机器学习纳米学位

DeepTesla

2017 年 4 月 30 日

1.定义问题

项目背景

这个项目基于课程 MIT 6.S094：自动驾驶汽车深度学习。【1】人工智能（Artificial Intelligence），英文缩写为 AI，是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。过去几十年来，人工智能在广泛领域得到应用，研究工作发展迅猛，包括车辆控制、机器人、感知、人机交互、机器学习等。

在追求效率的现代社会，交通工具成为了我们生活中非常重要的一部分。然而，目前的交通工具还无法达到人们所期待的安全、高效。同时，随着车辆的不断增加，交通状况越来越差。

在这种背景下，自动驾驶得到广大群众的期待。本项目选择自动驾驶众多研究领域中的 steering control 作为研究课题，选择 end-to-end learning 【2】的学习模型作为研究方向。

问题描述

自动驾驶控制系统是一个非常庞大、复杂的系统。

要实现自动驾驶，需要使用各种技术来探测周围环境，如雷达，激光，GPS，测距和计算机视觉。【3】

同时，控制系统需要对探测到的信息进行正确的处理，从而能即时地针对时刻变化的环境作出正确的决策。

控制系统包括： Adaptive Cruise Control、Sensor Fusion、Localization、Path Planning、Lane Keeping Assist Systems 等模块

以下为自动驾驶所使用的基础设施及相关技术的框架图。【4】

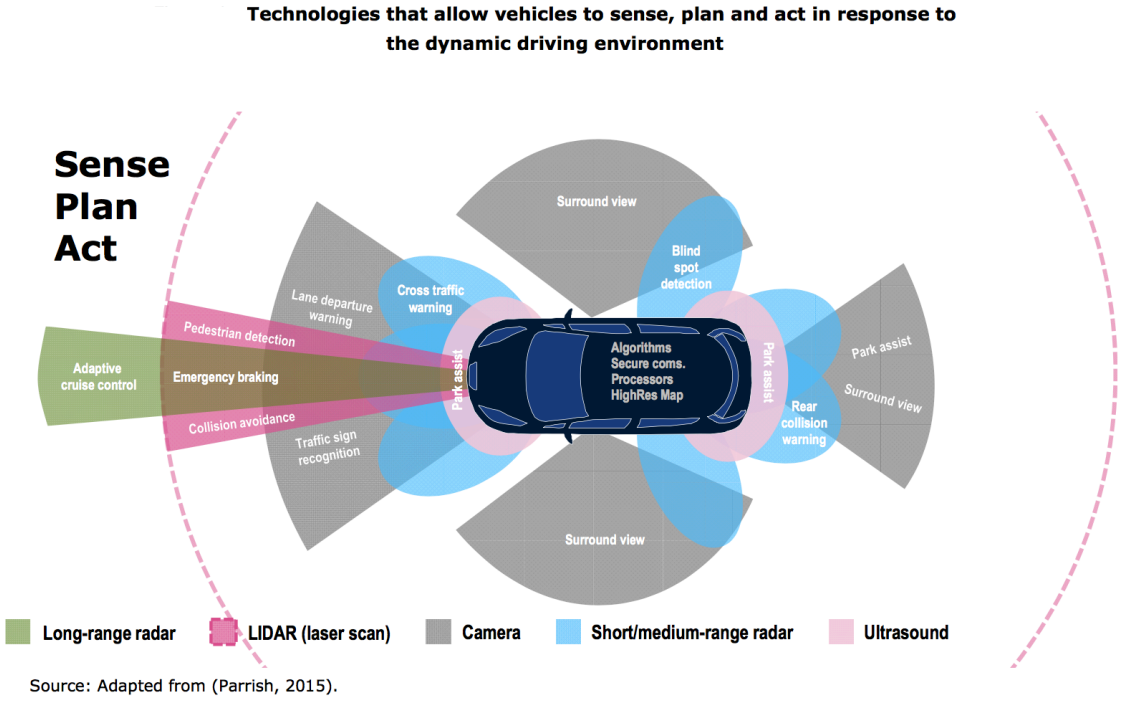

Figure 1

本项目所选的 steering control 仅仅是 LKAS(Lane Keeping Assist Systems)的一部分，除此之外车辆若想保持平稳行驶还需要结合 acc(Adaptive Cruise Control)模块来实现自适应巡航。steering control 只是复杂系统的一小部分，即使利用深度学习结合计算机视觉的手段，也是依然要和其他模块进行协作的。

