**机器学习纳米学位**

##毕业项目 黄煜祺 [qier\_241078450@qq.com](mailto:qier_241078450@qq.com) 2017年11月1日

**运行环境**

Python3.6.2+keras2.0.8+jupyter1.0.0+Windows10

**I. 问题的定义**

**项目概述**

随着时代的发展，随之而来的交通事故发生率也随之上升，而其中很大一部分是由驾驶者的不正当驾驶行为所导致，而无人驾驶技术的出现，是可以起到由人为导致的交通事故率的，其作为一项跨学科的存在，其中作为行车安全的因素，路面探测和动作选择占了十分重要的地位。而当车辆的前置摄像头把车辆行驶中的前方图像传回控制终端，就需要控制终端作出合适的判断，控制方向盘的转动方向，以保证行车安全，本项目是通过Tesla的行驶数据（来源：<https://github.com/nd009/capstone/tree/master/deep_tesla>），对其方向进行模拟和预测，在研究的同时本人将保证不在商业行为上使用这些数据集。

**问题陈述**

在Nvidia的end-to-end模型论文中提出，它希望该设计一个模型，能和以往需要分感知、检测、决策控制等分开的无人驾驶模型不同，仅以神经网络实现，通过摄像头看到的图像的分析和预测来完成。在实现时, Nvidia使用卷积神经网络（CNN）将从前向摄像机得到的原始图像映射成自动驾驶汽车的驾驶命令，这种端到端的模型意味着只需要极少的来自人为的训练数据，该模型就能自己学会驾驶，且不管有没有车道标志，也不管是在地方道路上还是在高速公路上，同时系统更能在停车场或未铺好的道路等没有明显视觉指引的地方运行。为此，我计划根据end-to-end模型的实现思路，以公开的Tesla数据集对车辆的前置相机所拍摄的路况图像，模拟端对端的模型，实现对车辆转向角度的预测，以使得车辆能在特定情况下做出正确的方向选择，最终提高行车安全。

**评价指标**

我计划使用以下参数值对模型进行评估：

1. 向前/向后反应耗时值：forward/back pass ms，通过从每次车辆遇到前进道路转向和前方车辆状况时到实际作出动作的时间间隔计算得出，计算公式为：

TIM截图20171115161023.png

2） 平均误差值：Mean absolute error，实际和预测间的差值，根据车辆实际作出的方向盘转向值与数据集中对应帧的转向值的误差值取绝对值的总平均值算出，计算公式为：

TIM截图20171115162311.png

3）均方误差值：Mean squared error, 参数估计值与参数真值之差平方的期望值，由于MSE可比较直接地表达出模型根据车辆实际作出的方向盘转向值与数据集中对应帧的转向值的相差程度，故将之选为评估指标，计算公式为：

TIM截图20171115162318.png

**II. 分析**

**数据的探索和可视化**

整个数据集中，共有10个前置摄像视频文件及10个转向数据文件，示例分别为：

1. 车辆前置相机所拍摄的路况图像视频（xxx\_front.MKV），示例为：

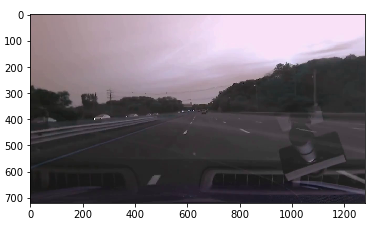


1. 车辆前置相机所拍摄的路况图像视频所得到的方向盘转向数据文件（XXX\_steering.csv），其中包含3列数据，示例数据为：

|  |  |  |
| --- | --- | --- |
| **ts\_micro** | **frame\_index** | **wheel** |
| 1464305394391807 | 0 | -0.5 |
| 1464305394425141 | 1 | -0.5 |
| 1464305394458474 | 2 | -0.5 |

ts\_micro是时间戳，frame\_index是帧编号，wheel是转向角度（以水平方向为基准，+为顺时针，-为逆时针

由于10个视频文件以mkv格式提供，而视频文件是通过具体的帧数对应转向数据文件，因此将使用OpenCV完成，具体代码及运行效果为：



可见，每帧文件大小为1280X720，颜色深度为3。

而对应的转向数据文件，使用pandas检查其数据格式，代码和效果如下图：



可见转向数据记录是按帧顺序记录的。

而由于共有十个,经过合并数据集后，10个steering数据集的wheel的值分布的代码如下图，运行效果如下图：



可见每个steering数据集有大概1500~3900行数据，总共有27000行数据，是足够训练一个普通的CNN模型的。但是，很显然由于多数帧的转向数据是直行决策远多于转弯决策，将有可能造成训练出来的模型更偏向直行，因此如何避免这种情况出现也是本项目的研究方向之一

