机器学习纳米学位

走神司机检测

行者周

ZNBJTU@126.COM

2018-04-24

1 问题定义

1.1 项目概述

有过开车或者乘车经验的人通常会遇到一种情况就是，在一个交通路口，交通灯变为绿色但是我们前面的车并没有往前走，在后边司机的鸣笛提醒下或者自己后知后觉才向前行驶，这给后边车辆造成了不必要的堵塞。还有一些情况下，我们会发现前车速度突然减了下来，并且有可能伴随左右摇晃的情况。

之所以发生上述这些影响交通安全和效率的状况，很多情况是因为司机在驾驶过程中没有集中注意力，而是一边开始一边做起了诸如发微信、刷社交网络或者拿着手机打电话的事情。据有关报告，在五分之一的汽车事故是由于司机走神引起的，每年大概有 425000 人因此受伤，3000 人因此死亡[1]。因此需要有一些手段对司机的驾驶行为进行监控、判断以及提醒，减少因此发生的交通事故。

该项目旨在通过学习标记好的司机驾驶行为图像数据，对测试集中司机的行为进行检测，属于深度学习中的一个分支——计算机视觉范畴的问题。

之所以选择这样一个项目有以下几个原因：

- 该项目具有很强的实用价值和现实意义，好的解决方案能够减少交通事故的发生
- 该项目相较于其他项目更有挑战性，本人对深度学习十分有兴趣
- 该项目跟自身的业务方向契合

卷积神经网络应用于图像分类和识别已经有很长的历史了，但是直到近几年才得到了广泛的关注和发展。原因在于近些年尤其是移动互联网时代的到来，研究人员可以非常低成本地获得各种各样的数据，加上计算机和存储性能的大幅提高，应用卷积网络进行深度学习的门槛也越来越低。当然，深度学习领域研究的发展和进步也得益于 Kaggle、ImageNet 这样高水平的比赛。很多深度学习界的经典模型都是在这种比赛中被提出来的，这些模型不但在各自的比赛中表现出了卓越的水平，而且被证明在很多其他问题上也能表现的不错，非常值得借鉴。而且很多框架比如 Keras 就集成了一些经典的模型，利用这些预训练模型做迁移学习，可以加快模型构建的速度，提升模型的表型。

AlexNet[2]就是一种经典的网络，是由 Krizhevsky 提出的，他是在 ImageNet 的比赛中提出了这种卷积神经网络，并且取得了非常优异的成绩，这吸引了更多的研究人员在处理大规模图像识别问题时把目光关注在深度神经网络中。Simonyan 提出了一种具有小尺寸卷积核的多层卷积网络（VGG）,参数更少精确率足够高，取得了业界领先的表现[3]。He Kaiming 提出了一种比 VGG 更深的网络 ResNet，尽管层数更多，但是参数反而更少，而且非常易于训练和优化，在 ImageNet 数据集上的表现比 VGG 更好[4]。 谷歌的工作人员提出了一种叫做 Inception 的网络，据说是受到了电影盗梦空间的启发，这种网络的优点是不用人为指定卷积核的尺寸以及是否需要池化，由模型自动选择[5]。

1.2 问题描述

该项目的数据集来源于 Kaggle，训练集是由包含了 10 种司机驾驶行为的图像组成的，而测试集是的图像是所有待分类的图像。很显然这是一个监督学习的多分类问题，我

们期望利用训练集训练得到一个准确的分类器，对测试集中的图像进行归类，归类的范围就是训练集中的 10 种司机行为。每个图像只可能是 10 种分类的一种。

所以我们的目标就是去构建和训练这样一个分类器，由于数据集是大规模的图像，鉴于深度学习模型在大规模图像分类和识别中的成功案例，本文选择卷积神经网络作为模型。

本文选择以及训练卷积神经网路的策略如下：

1. 研究经典神经网络模型论文，掌握经典网络模型的配置，在应用中的数据集特征以及性能表现。
2. 研究 Kaggle 论坛中该项目的一些前人经验，参考一下前任的思路。
3. 结合对经典神经网络的研究，以及对本项目数据集的分析，选择模型，可以选择多个模型。
4. 针对模型所需要的数据集特点，对本项目数据集进行预处理。
5. 利用预训练模型进行单一模型下的训练和调优，直到结果满足。
6. 如果在单一模型下的效果始终不满意，考虑 bagging 以及模型融合。
7. 如果还想进一步提高，可以进一步考虑数据增强以及其他一些改进措施，这取决于对于模型和数据的理解有多深。
8. 最终模型的评价结果是需要在整个测试集上进行预测，然后提交到 Kaggle 上计算 logloss，logloss 值越低，表明所训练的模型越准确。