End-to-End Deep Learning，【2】在这个应用中，我们通过汽车的前置摄象机来捕获车辆的行驶环境，通过车辆控制器同步记录车辆行驶的方向。利用卷积神经网络将摄像机捕获的视频的每帧图片和每帧图片图片对应的 Steering Angle 对应起来，实现模型从原始像素映射到 Steering Angle 的学习，也就是 end-to-end 学习。

end-to-end 是指输入的是原始数据，输出的是最后结果，提取 feature 以及 feature 与最终结果的映射这个过程交给模型自己完成。

end to end 的好处：通过缩减人工预处理和后续处理，尽可能使模型从原始输入到最终输出，给模型更多可以根据数据自动调节的空间，增加模型的整体契合度。

评估指标

采用 MSE 作为评估函数,公式如下：【5】

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

其中 n 是训练样本的的数量，$\hat{Y}_i$是系统预测 wheel，Y_i是司机实际转向。评估指标为误差平方求和，然后求其平均值。

为了对模型的学习效果进行检测，设置了训练组和测试组。训练结束后，利用已经训练好的模型来对测试组进行测试。具体包括两个方面：

- 将我们预测的结果与测试组的 label 直接进行比较，平均误差控制在 2.5 以内。
- 将模型预测的结果与对应每帧图片重新生成视频，并与司机驾驶的方向进行比较。车辆驾驶过程中连续出现方向错误的时间不超过 5 秒且不多于 3 次。

2. 分析

数据的探索

本项目数据来自 DeepTesla。主要包括 tesla 在两种不同驾驶模式（human driving 和 autopilot）下的前置相机录制的视频和车辆的转向控制信号。数据格式如下：

- 前置相机记录的视频：mkv 格式
- 行驶过程中的控制信号：csv 格式

本项目的原始数据为 10 段 mkv 视频和与每段视频的 steering angle（csv 文件）。csv 文件记录了视频中每帧图片对应的方向转角。csv 文件总共有 2700*10=27000 条记录。

数据探索可视化

1. 数据分布

steering angle 的分布情况见下图。从图表可以看出，绝大多数情况，steering angle 出现在-5 至 5 之间。最大拐角大约为 15 度，而且出现的频率非常低。根据数据分布，大致可以推断，转角在-5 至 5 之间的训练样本比较充足，而其他转角的训练样本相对不足。