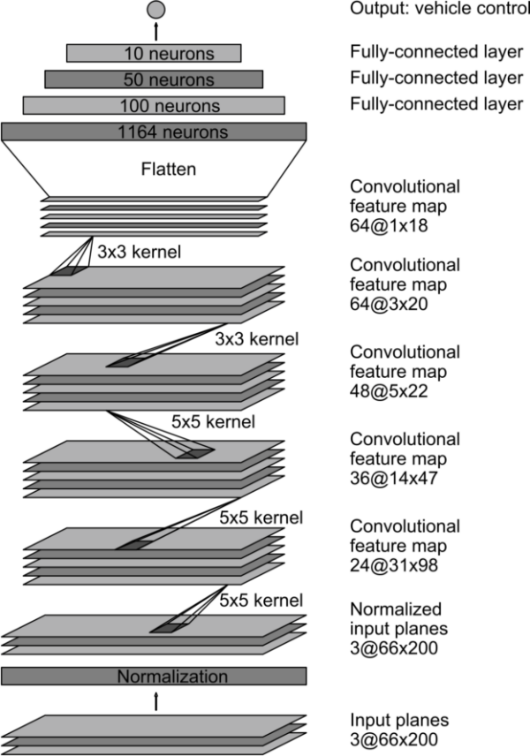
**算法和技术**

在本项目中，视频文件的每一帧作为数据的输入，而每一帧对应的方向盘转向作为对应的决策，同时，由于平均一段视频中有2000多帧，每一帧为1280X720X3的图片，且最后的决策为角度转向。

卷积神经网络（下称CNN）为一种前馈神经网络，其结构主要包括两层，其一为特征提取层，每个神经元的输入与前一层的局部接受域相连，并提取该局部的特征；其二是特征映射层，网络的每个计算层由多个特征映射组成，每个特征映射是一个平面，平面上所有神经元的权值相等。相对于其他学习模型，CNN避免了对图像的复杂前期预处理，可以直接输入原始图像，最后输出其特征。而CNN也由于其最大特点就是局部感知和权值共享，其相比可以减少所要训练的参数，减少计算复杂度。在CNN中，将以矩阵作为图片像素输入,使用一个指定大小二维滤波器kernel作为卷积核，然后对每个像素点平移指定步长strides，用以对图像作为特征提取。而作为输出前的激励函数，我之所以想使用ELU函数，是因为ELU融合了sigmoid和ReLU，使得ELU能够缓解梯度消失，能够对输入变化或噪声更具鲁棒性。同时，最后我使用了MAE和MSE作为评估指标，之所以使用MAE和MSE，是因为据对数据集的观察，值存在正和负，因此将取绝对值及平方值以防止数据被抵消，而相对于MAE，MSE能更正则化error，更好地反映预测值误差的实际情况，同时，MSE的值越小，说明模型描具有更好的精确度。而使用adam作为优化函数，是因为adam的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。

**基准模型**

在Nvidia的无人驾驶的一篇论文（[论文正文](https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf)）中，提出了一个无人驾驶可用的CNN基准模型，结构如下图：



此模型共有一个输入层，一个标准化处理层，五个卷积层，三个全连接层，一个输出，本项目最终的模型结构将与该模型进行比较。

**III. 方法**

**数据预处理**

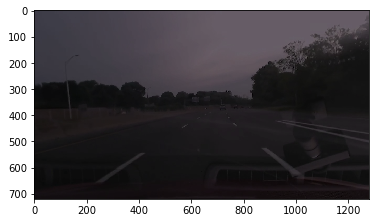
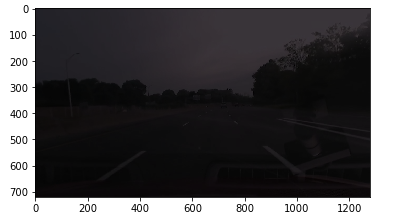
在上一部分中，已对数据有了初步的了解，而在除去第10个视频文件和数据文件后，剩下大约25000帧图片及对应的转向数据可供模型训练，且也已知大部分转向决策为直行，显然如果模型按照这些训练集训练出来后，不足以应对现实世界中千变万化的行驶状况，即训练集的数据过少，将造成模型的训练不够充分。因此，将使用数据增强技术（本文采用Regularization相关的方法），改变已知帧数据进行颜色变换，平移变换，缩放变换等处理，以增强训练效果。