1.3 评估指标

上文提到了，本文是一个多分类问题，本解决方案采用交叉熵做为评估指标，公式如下：

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

其中 N 是测试集中图像的数量，M 是分类标签的个数，log 是自然对数，y_{ij}当样本 i 标签为 j 是值为 1，否则为 0。p_{ij}为样本 i 归属为标签 j 的预测概率。

2 分析

2.1 数据探索及可视化

数据集是 Kaggle 上提供的，一共有 22424 张照片以及对应的标签信息。每张图像的内容都是一个司机驾驶一辆车伴随有各种各样的行为，比如发信息，吃东西，打电话等。

数据集可能的标签有 10 类，分别是：

- 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

数据集中已经去掉了创建日期等原信息，并且保证了同一个司机不同即出现在训练集又出现在测试集。数据集中 imgs.zip 是所有图像的压缩文件，driver_images_list.csv 提供了图像的的标签信息。训练集样本中各类的分布情况如下：

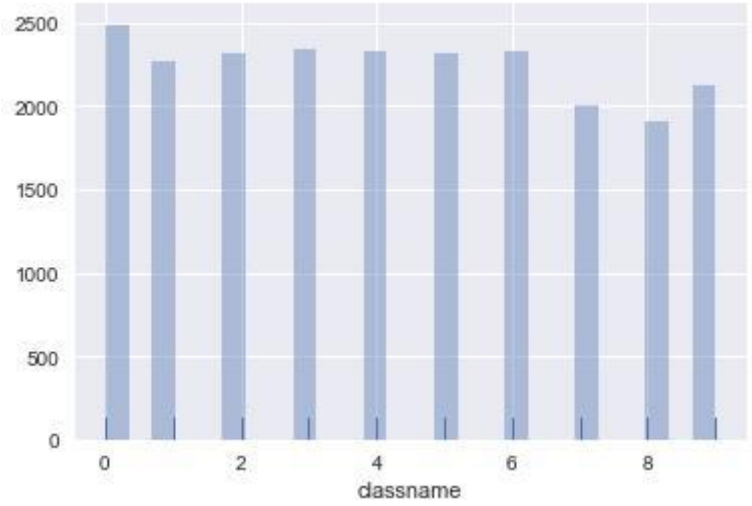

图 1 训练样本分布图-类别

如图 1 所示，训练集数据在每一类的分布较为均匀，不存在某一类特别多、某一类特别少的情况，即不存在数据偏斜类问题。

测试集的数据均位于 test 文件夹下，为了保证模型测试结果的有效性，训练集中出现的司机绝不可能在测试集中出现。这里需要注意的是测试集有 79726 个待分类的图像，几乎是训练集的 4 倍。所以如果出现过拟合严重的情况，可以考虑进行数据增强。

由于测试集和训练集司机不重合这一特点， 本项目在进行训练集和验证集划分时，需要按照司机来进行划分，这样能够使得在验证集的结果更有参考意义。

训练集图像在不同司机间的分布如图 2 所示：

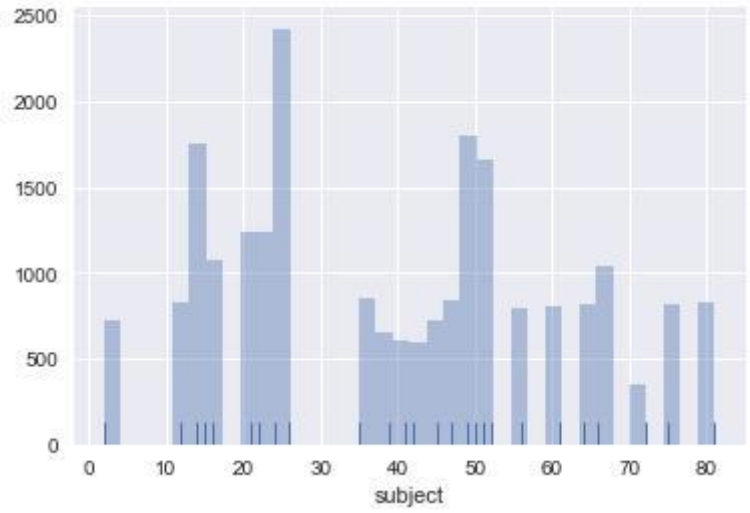

图 2 训练样本分布图-司机

如图所示，训练集在司机间的分布较为不均，因此不同的划分，可能产生较大的差异。

训练集和测试集均为如下图所示的彩色图像，对于彩色图像来说，其特征就是图像上每个像素的 RGB 值，每个图像的 RGB 数据就构成了模型的输入。而输出即模型的预测结果是一个归属于 10 种类别中各个类别的概率。最终期望的结果是测试集中每个图像归属于各个类别的概率值。典型样本图片为图 3 所示的 640*480 像素的彩色图像，图像是连续采集的，因此某些样本的差距非常小，尤其是同一个司机的相似度很高。因此在分割训练集和验证集时要特别注意按照司机进行分割，这样能够保证验证结果的有效性。