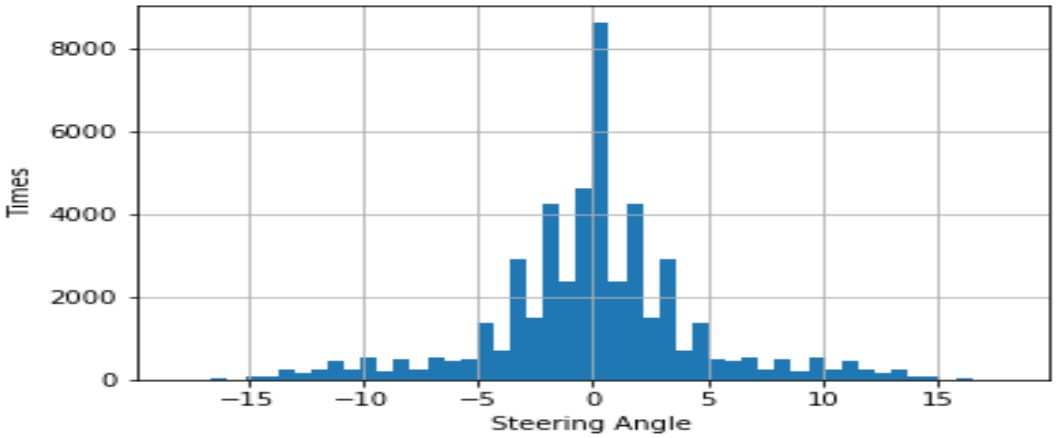

Figure 2

2. 数据变化幅度

从所有训练视频中随机抽取一段，方向转角与时间的对应关系如下。从图表可以看出，转角的变化幅度总体看来不大，但转角的变化并不是连续性的。

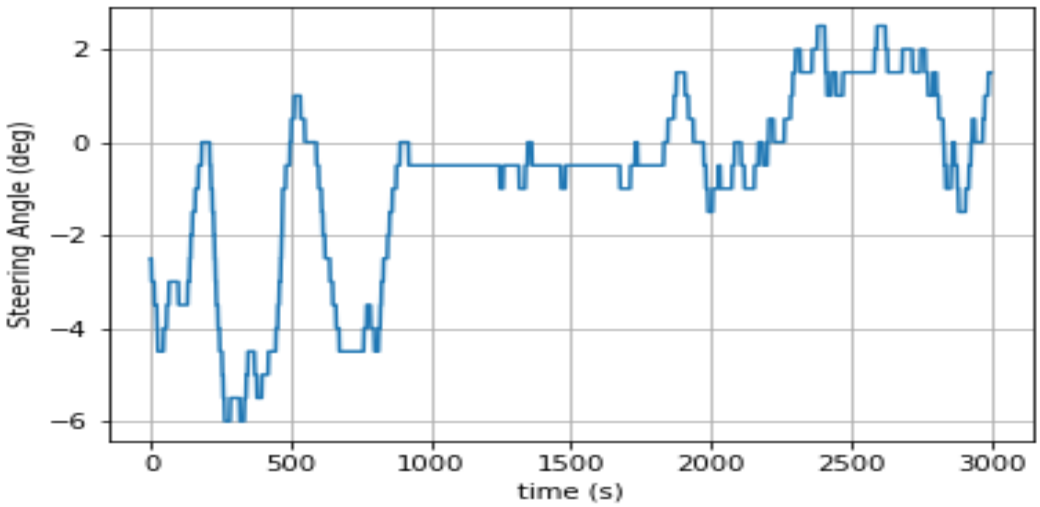

Figure 3

算法和技术

为了实现端到端的学习，我们需要建立原始数据（像素）与最终的方向转角之间的联系。这种联系可以构建神经网络来建立，而对原始数据的特征提取，卷积神经网络表现尤为出色。

卷积神经网络【6】

卷积神经网络是人工神经网络的一种，它的权值共享网络结构使之更类似于生物神经网络，降低了网络模型的复杂度，减少了权值的数量。 卷积网络是为识别二维形状而特殊设计的一个多层感知器，这种网络结构对平移、比例缩放、倾斜或者共他形式的变形具有高度不变性。