由于帧有25000多个，下面我将在使用数据集中的示例图片（路径：.\deep\_tesla\images\img\frame\_1173.jpg）表述处理流程：



**处理一：颜色变换**

更改图片颜色色调，可模拟同模型在同一路况下不同的亮度所作出的不同的转向处理。可使用OpenCV中的cvtColor将RGB色域转化为HSV色域，将其中第三通道乘以一个随机的常数，再将其转回RGB就可以达到更改图像亮度的效果，效果如下：



**处理二：平移变换**

通过平移变换，可使得同一图片的前方路况被改变，可使得模型在同一环境下可学习到更多的决策。可使用OpenCV中的warpAffine并设置随机偏移量实现，效果如下：



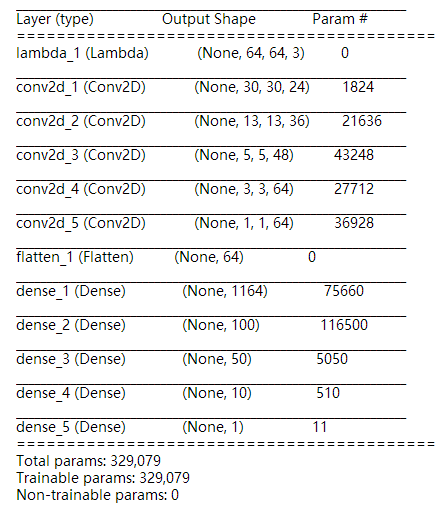
**处理三：缩放变换**

除了平移变换，对图片进行缩放变换也可达到模型在同一环境下，不同路况下学习不同的决策，可使用OpenCV中的resize并设置随机压缩大小实现，代码如下：



**模型设计**

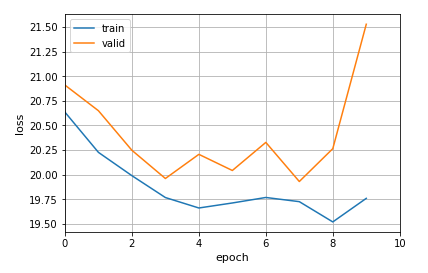
开始执行前，首先需设计合适的模型，在这里将选择基于Nvidia的end-to-end基准模型，以keras作为较为清晰易懂的CNN结构包作为主要工具以展示。结构中，输入将先通过标准化层，降低其离散区间（每个维度除以255，以keras.layers.Lambda实现），以加速在不同GPU上的处理时间；然后，将通过三个5X5的kernel和2x2的stride，两个3X3的kernel和无stride的卷积层（以keras.layers.Convolution2D）；接着经过flatten（以keras.layers.core. Flatten实现）后，将通过3个全链接层（以keras.layers.Dense实现），将输出的提前特征，用于训练其转向预测，模型结构如下：



**执行过程**

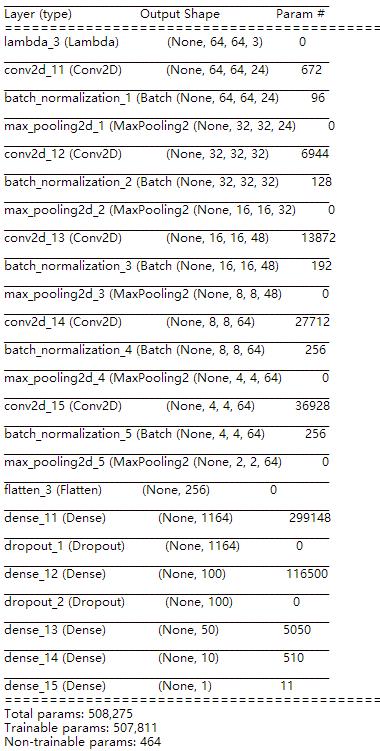
完成模型建立后，进行预测前，首先需对模型进行训练，此处将使用编号1~9的视频和转向文件合并作为训练集和测试集，具体思路是对每个视频文件的每一帧生成经过预处理和未经过预处理的帧文件，同时生成2份对应的转向数据，以确保为经过预处理的帧和经过预处理的帧具有相同的转向决策，并使用pickle保存至硬盘，方便二次读取，

