打乱。 这两列表里面保存的都只是 excel 文件里的一行行数据（包括文件名，分类，司机 id），而不是图片本身。等训练时通过生成器读取图片，这样节约内存，不用把所有图片一次性都放进内存。

这里使用的图片生成器是我自己编写的，每次 yield 一批指定数量的经过数据增强的图片和分类标签，使用模型的 fit_generator 方法进行训练。这里使用自己定义的生成器，因为训练数据集是按照分类放在 c0 到 c9 文件夹里，而不是按照

司机 id 分的。如上所述，每一个类别文件夹里的图片需要根据司机 id 切分成训练集和验证集，而 keras 的图片生成器无法实现这一功能，所以使用了自定义的生成器。

搭建模型时，为了模型的多样性，尝试使用了多种预训练模型，最后效果比较好的有 3 种：ResNet50，InceptionV3，Xception。在预训练模型最后接全局平均池化层，dropout，全连接分类器输出 10 类的概率。为了减轻过拟合，在全连接层采用了 L2 正则。锁定一些层，对预训练模型进行再训练。通过调节锁定的层数，也可以获得效果不同的模型。对同一种模型，选择锁定层数效果最好的一个，换一批验证司机，以此增强模型对不同司机的泛化能力。最后一共训练得到了 12 个不同的模型，总结在下表中：

模型编号	文件名	使用的结构	锁定层数	验证司机编号
1	M1	ResNet50	77/174	52, 56, 61, 64
2	M2	ResNet50	35/174	52, 56, 61, 64
3	M3	ResNet50	90/174	随机的 4 名司机
4	M4	ResNet50	77/174	2,12,14,15
5	M5	InceptionV3	160/314	52, 56, 61, 64
6	M6	InceptionV3	130/314	52, 56, 61, 64
7	M7	InceptionV3	180/314	52, 56, 61, 64
8	M8	InceptionV3	160/314	2,12,14,15

9	M9	Xception	70/135	2,12,14,15
10	M10	Xception	55/135	2,12,14,15
11	M11	Xception	80/135	2,12,14,15
12	M12	Xception	70/135	16，21,22,24

对训练集图片完整遍历一遍视为一个 epoch，最多训练 5 个 epoch。训练过程用了 early stopping，当验证集 loss 不再下降时停止训练并保存最佳模型。另外还使用了 learning rate scheduler 来自动调整学习率，每一代学习率自动降低 10 倍。

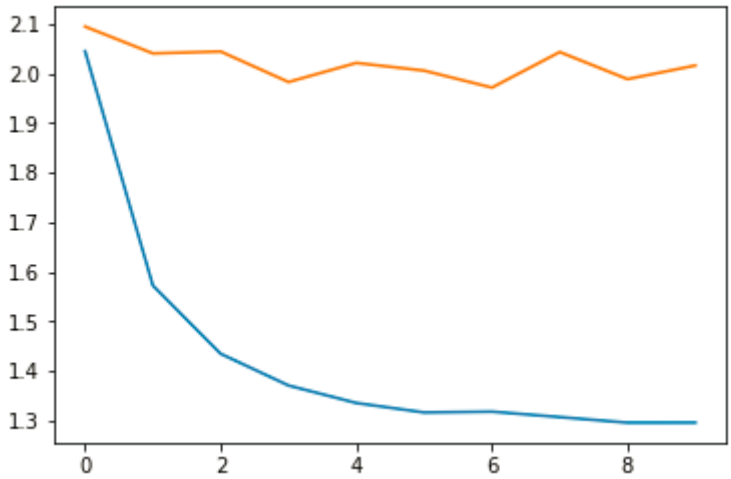
每个模型训练完成后，利用 keras 自带的 ImageDataGenerator 逐批次生成 test 数据并做 predict，将预测的结果即每张测试图片在每个分类的概率写入 CSV 文件。

最后对选定的若干个结果比较好的模型进行融合。方法是最简单的求均值，使用 Merger.ipynb 文件把选出的若干个模型对应的 CSV 文件里的输出结果求平均值。

把融合后的 CSV 文件提交上 kaggle 即可获得最终成绩。

完善

一开始预训练权重全部锁定只训练最后的分类器，训练的 loss 始终无法降低到 1 以下，类似如下的 loss 曲线，蓝色为训练集 loss，橙色为验证集 loss。



以为是造成了过拟合，所以没有考虑放开前面的一些层进行 fine tune，而是尝试了一些避免过拟合的方法比如加入 L2 正则，做数据增强等，但是效果一直不佳。后来参考了论坛和其他同学的一些做法，发现这个现象反而是模型拟合能力不够，loss 才不下降。所以放开预训练权重进行 fine tune，进一步增强模型拟合能力，终于取得了重大突破。

做数据增强时，发现如果加入色彩 PCA 变换和 HSV 空间随机变化，训练的 loss 无法下降。考虑到这些图片是连续拍摄的，和测试集的色彩差不多，所以这两项增强技术可能并无必要，实际项目中没有使用，仅仅进行了随机偏移和旋转，以及左右拼接。

在尝试不同的模型时，还使用过 VGG 和新版 keras 包括的 NasNet。使用 VGG 预训练模型即使全部层都放开训练，也无法让 loss 下降，可能是模型层数相对较少，拟合能力不够。而 NasNet 训练集 loss 正常下降，但是验证集 loss 无法下降，明显是过拟合了，可能锁定更多的预训练层数可以缓解这个问题。实际项目中这两个模型都未使用。