卷积神经网络是一个多层的神经网络，其基本运算单元包括：卷积运算、池化运算、全连接运算和识别运算。

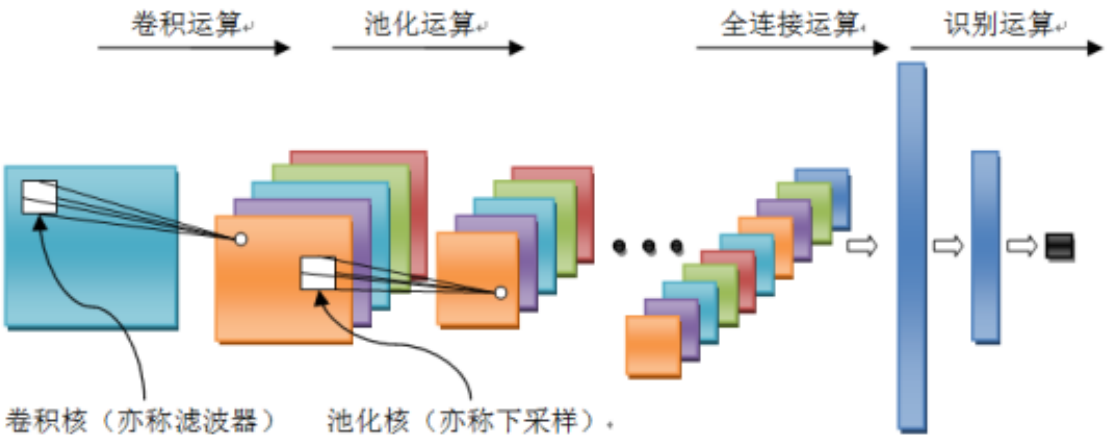

Figure 4

卷积运算：前一层的特征图与一个可学习的卷积核进行卷积运算，卷积的结果经过激活函数后的输出形成这一层的神经元，从而构成该层特征图，也称特征提取层，每个神经元的输入与前一层的局部感受野相连接，并提取该局部的特征，一旦该局部特征被提取，它与其它特征之间的位置关系就被确定。

池化运算：它把输入信号分割成不重叠的区域，对于每个区域通过池化（下采样）运算来降低网络的空间分辨率，比如最大值池化是选择区域内的最大值，均值池化是计算区域内的平均值。通过该运算来消除信号的偏移和扭曲。