然后便进行分割训练集和测试集，比例为8：2，完成后，便对模型进行训练，训练过程及结果为：

`

**完善**

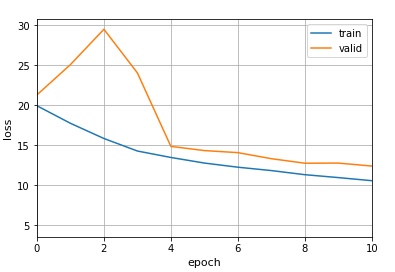
在上一步骤中，根据首次预测过程及可视化结果，可见在预测过程中，虽然训练loss整体有向下降低的趋势，但下降结果并不比初始降低多少；而验证loss在下降到20左右时，出现了反复的情况，针对此情况，在模型中，激活函数修改成elu，padding改成same，同时加入池化层(池化前使用BatchNormalization规范化池化输入)及降低adam的学习率，新的模型结构为：



在训练时，先以数据集1训练模型，模型修改点及MAE，MSE，MPS如下表所示

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改点 | **MAE** | **MSE** | **MPS** | **epochs** |
| 基准模型 | 2.56 | 10.39 | 0.0013 | 10 |
| Dropout(0.25),Dropout(0.25) | 2.58 | 10.48 | 0.0018 | 10 |
| All activation change to 'elu',  remove activation’tanh’ | 2.51 | 9.89 | 0.0019 | 10 |
| MaxPooling2D after conv  All padding change to’same’ | 2.48 | 9.57 | 0.0029 | 10 |
| Add BatchNormalization after Lambda | 2.41 | 9.07 | 0.0044 | 10 |
| Dropout(0.25),Dropout(0.5) | 2.39 | 9.14 | 0.0048 | 10 |
| Change epochs to 30 | 2.09 | 7.57 | 0.0052 | 30 |
| Change epochs to 70 | 1.07 | 3.60 | 0.0053 | 70 |
| Change epochs to 100 | 0.94 | 3.46 | 0.0052 | 100 |

接着便可对数据集1~9进行训练，将训练迭代次数调整为35，训练过程及结果为：

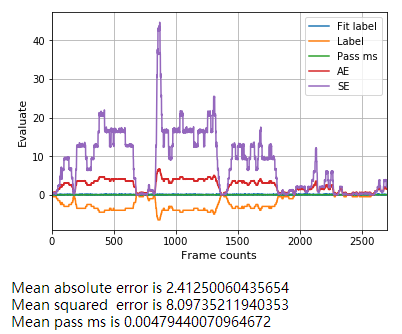


可见相同数据集在此模型下，训练loss和验证loss相对原来的nvdia模型，表现有较大改善。完成模型训练后，接着可将模型及参数保存，生成model.json和model.h5，以供运行最终epoch10数据集使用。

**IV. 结果与项目结论**

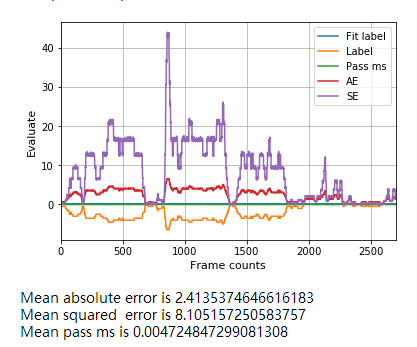
**模型的评价与验证**

生成好模型文件后，可运行run模块，在epoch10数据集上生成结果，同时输出评估目标forward/back pass ms，MAE,MSE，结果图如下：



从图中结果可看出，预测的转向值只是浮动于0左右，相对于人工驾驶的转向值，模型的预测的AE和SE较高，最终的MAE为2.34，MSE为7.64。而对于判断时间pass ms指标，其响应时间十分短，几乎为0，所以这个指标的值是较为满意的。

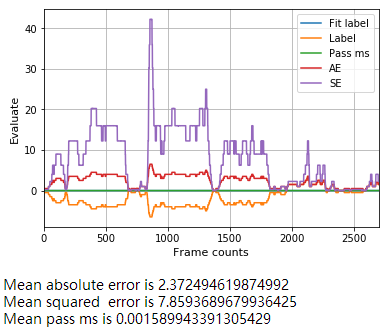
而假设在epoch1的第500，1000，1500，2000，2500帧，进行2次预处理，模拟当模型遇到输入极端数据时的效果，其效果如下图：