图 3 样本示例

2.2 基准模型

由于本项目是一个大规模图像数据的多分类问题，所以选择了卷积神经网络这一近些年在处理类似问题表现优异的模型。在众多优秀的基准模型中，选择了 VGG16 作为主要参考模型，VGG19 和 ResNet50 作为基准模型进行对比。其中 VGG16 的模型结构较为简单，尽管需要较多的参数，由于可以得到训练好的权重，所以可以排除计算消耗上的顾虑。而 ResNet50 尽管层数多，但是参数却更少，在某些数据集上的表现超过了 VGG16。而 VGG19 相较于 VGG16 稍复杂一些，通过对比不同模型的表现，能够加深对模型的认识，寻找调优的方向。

Keras 提供了这几种优秀的预训练模型，可以很方便的使用和集成。由于事前不知道哪个模型表现的更好，因此可以互为基准模型，本文将在三个模型下分别进行训练和调优，再对三种训练好的模型进行集成。此外本文还将以 Kaggle LeaderBoard 上的排名做为参考，目标是进入前 10%，即排在 1440 个成绩的前 144 名，对应在测试集的 logloss 表现需要比 0.2563 小。

3 方法

3.1 数据预处理

本文是基于预训练模型做迁移学习，所以预处理的方式跟各预训练模型保持一致。其中 vgg16、vgg19、resnet50 都是读取 224x224 的 RGB 信息然后转换为 shape 为（224,224,3）的数组，之后运用 keras 预处理模型内置的 preprocess_input 函数做预处理。

除了图像本身的预处理之外，本文还对数据读取的过程进行了特殊处理。首先是通过读取项目提供的 csv 表格，得到按照司机汇总的图像信息，目的在于之后把数据集按照不同的司机做 k 折划分。之所以要进行 k 折划分，是因为训练集数据较少，做一次划分训练得到的模型性能较差。而之所以要按照司机做划分，是因为同一个司机相似度极高，如果训练集和验证集均存在同一个司机，那么就很容易识别出来， 这样得到的模型在测试集会过拟合严重。项目本身为了保证模型有效性，训练集中出现的司机，在测试集中绝不可能出现。所以按照司机进行划分能够使得验证结果更加可信。

3.2 执行过程

在项目执行过程中遇到了很多困难，绕了很多弯路，进行了很多尝试，我对深度学习的理解也越来越深入。刚开始完全没有思路，就参照了知乎提取特征向量进行预测的方法，然后只做了一次训练集划分，也没有按照司机进行划分。在验证集的表现还不错，但是测试集表现一塌糊涂。

然后在划分训练集的时候选择了按照司机进行划分，这次验证集和测试集的表现比较一致了，但是经过多次调节学习率，验证集 loss 始终在 1 以上。

在参考了 Kaggle 论坛上的一些论点和实现，并且在指导导师的指导下，明白了自己的问题所在，并明确了改进的方向。首先是数据预处理的方式没有跟各预训练模型一致，然后是提取特征向量的方式对于本项目可能不是十分适用，然后就是运用 bagging。

在明确了方向之后，还是踩了很多坑，比如 bagging 的时候上来就对训练集做 k 折划分，然后再对每个划分做训练。这种思路受到了内存的限制，直接爆出了内存错误。之后通过改进，每次划分完成就进行模型训练，然后将训练的结果保存起来，从而解决了内存问题。不正确的预处理也可能导致内存问题。因为在很大的数组中存入整数和浮点数占用内存是完全不一样的，所以在读取的时候进行不要做预处理，把预处理放在划分之后训练之前做能缓解内存占用过多的难题。

刚开始运用 bagging 的时候，发现验证集 loss 在后边的划分越变越小，而在测试集上反而更大了。后来才发现没有正确的使用 bagging，每次划分都应有新建一个模型，而不是利用之前划分训练过的模型。

测试集将近 80000 张图片一次性读到内存并做预处理是一件非常占用内存的事儿。仿照 ImageDataGenerator 的做法，分批读入、预处理、预测，最后将预测结果进行汇总和暂存。数据暂存这个思路对于成功训练和预测非常重要，首先能够用硬盘来代替内存存储数据，其次是数据可以重复利用。