全连接运算：输入信号经过多次卷积核池化运算后，输出为多组信号，经过全连接运算，将多组信号依次组合为一组信号。

识别运算：上述运算过程为特征学习运算，需在上述运算基础上根据业务需求（分类或回归问题）增加一层网络用于分类或回归计算。

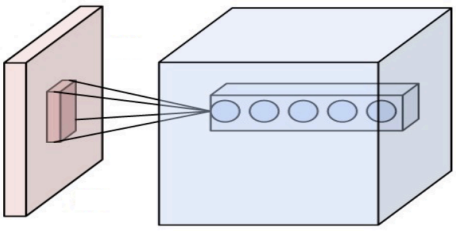

Figure 5

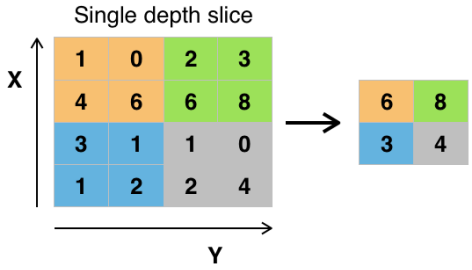

Figure 6

基准模型

在本项目中，使用 NVIDIA 模型作为基准。具体见下图：【2】

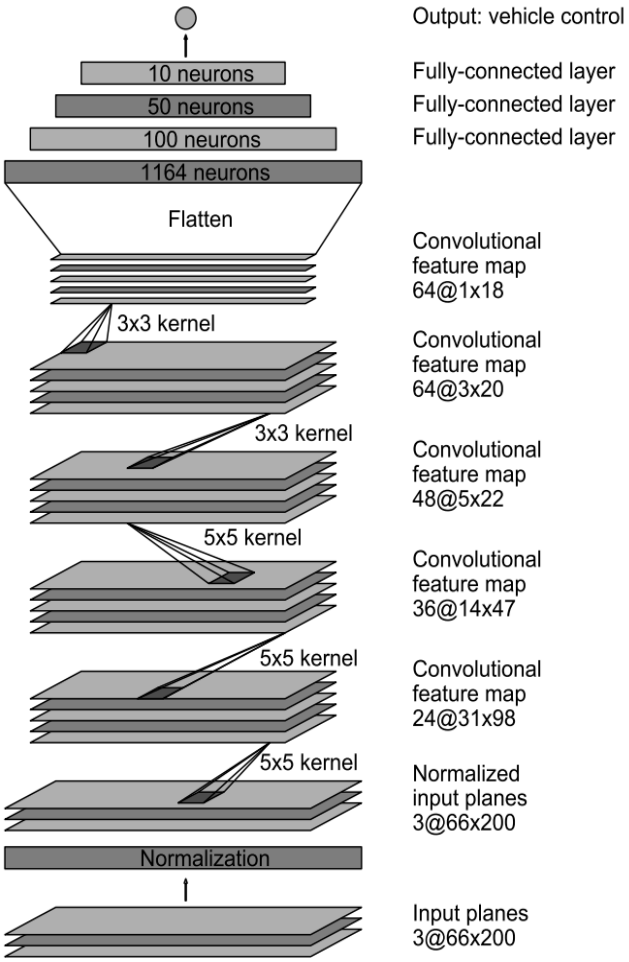

Figure 7

模型由 9 层组成，包括标准化层、5 个卷积层和 3 个全连接层。

标准层对输入数据进行 normalize。输入尺寸为（200,66,3），宽度为 200，高度为 66，3 通道（红 - 绿 - 蓝）。

卷积层被设计为执行特征提取，并且结合经验改变层配置的一系列实验来选择卷积层。 前三个卷积层中采用 2×2 步长的 5×5 卷积核，后两个卷积层采用 1×1 步长的 3×3 卷积核。5 个卷积层之后是 3 个完全连接层，作为对最终结果的输出函数。但在端到端的学习中，特征提取层和结果输出层之间也有着紧密的联系。

3.方法

数据预处理

数据预处理代码见 preprocess.py 文件及 data_pickle.ipynb 文件。处理包括：

- 利用 ffmpeg 将 mkv 格式视频转换为 mp4 格式视频。
- 利用 opencv 抓取视频中的每段视频，然后去掉图片上部的三分之一（天空），以及下部的 150 像素（车头）。
- 按照图片原有的高宽比等量去掉图片两边的部分，然后将图片缩放至 66*200*3 像素大小。
- 将第 1-9 段视频抓取的每帧图片作为训练数据，将第 10 段视频抓取的每帧图片作为测试数据。
- 将训练数据中的每帧图片沿 y 轴方向进行翻转并增加至原训练数据，从而加倍训练数据，同时对 csv 所对应的角度进行相应的变换。
- 将处理后训练数据中每帧图片转换成 YUV 模式，从而在原有 RGB 模式的基础上增加 YUV 模式训练数据。
- 上述处理完成后，将数据以 RGB 模式和 YUV 模式分别存至 pickle 文件，以备调用。

Figure 8

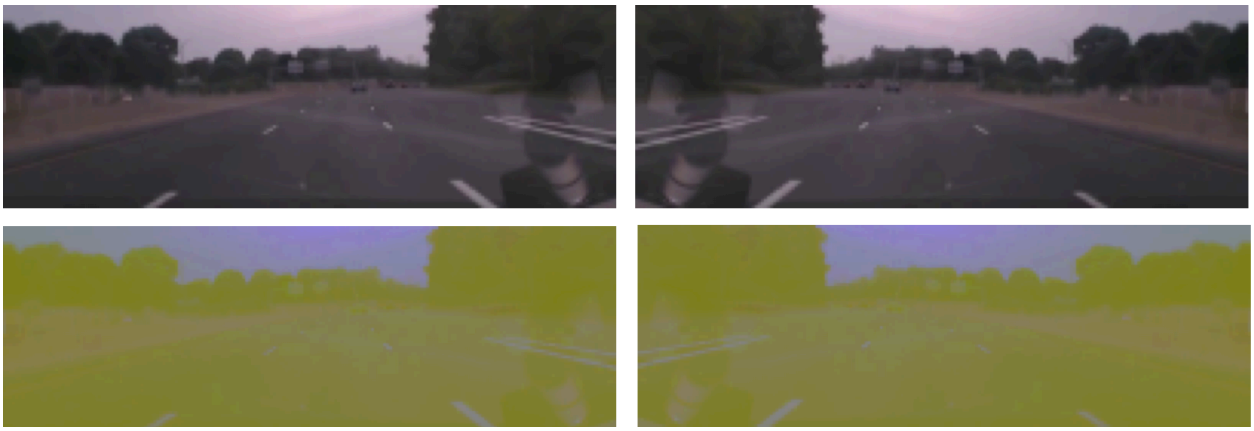

Figure 9

数据预处理前后对照分别见 Figure8、Figure9。

执行过程

- **基准模型**

开始时，epoch 设置为 10，结果 train_loss 和 val_loss 都非常小，然而，test_loss 很大，这说明训练出现了过拟合，因此将 epoch 重置为 8，测试结果变好。分别训练 RGB 和 YUV 两种模式的样本，train_loss 和 val_loss 随 epoch 的变化见 Figure7。