可见，在进行2次预处理的帧的表现，与只进行一次预处理的表现，模型的预测结果均未发现十分明显，且MAE和MSE值变化较低。这说明，此模型在遇到一定程度的输入改变时，并不会出现极大的影响结果。因此，模型得出的结果是健壮且可信的。

**合理性分析**

为了能与nvidia的模型有比较有说服性的对比，我使用基准模型在epoch10数据集上运行了预测，其结果如下：



所以，根据上述三次对数据集10的预测后，可得出以下统计表格：

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MAE** | **MSE** | **MPS** |
| 基准模型 | 2.37 | 7.86 | 0.0016 |
| 修改模型 | 2.41 | 8.09 | 0.0048 |
| 加入异常处理的修改模型 | 2.41 | 8.11 | 0.0047 |

根据结果图可见，基准模型的预测结果，AE和SE浮动更加平滑，MAE和MSE值也比修改模型高。而也可以从统计表格中看出这说明，在基准模型的预测结果中，更多的方向盘转向预测更偏向于0，也就是其更偏向于使车辆往前行，在当今路况复杂的行驶环境中，这种决策是十分不利于安全行驶的，基于此结果，我认为相对于基准模型，经过我修改的模型更偏向加适合模拟行驶。

**对项目的思考**

在本项目中，我在拿到数据集后，首先对其内容进行了预览，接着确立了研究目标和评估指标pass ms mean absolute ，在研究了数据集的整体情况后，确立了预处理方案，然后将数据集读出并缓存，接着根据nvidia的基准模型和自己的模型分别预测了数据集，最后获得结果指标。在整个过程中，我花了大概2天时间确立并测试出预处理方案，然后才开始训练基准模型，在完成基准模型后，发现效果一般，再根据基准模型修改出最后的模型。在整个过程中，我把一些图片和数据集处理的方法封装成一个单独的py文件方便调用，使得代码更简洁。而针对图片的预处理和测试出合适的修改模型，是对于我来说耗时比较久的地方。

对于最终模型，虽然比基准模型表现较好,达到了我一开始比基准模型较好的期。但在通用环境下解决其他问题，需要进一步的优化，以及更多的路况来供模型学习。

**需要作出的改进**

虽然到最后得到了比基准模型表现更好的模型，但该模型仍有以下不足的地方：

1. 相同硬件环境下，训练时长比基准模型大幅增加
2. 模型预测的转向值与真实值误差较大
3. 模型的loss值仍然过高

### 为此,在后续的修改中,我将考虑使用其他基准模型或技术为根据,在此基础上探索出适合该数据集的模型。比如，根据大规模图像识别的深度卷积神经网络所提出的迁移学习模型VGG16 net及百度针对中国路况所提出的[Road Hackers](https://www.baidu.com/link?url=OeRLfBqkv3N03zvDSNQtIuEoD9JjMkFiheUbiyCWFKcee6PIEKjnkNwzOCRRAW5e&wd=&eqid=9e36c41700007305000000065a001631)模型便是我想继续探索的基准模型方向。在选择基准模型或技术的同时，我也考虑了如果在我修改后的模型的基础上，是否能够对图片预处理方面做更深层次的优化，比如调转方向等，或使用其他类型的优化器，预计也能完善出更好的模型及其结果。

**V. 参考资料**

### 1）毕业项目主页（[地址](https://github.com/nd009/capstone/tree/master/deep_tesla)）

### 2）Nvidia end to end 模型论文（[地址](https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf)）

### 3）Keras文档（[地址](https://keras.io/)）

### 4）知乎-让特斯拉（Tesla）学习人类的驾驶技术（[地址](https://zhuanlan.zhihu.com/p/25334400)）

### 5）阿里云GPU（[地址](https://yq.aliyun.com/topic/105)）

### 6）VGG 论文笔记（[地址](http://www.jianshu.com/p/9c6d90e4f20e)）

7）百度Road Hacker（[地址](http://roadhackers.baidu.com/)）

8) Github – gwwang（[地址](https://github.com/gwwang16/MLND-Capstone-Project-DeepTesla)）

9) Github - commaai （[地址](https://github.com/commaai/research/blob/master/train_steering_model.py)）

10) CS231n Convolutional Neural Networks for Visual Recognition（[地址](http://cs231n.github.io/neural-networks-1/)）

11)Github – lexfridman （[地址](https://github.com/lexfridman/deeptesla)）