在明确了数据预处理要和各预训练网络保持一致之后，本文首先是参照论文中提到的，减去平均值，然后显示转置然后在减均值，之后再转置。最终的结果还是不错。偶然看 keras 文档的时候发现预训练模型内置了预处理函数，之后又尝试用这些内置的预处理函数做预处理，结果比自己处理半天的表现要好很多。

整个项目编程部分最复杂的部分就是训练集图像按照司机机型 k 折划分了。应该说这里的实现方式很灵活，但是不同实现可能得到的表现不尽相同。本文首先借助 csv 文件将训练集图像按照司机的视角进行归类，给每个图像打上属于某个司机的标签，这样在后边进行 k 折划分的时候，拿到一张图像，如果是验证集的司机就归入验证集，反之归入训练集。

刚开始构建模型做训练时，首先想到了生成器的方式，觉得可以省内存，而且很方便地做预处理。但是在之后的尝试中发现不能跟 bagging 很好的整合，于是放弃了生成器的方式。但是多多少少运用了生成器分批读取的思路。

3.3 完善

本文完善的思路是首先在单个模型下通过调整超参数（主要是学习率）得到一个较为合理的分数，这里还有一个小技巧，就是每种模型选择在两个划分上进行训练，控制单一变量进行对比，这样能够加快调参的速度，避免跑偏。

本文首先在 vgg16 基础上做调优，调优主要调节的是学习率。由于 adam 优越的表现以及不需要手动调节学习率。因此首先在 adam 上进行了两种学习率 1e-5 和 1e-6 的尝试（预处理均为手动减去 ImageNet 均值）。由于在 10 折数据上训练需要花费的时间太久，因此分别在两种学习率情况下进行两种划分下的训练。结果是在 1e-5 时，第一种划分的验证集 loss 为 1.01，而第二种划分的验证集 loss 为 0.27。而在 1e-6 时，第一种划分的验证集 loss 为 0.68，第二种划分的验证集 loss 为 0.45。鉴于结果难分伯仲，所以之后分别在两种学习率下对其余八个划分进行了训练。两种学习率下的验证集 loss 以及预测结果取平均后结果记录如下：

learning rate / loss name	1e-5	1e-6
fold 1 val loss	1.0123	0.6801
fold 2 val loss	0.2796	0.4573
fold 3 val loss	0.2829	0.3531
fold 4 val loss	0.3990	0.5643
fold 5 val loss	0.1935	0.3305
fold 6 val loss	0.3764	0.7287
fold 7 val loss	0.5664	0.5813
fold 8 val loss	0.9008	0.9765
fold 9 val loss	0.1579	0.3289
fold 10 val loss	0.5282	0.5056
test prediction kaggle logloss	0.2829	0.4019

表 1 vgg16 减均值预处理两种学习率的验证集和测试集表现

之后选择了在 1e-5 的学习率下继续做单个 vgg16 的调优，选择了 keras 内置的 preprocess_input，重新进行模型训练，在取得了更好的结果之后，决定放弃自己手工预

处理，而选择内置的预处理。为了进一步提升预测结果，同样的方式分别对 vgg19、resnet50 做 10 折训练取平均，得到各模型在验证集和测试集的表现如下：

model / loss name	vgg16	vgg19	resnet50
fold 1 val loss	0.4335	0.6124	0.5233
fold 2 val loss	0.3494	0.4797	0.4668
fold 3 val loss	03033	0.3472	0.3435
fold 4 val loss	0.3952	0.5223	0.3169
fold 5 val loss	0.2269	0.2291	0.2333
fold 6 val loss	0.3407	0.3150	0.2384
fold 7 val loss	0.5531	0.7193	0.6515
fold 8 val loss	0.8077	0.7731	0.6346
fold 9 val loss	0.3049	0.2354	0.2647
fold 10 val loss	0.3759	0.7362	0.4551

表 2 三种模型运用内置预处理函数的验证集表现

4 结果

4.1 模型的验证和评价

本文经过大量的调试以及尝试，最终各模型以及组合在测试集上的最佳表现如下：

model	test prediction kaggle logloss
vgg16	0.2465
vgg19	0.2625
resnet50	0.3256
vgg16+vgg19	0.2430
vgg16+resnet50	0.2613
vgg16+vgg19+resnet50	0.2505