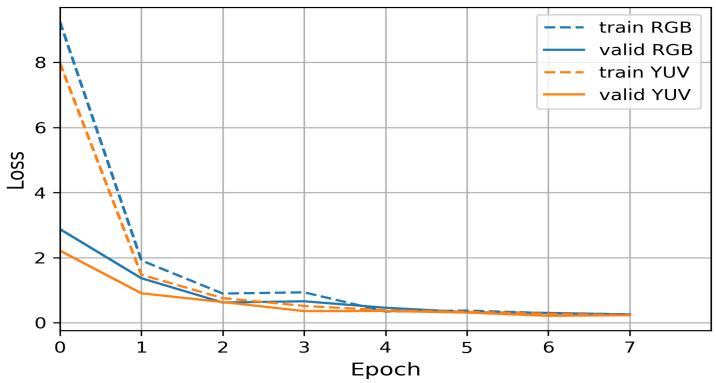

Figure 10

通过上图可以看出，train_loss 和 val_loss 随着 epoch 的增大不断减小，并且收敛性很好，但 test_loss=3.19（RGB）和 test_loss=3.45（YUV），远远高于 train_loss 和 val_loss，说明模型过度拟合。

- **改良模型**

根据前面对基准模型表现的分析，基准模型出现了过拟合，在此改良模型中加入 dropout 层。开始时，dropout 分别设置为 0.4，0.4，0.5，0.5，训练 12 次，train_loss 和 val_loss 还是比基准模型高，所以最后将 dropout 重置为 0.3，0.3，0.4，0.4。epoch 设置为 10，经过 10 轮训练，表现如下：

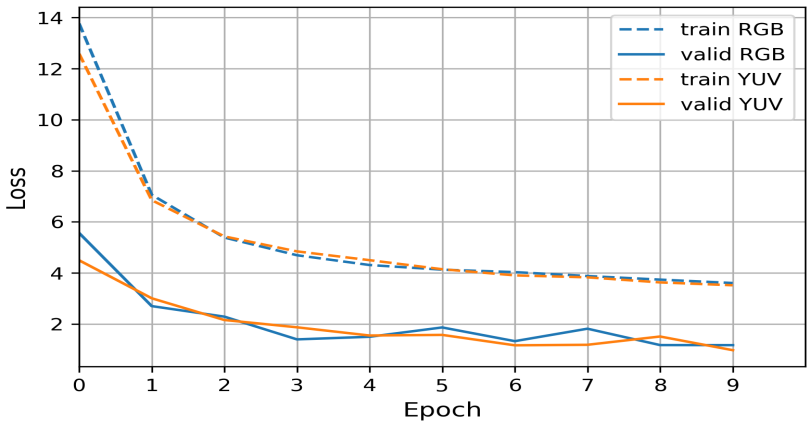

Figure 11

通过上图，我们可以发现改良模型一定程度地解决了过拟合，但经过更多轮的训练，模型的表现还是会变差。

除了改善模型的过拟合问题外，还分别尝试过，将 kernel_initializer 设置为 'TruncatedNormal'、'glorot_normal'，但结果都不如'he_normal'。Activation 设置为'elu'，是因为 elu 在零处的导数更为平滑，因此对于预测的连续值，预期会更好。【7】

最终结果，test_loss=2.84（RGB）和 test_loss=2.57（YUV），比基准模型有所改善。

- **最终模型**

为了进一步解决过拟合的问题，在其中两个卷积层后接入 BatchNormalization 层。在实际操作的过程中作了很多尝试。由于 BatchNormalization 可以加速收敛、 可以控制过拟合，可以少用或不用 Dropout 和正则，因此 BatchNormalization 所添加的位置是和 dropout 一起调整，经过对模型的表现决定的。

