机器学习（进阶）毕业项目开题报告

童豪良

2017.07.25

项目背景

本项目源于State Farm公司发布在Kaggle上的竞赛 - https://www.kaggle.com/c/state-farm-distracted-driver-detection，通过车内装置的摄像头来检测司机不安全驾驶的行为，如打电话、手机打字等。识别这些行为，将有利于对用户的驾驶行为进行评估，对不安全的行为予以警示，甚至通过调整保险价格的方式来激励用户安全驾驶。深度学习技术可以对图像进行分类识别，已经在工业上有广泛的应用，可以很好的解决上述问题。

问题描述

在Kaggle竞赛上，State Farm提供了一批已标注的在视频中截取的彩色图片数据，每个照片对应的标注为十种状态中的一种。

State Farm另提供了若干测试集数据，要求我们通过深度学习方法对输入数据进行分析，输出该照片对应十种状态的概率。

数据或输入

State Farm提供了22424张已标注分类的车内摄像头拍摄的司机姿势的jpeg图片，分辨率为640*480，数据分为10类，每类约2000张，分别放在对应文件夹名为train/c0-tain/c10的文件夹中：

• 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

另有79628张测试集数据，放在test文件夹中，需要对这些数据进行分类。

我们将进一步将train文件中的文件，按照2:1的比例分为训练集（train set）和验证集（validation set）

解决方法描述

学生清晰的描述了解决问题的方法。这个解决方法能够配合数据集或者输入解决这个问题。 此外，还要求这个解决方法可以被量化，被衡量以及可重现的。

在此项目中，我们将使用深度学习方法，会搭建一个多层的神经网络来学习训练集中的图片和对应的分类标签的关系，并在验证集上验证其准确率。通过调整参数设置和使用不同神经网络模型，并分别进行多轮训练，选出在验证集及测试集上均能取得较为理想准确率的模型和参数设置。

评估标准

我们将沿用State Farm竞赛中所使用的评分标准：多分类对数损失（multi-class logarithmic loss） 其定义可参见这里，其公式如下：

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

基准模型

我们将在此沿用State Farm的在线竞赛的Leader Board成绩表来和我们的实际结果进行对比。该比赛共有1440支队伍参加比赛。我们希望达到前30%的成绩，对照榜单排名在400名以内。即最后的log loss<0.67。关于该榜单的信息可详见这里：

项目设计

学生总结了一个针对问题解决方案的实施理论流程。探讨了计划采取的策略，对数据需要进行哪些分析，考虑哪些算法。这些流程和探讨符合该问题的特点。我们鼓励把数据简单可视化，加入一些对解释有帮助的伪代码及图表。

搭建环境

深度学习工具

计划使用 OpenCV, tensorflow, Keras 完成该项目

- OpenCV
- tensorflow
- Keras

AWS

由于此项目要求的计算量较大，计划使用亚马逊 p2.xlarge 云服务器来完成该项目，利用杨培文在弗吉尼亚北部已经配置好的 AMI 。参考：在aws上配置深度学习主机

数据预处理

由于Keras对数据的存放有特定需求，我们会对数据进行预处理。将State Farm提供的训练集数据分为训练集和验证集。其中，训练集分为c0-c9，一共10个文件夹，每个文件夹对应一个类别的驾驶行为，每个文件夹选取前1200张照片。验证同样分为c0-c9，一共10个文件夹，每个文件夹对应一个类别的驾驶行为，每个文件夹从初始的训练集原始数据中选取后600张照片。原测试集保留不变。

数据提升

由于原始的训练数据不多，对数据集进行切分后数量更少。故我们会采用keras.preprocessing.image.ImageGenerator来对数据进行一系列的随机变换，如旋转，在水平和垂直方向进行平移，剪切或放大。

训练神经网络

本项目参考了“使用Keras面向小数据集进行图像分类”的方法和代码。原文中解决的是2分类的问题，而在我们的项目中需要解决的是10分类的问题，故会做出相应修改。原文中只使用了预训练的VGG16模型，我们会尝试使用Googlenet，Resnet等多种不同的模型，并尝试不同的数据提升、dropout、权重衰减设置，以获得更高的准确率和更低的对数损失。

在小数据集上直接训练神经网络

我们将使用一个3层（卷积+Relu+maxpooling）+ 2层全连接的简单模型来进行初步的尝试。我们可以借此来测试我们对数据预处理的操作是否正确。我们也可以借此获得一个参考的准确率和损失函数，并对输出结果可视化。

利用预训练网络的bottleneck（瓶颈）特征

更进一步的方法是利用vgg-16/GoogleNet/ResNet等在ImageNet这样的大规模数据集上已经预先训练好的网络模型。我们将仅保留预训练模型的卷积层，将其在我们的训练集和验证集上运行并记录其输出（即“bottleneck feature”，网络在全连接之前的最后一层激活的feature map）。然后我们基于这个bottleneck数据对全连接网络进行训练，这样可以大幅节省训练时间。同时，我们会记录下来全连接层的权重设置，这会在下一步的训练中用到，因为如果使用完全随机的初始权重设置会破坏卷积层预训练的权重。

fine-tune预训练网络的高层

为了进一步提高准确率，我们可以对预训练的网络模型进行finetune。由于整个高网络具有巨大的熵容量，很容易过拟合，且调整更多的层需要更多的计算消耗，故我们只选取finetune最后一个卷积块。我们会载入预训练模型的权重，并锁定其最后一个卷积块之外的层，在预训练模型之上加载我们的全连接层和在上一阶段已记载的权重，然后开始我们的训练。在训练的过程中，我们会尝试对超参数进行调整，如使用不同的dropout rate，以及尝试数据提升的影响。

训练结果可视化输出

在训练完成后，我们将随机在测试集中抽取若干张图片，显示这些图片，并利用训练好的模型预测这些图片所对应的分类和预测的概率，并以图表形式展现。

训练结果评估

我们将在完整的测试集上运行模型，并计算其log loss，最后和Kaggle的Leader Board进行得分的比对。我们计划达到前400名的水准