表 3 三种模型及组合下的测试集表现

本文在 resnet50、vgg19、vgg16 三个预训练模型上进行训练和调优，对比三者表现以及各种组合下的表现，最终选择了表现最好的 vgg16+vgg19 作为最后的模型。这个结果在 kaggle Leader board 上能够排到 127，进入了前 10%（144/1440）。由于对于训练集和测试集的预处理方式保持一致，验证集 loss 表现和测试集 loss 表现也较为接近，因此模型和结果均较为合理，达到了预期的目标。

4.2 Kaggle 表现

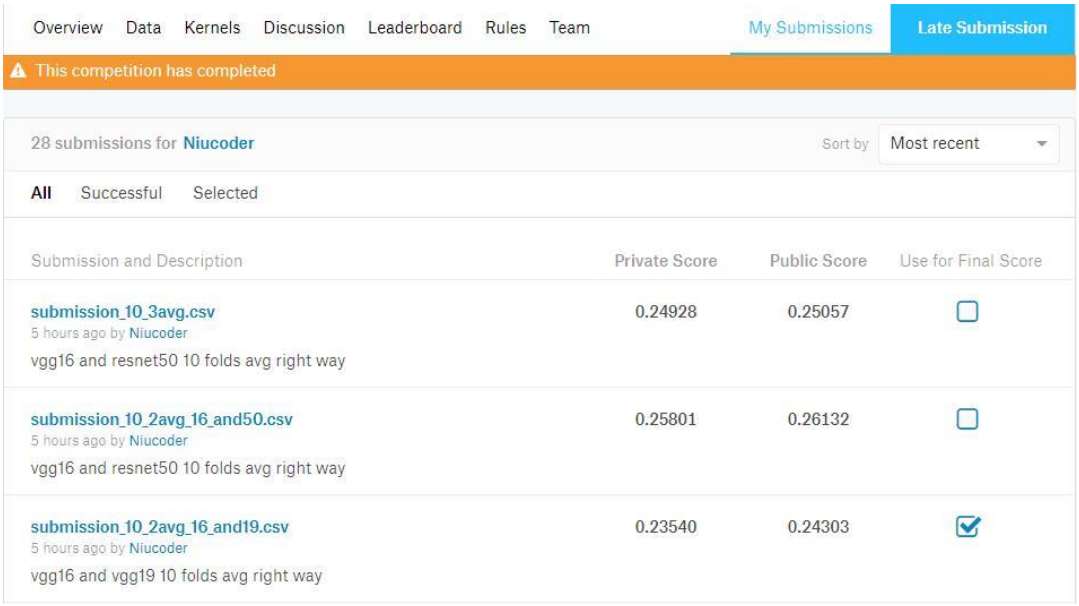

图 4 Kaggle 表现

5 结论与展望

5.1 总结

本文运用三种预训练模型 vgg16、vgg19、resnet50，分别进行模型训练、调优、预测。最终在集成了 vgg16+vgg19 时得到了最佳的表现，达到预期目标的同时，加深了应用 vgg16、vgg19 以及 resnet50 处理大规模图像分类问题的理解。为今后从事深度学习以及图像分类相关研究奠定了坚实的基础。

5.2 思考

本文之所以进展的比较顺利，主要得益于预训练模型的存在以及整个 kaggle 社区和 keras 的帮助。如果自己从头到尾地利用 tensorflow 去构建一各个的卷积层，一个个的池化层，那还不知道要进行多少次尝试才能达到现在的表现。但我发现这种从头构建自己网络的能力也正是我缺乏的，需要在今后的学习中进一步加强。

整个项目做下来，对于利用 keras 构建深度学习模型进行大规模图像分类问题有了更进一步的认识。对于数据预处理，模型调优，bagging，模型集成的应用为今后从事相关的研究打下了坚实的基础。

项目最优挑战性的一个地方就是训练集非常少，是测试集的四分之一，所以应用 bagging 可以说直接解决了这个难题。在编程实现过程中着重注意的一点就是编写内存占

用少的代码，适当地运用数据暂存，恰当的使用生成器，这样能够尽可能地避免内存耗尽。

5.3 展望

尽管最终的结果还算满意，但是距离最优秀的模型还是有较大的差距。根据对本文问题的认识以及参考的一些文章，提出以下进一步改进的方案：

- 提升硬件水平，比如增加 GPU 和内存，利用并行化的代码实现，加快数据处理和训练速度
- 由于训练集比较少，相信进行数据增强后会有更好的表现
- 尝试更多优秀的模型，尝试进行集成
- 尝试在单个模型上锁住一部分层进行调优

参考

[1] https://www.cdc.gov/motorvehiclesafety/distracted_driving/

[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[3] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:14094.1556, 2014.

[4] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[5] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 2818-2826.