通过 8 轮的训练，模型表现如下：

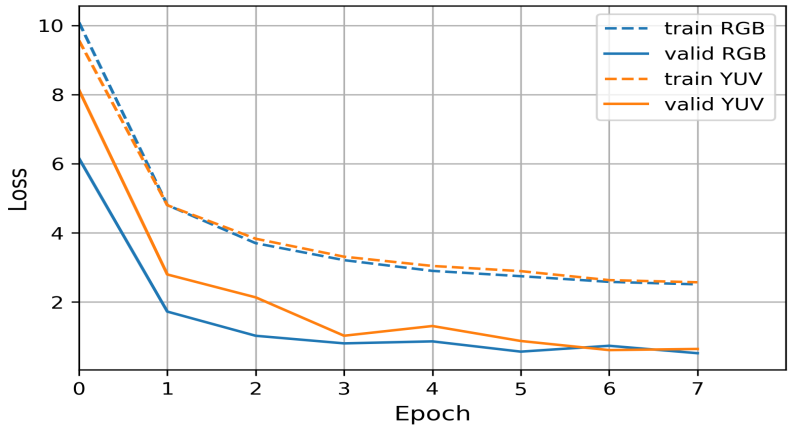

Figure 12

通过这次的调整，test_loss 分别降到了 2.45（RGB）和 2.72（YUV）。

三个模型比较如下表：

model	type	train_loss	val_loss	test_loss
nvidia_model	RGB	0.2571	0.2521	3.19
	YUV	0.2285	0.2285	3.35
refined_model	RGB	3.606	1.1728	2.84
	YUV	3.5172	0.9758	2.57
final_model	RGB	2.5110	0.5188	2.45
	YUV	2.5714	0.6422	2.72