IV. 结果

模型的评价与验证

12 个单个模型，以及一些尝试的融合模型的最终输出 csv 文件提交到 kaggle 上计算测试集分数，public score 和 private score 分别如下（保留 2 位小数）：

模型编号	最后一代 val_loss	public score	private score
1	0.28	0.42	0.35
2	0.82	0.99	0.89
3	0.52	0.73	0.51
4	0.39	0.80	0.53
5	0.43	0.68	0.53
6	0.84	NA	NA
7	0.57	0.76	0.64
8	0.66	0.66	0.62
9	0.22	0.37	0.33
10	0.35	0.64	0.52
11	0.35	0.34	0.35
12	0.14	0.50	0.36
1,9,11,12	NA	0.27	0.23

1 到 12 所有（除 6）	NA	0.36	0.29
1,5,9，11,12	NA	0.29	0.24
9,11,12	NA	0.31	0.26

可见最终成绩最好的模型是选取单个成绩最好的 4 个模型（编号 1,9,11,12）进行融合。如果融合的模型太多，一些比较差的模型会拉低总成绩。如果选取更少的模型，比如不含模型 1，模型的多样性会损失，可能降低了泛化能力。

最终成绩提交到 kaggle 上，public score 排名 166，private score 排名 107，成功完成预定目标进入前 10%。

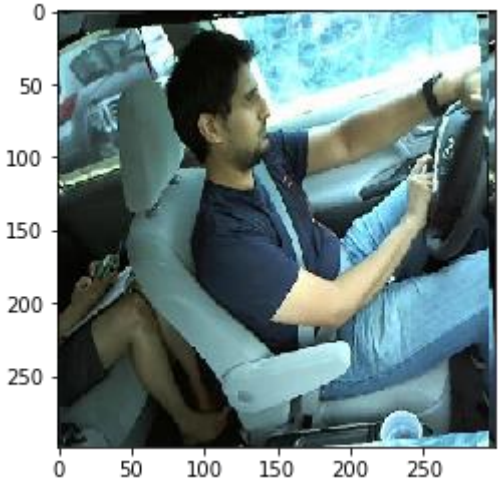
V. 项目结论

结果可视化

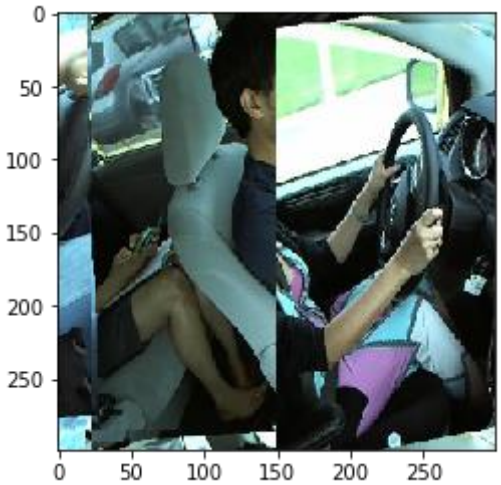
在本项目中，一个重要的防止过拟合的手段就是数据增强。具体用到的增强方式有 3 种，图片旋转，便宜，以及左右拼接。在 notebook 中展示了一些经过增强后的图片实例。一个注意点是尽管输入不同模型的图片经过了不同的预处理，像素值范围各异。但是使用 matplotlib 画图时，输入的像素值必须调整到 0-255 的整数或者 0-1 的浮点数，否则无法正常显示图片。

1. 绕中心随机旋转-10 到 10 度，在水平方向上平移-50 到 50 个 pixel。这个范围是观察一些训练和测试图片后，根据不同图片之间拍摄的角度和位置差异，决定在这个范围内做随机增强。目的是加强模型的泛化能力，防止过拟合。

   下图是一个经过增强的图片 img_1078，已经 scale 到了 299*299 的尺寸。



2. 另外还有 50%的概率随机选取另一张同类里的图片，将两张的左右各半边拼接在一起。这也是为了训练模型对分类的关键部位进行学习，而不是记住司机的样子。下图是一个例子，选择 c0 中的 img_1078 和 img_327，截取他们的各半边拼接在一起。图中从两手放在方向盘上，头朝向前方可以正确判断这是正常驾驶。



对项目的思考

这个项目使用了 CNN 进行图像识别，总体上完成了数据分析，预处理，模型构建，模型训练，验证结果，参数改进这整个流程。最终结果达到了预期的目标，能有效的对陌生的司机判定其驾驶状态类别。

过程中比较有挑战性的是自己编写了图片生成器，逐批次给模型提供训练图片，在这个过程中掌握了用 opencv 进行图像处理，深度学习中常见的预处理方法和图片增强技术。另一个启发是对学习过程中的过拟合，欠拟合的区分更加谨慎，不能光从 loss 曲线的外形上做出判断。这也是我自己在项目中遇到了难点，通过和同学交流经验学到的。

可以作出的改进

本项目可能还可以通过以下方法改进。

1. 使用更多模型如 InceptionResNet，NasNet 等增加融合模型的多样性，进一步优化其参数，从而提高最终模型的成绩。

2. 训练模型时，手动调节学习率。当验证 loss 不下降时，进一步降低学习率，继续学习。这样可能耗费的训练时间比较多，但是模型可以有更进一步的优化。

3. 使用软连接对训练和验证图片按照司机 id 区分。这样就可以使用 keras 自带的图片生成器，相比我自己写的生成器更方便而且可能对 GPU 训练更优化。这一点项目开始时没有想到，不过也使得我通过这个过程学到了更多东西。