Table 1

根据上面的比较，选择 final_model（RGB）作为最佳模型。

- 可视化模型

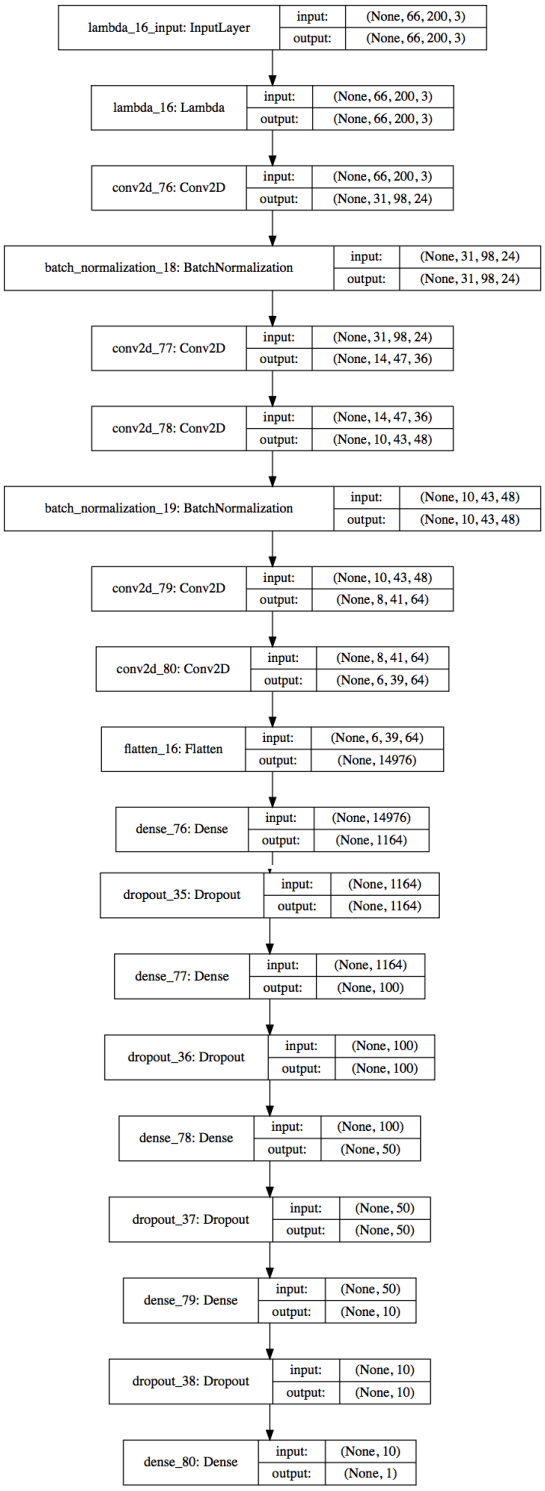

Figure 13

模型的每一层如上图所示。

4. 结果

模型评价与验证

本项目设置第 10 组数据作为测试数据，利用训练好的最佳模型对测试数据进行预测，然后将其与原始司机驾驶的纪录进行比较，比较结果如下：

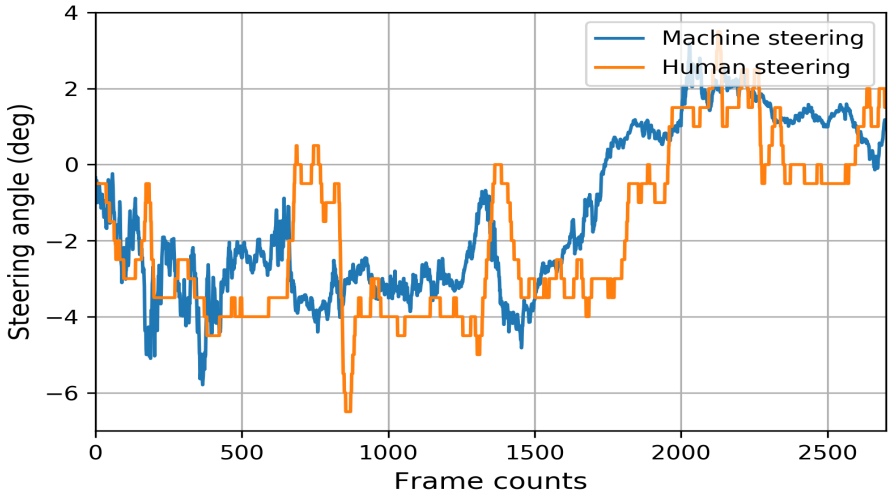

Figure 14

根据上图，可以看出，模型结果的预测大体和司机驾驶记录一致，对于转角的趋势的预测是没有问题的。

但在方向转角变化幅度很大的地方，出现了很明显的错误（预测的转角与司机驾驶记录完全相反）。这和前面的数据探索的分析是一致的，方向转角较大的数据样本太少，而且原始记录的方向转角的变化是不连续的，也就是会出现方向突变的情况，而我们模型的训练函数都是可导的，很难实现方向的突变。并且根据方向转角的分布图，可以发现，方向转角较大的数据样本非常少，训练不充分，所以模型也测会出现偏差甚至错误。

但总体上，模型的训练还是取得了很不错的效果，vitalization 的结果显示，车辆在行驶过程并没有出现大的问题。

合理性分析

最后的结果显示，模型在原有的基准模型上有一些改善。但由于在训练过程中增加了正则化，导致训练时间变长，当然这也是在可接受范围内。

5. 结论

在本次模型的实现过程中，通过尝试调整各种参数，比较好地改善了训练过程过拟合的情况，从而也实现了对基准模型的一些改善，最终也基本达到了预期的要求。

改进

- 训练数据

  通过对原始数据的旋转、调色等处理，制造更加全面的数据，以弥补训练样本在某些区域的缺失。

  对训练数据中出现的一些干扰因素进行清除，让训练数据更加高效。
- 训练模型

  进一步增加神经网络的深度。

  对足够优化的模型进行迁移学习，构造更优模型。

  利用网格搜索的方法寻找模型的最优参数。【8】

参考文献

【1】DeepTesla: End-to-End Learning from Human and Autopilot Driving

【2】End to End Learning for Self-Driving Cars

【3】Autonomous car

【4】Automated and Autonomous Driving

【5】Mean squared error

【6】Convolutional neural network

【7】Learning human driving behavior using NVIDIA's neural network model and image augmentation.

【8】How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras