I. 问题的定义

项目概述

汽车的发明给人类社会带来了巨大的发展，人们的出行变得方便快捷，但随之也带来了交通拥堵与交通事故等问题。而随着计算机技术的发展，越来越多的自动控制技术被应用在汽车上，无人驾驶汽车也成为了汽车产业的一大变革。早在20世纪80年代，美国就提出自主地面车辆(ALV)计划，这是一辆8轮车，能在校园的环境中自主驾驶，但车速不高。世界各大强国也在这方面投入资源。

近年来，随着深度学习的发展，无人驾驶发展非常迅速。其中比较具代表性的分别是Mobileye的自动驾驶布局，英伟达的DRIVE解决方案以及Commaai的方法，后两者都有采用的端到端方法，指以摄像头的原始图像作为输入，中间通过前期使用数据训练出来的模型直接输出车辆的速度和方向。[1]

本项目基于"MIT 6.S094: Deep Learning for Self-Driving Cars"。在该项目中，我们利用MIT 6.S094 这门公开课中的Tesla行驶数据集中前置相机所拍摄的路况图像，训练深度学习模型，利用该模型对车辆转向角度的预测。[2]

问题陈述

该项目是一个监督回归问题。训练数据集中对应每一帧图像都给定了相应的方向盘转向角度。此处使用端到端(end-to-end)模型，端到端模型指的是输入是原始数据，输出是最后的结果。在本问题中，输入X是前置摄像头获取的单帧图片，而最后的结果Y为转向角度，我们的目标就是通过训练学习到一个端到端模型f，利用该模型则可以预测转向角度Y，其关系为Y=f(X)。

评价指标

该项目是回归问题，该问题的评价指标主要有两个，预测结果平均损失函数(MSE)以及模型的训练和预测时长。MSE表达式如下，其中 y_p 表示预测转向角度，y为实际转向角度，N为样本总数。MSE值越小，则模型效果越好。

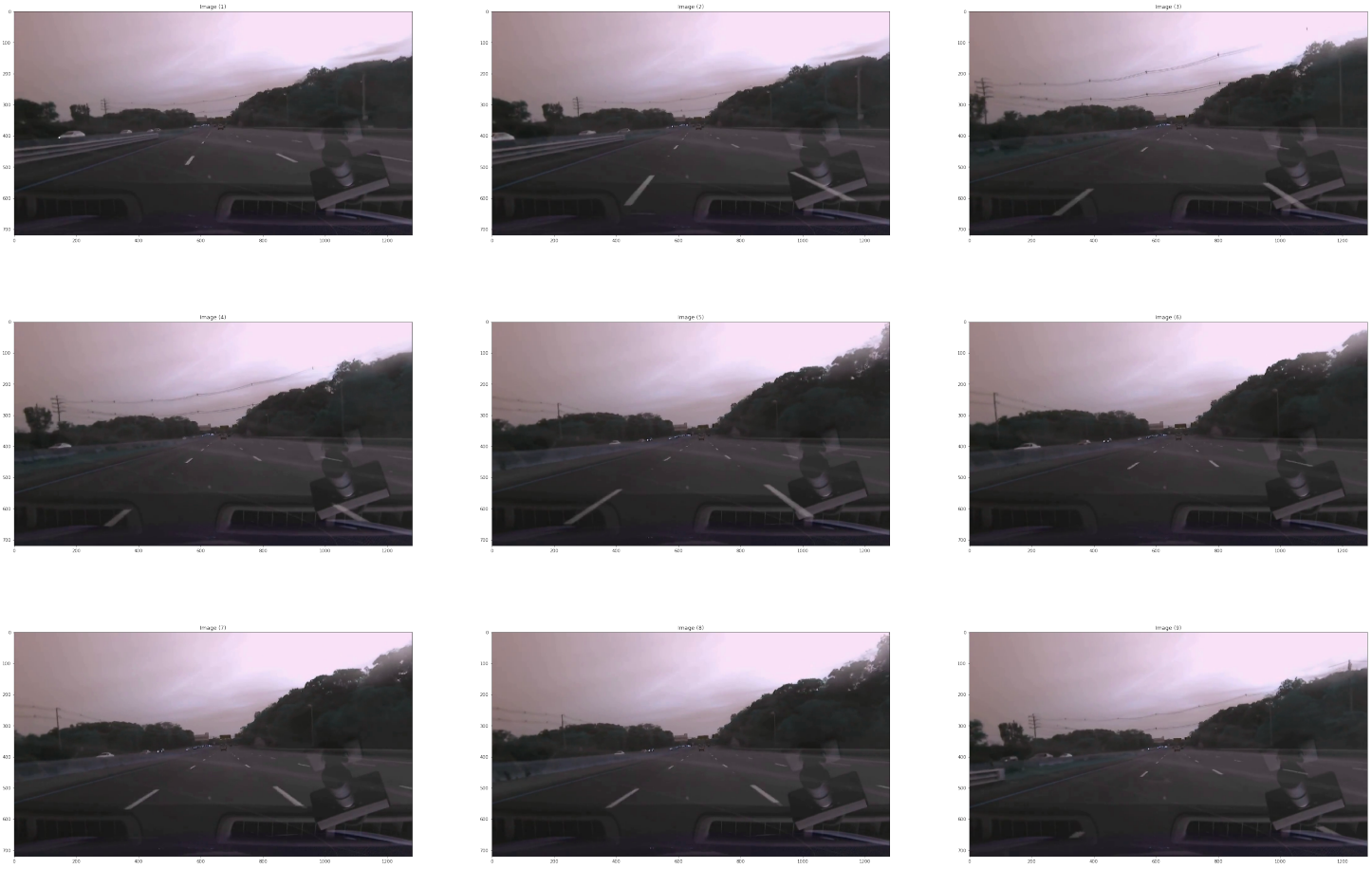
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_p - y)^2$$

II. 分析

数据的探索

先来看一下项目的输入，项目的原始数据来处于汽车的前置摄像头，以mkv的视频格式提供的，共10段视频，其中第10段视频将用于测试，第1到第9个视频经提取后，共有24300张图片，图片形状为（720，1200，3）。

从第1个视频中随机获取9张图如下，我们可以看到，照片中会有大量无用区域(如天空，车辆，相机反光等)，在预处理时需要把这些部分去掉，可以压缩训练数据加快模型训练。

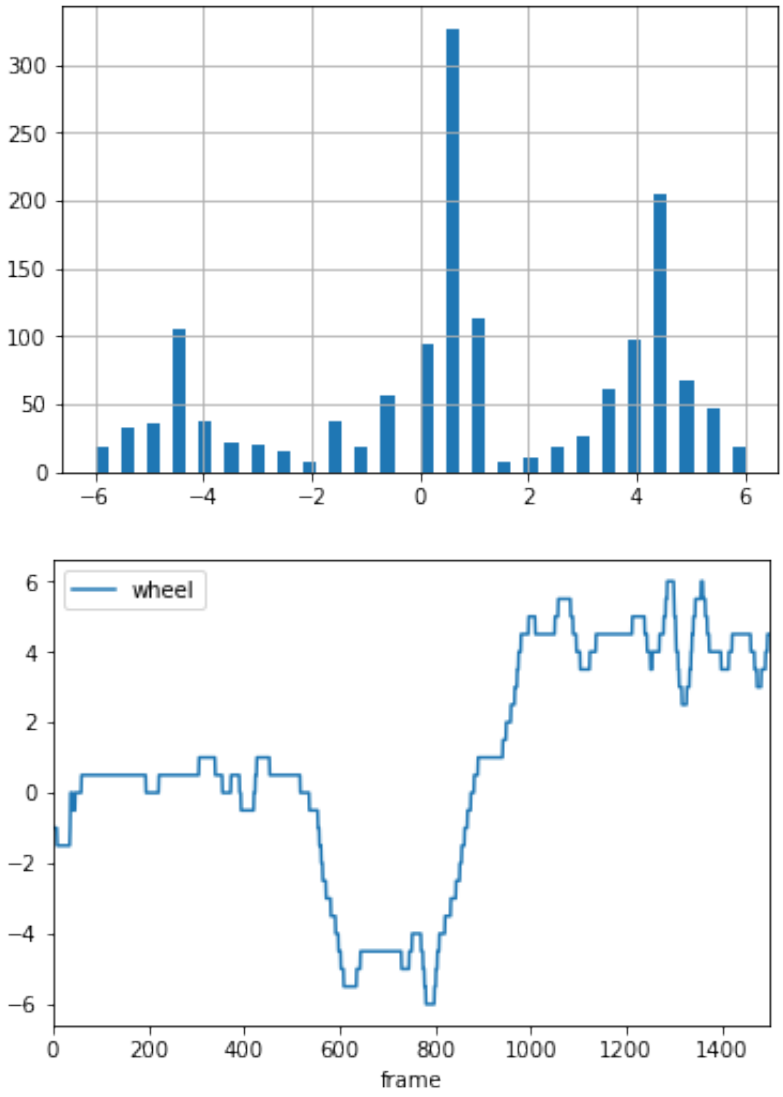


再来看看项目输出，其中，ts_micro表示时间戳，frame_index表示帧编号，wheel表示方向盘转向角度（以水平方向为基准，+为顺时针，-为逆时针）
输出

	ts_micro	frame	wheel
0	1464650070285914	0	-1.0
1	1464650070319247	1	-1.0
2	1464650070352581	2	-1.0
3	1464650070385914	3	-1.0
4	1464650070419247	4	-1.0

探索性可视化

将第一段视频的输出汇总成柱状图和折线图如下。从这两个图可以看到，分布在0附近的信号较多，这里尝试将所有数据进行翻转，让两边的数据进行平衡，模型可以学到有价值的转身策略。



算法和技术

该项目是一个监督回归问题。训练数据集中对应每一帧图像都给定了相应的方向盘转向角度。此处使用端到端(end-to-end)模型，端到端模型指的是输入是原始数据，输出是最后的结果。在本问题中，输入X是前置摄像头获取的单帧图片，而最后的结果Y为转向角度，我们的目标就是通过训练学习到一个端到端模型f，利用该模型则可以预测转向角度Y，其关系为Y=f(X)。

由于具有图片状态空间巨大、图像含义在不同位置的平移不变性(translation invariance)等特点，传统的监督学习算法并不适用，目前最适合于图片检测的算法是卷积神经网络(CNN)。卷积神经网络不同层之间的神经元并不是全连接，而是采用局部连接的方式，这使得网络上的参数大大减少，并且共享权重，具有很好的物体平移不变性。

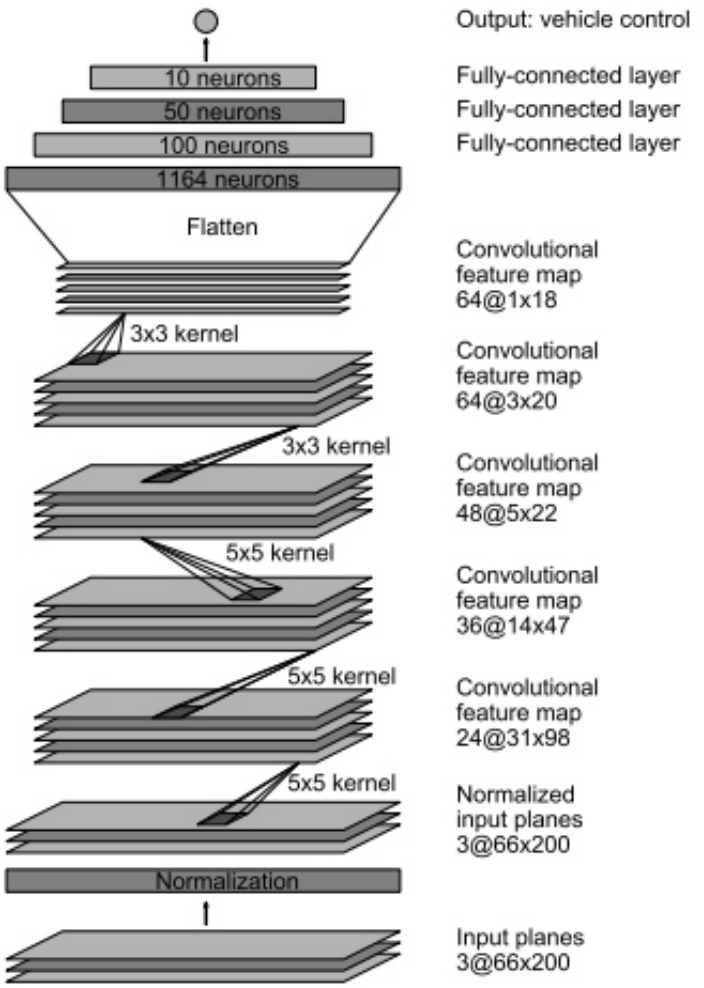
在训练中，涉及到的参数如下：

- 预处理:图片尺寸(size)，感兴趣区域（region of interest,ROI）
- 超参数:训练次数(epoch),批处理大小(batch size),优化器类型(optimizer type),学习速率(learning rate)
- 神经网络架构:网络层数(layers),网络层类型(layer types)，网络层初始化参数及偏置项(weights and bias)

基准模型

英伟达（NVIDIA）一直以来发力进行深度学习和无人驾驶的研究，今年年初NVIDIA就发表了一篇论文介绍如何利用CNNs实现端到端的无人驾驶[3]。此处采用该模型来作为基准模型。

如下图所示，该模型第一层为归一化层，接下来的紧跟着5个卷积层，在卷积层后增加3个全连接层。



III. 方法

数据预处理

数据预处理中，本项目做了以下处理。

先将视频中的所有图片提取出来，并且提取输出结果，使图片与输出一一对应。

由于图片尺寸比较大（720，1200，3），并且无用区域较多，此处会对所有图片进行裁剪，只留下感兴趣区域。

执行过程

英伟达端到端卷积神经网络模型

英伟达模型的情况在基准模型章节已经有介绍，此处就不再描述。代码实现如下

```
from keras.layers import *
from keras.models import Model
from keras import optimizers
from keras.models import model_from_json
import keras
import os
import params


inputs = Input(shape=(params.FLAGS.img_h, params.FLAGS.img_w, params.FLAGS.i
mg_c))
x=Lambda(lambda x:x/255.0)(inputs)
x=Conv2D(24, (5, 5), activation="relu", strides=(2, 2), padding="valid")(x)
x=Conv2D(36, (5, 5), activation="relu", strides=(2, 2), padding="valid")(x)
x=Conv2D(48, (5, 5), activation="relu", strides=(2, 2), padding="valid")(x)
x=Conv2D(64, (3, 3), activation="relu", strides=(1, 1), padding="valid")(x)
x=Conv2D(64, (3, 3), activation="relu", strides=(1, 1), padding="valid")(x)
x=Flatten()(x)
x=Dense(1164, activation='relu')(x)
x=Dense(100, activation='relu')(x)
x=Dense(50, activation='relu')(x)
x=Dense(10, activation='relu')(x)
outputs=Dense(1)(x)
model = Model(inputs=inputs,outputs=outputs)
model.compile(optimizer=optimizers.Adadelta(),loss='mse',metrics=['mse'])
model.fit_generator(
        myGenerator(),
        steps_per_epoch=84,
        epochs=10,
        validation_data=(X_val, y_val))
```

```
test_loss= model.evaluate(X_test, y_test)
print('Test loss is:{}'.format(test_loss))
```

执行完后得到的val_loss为1.9647，test_loss为4.9945

模型中使用了adadelta优化器，Adadelta是对Adagrad的扩展，最初方案依然是对学习率进行自适应约束，但是进行了计算上的简化。Adagrad会累加之前所有的梯度平方，而Adadelta只累加固定大小的项，并且也不直接存储这些项，仅仅是近似计算对应的平均值。它的优点是在训练初中期，加速效果很快，但缺点在于在训练后期会出现反复在局部最小值附近抖动。另外，Adadelta无需设置全局学习率，因此在这里选择其作为模型优化器。

完善

在实现过程中尝试对该结构进行池化，以及替换了其激活函数，得到的结果如下：

模型	val_loss	test_loss
原始模型	1.9647	4.9945
池化模型	2.2577	3.1688
elu模型	4.7659	3.1192
最终模型	3.6358	3.044

完善过程如下：

在尝试了英伟达的模型后，还可以进行一些别的尝试，看是否能得到更优的模型，在这个过程中最先想到的是尝试了修改epoch，增加训练次数，但从实验结果来看，在第二次时loss就下降了许多，10次后就基本收敛，增加训练次数已达不到优化目的。

另外还尝试修改模型的结构，增加了池化层，它主要有以下两个作用，
1.不变性，这种不变性包括平移，旋转，尺度
2.保留主要的特征同时减少参数(降维，效果类似PCA)和计算量，防止过拟合，提高模型泛化能力。
在增加了池化层后test_loss有所下降。

原始模型使用的激活函数是Relu,当x<0时，ReLU硬饱和，而当x>0时，则不存在饱和问题。所以，ReLU 能够在x>0时保持梯度不衰减，从而缓解梯度消失问题。这让我们能够直接以监督的方式训练深度神经网络，而无需依赖无监督的逐层预训练。

而elu则是融合了sigmoid和ReLU，左侧具有软饱和性，右侧无饱和性。右侧线性部分使得ELU能够缓解梯度消失，而左侧软饱能够让ELU对输入变化或噪声更鲁棒。ELU的输出均值接近于零，

所以收敛速度更快。

在尝试把模型的激活函数换成elu后，得到的test_loss要比原始模型有所下降。

最后的模型是在原始模型中加入池化层以及替换激活函数为elu，得到的模型的test_loss为3.044，较原始模型有了较为显著的提高。

IV. 结果

模型的评价与验证

每个模型使用前8个视频来训练，第9个视频作为校验集，而第10个视频作为测试集，最后使用预测的角度生成新的行驶视频。与人类驾驶数据相比，会存在一定的偏差，但可从视频中可见，并未出现明显的错误，考虑到人类驾驶的方案也不一定是最完美方案，在未出现明显错误情况下，我们可以确认该结果是可信的。

合理性分析

最终结果对比基准模型表现上差不多，但最终结果可见该模型并未出现明显错误，因此最终结果是解决了实际问题。

V. 项目结论

结果可视化

具体结果已经生成为对比视频，从生成的视频中可以看出，在尝试对英伟达原始模型后，为了提高性能，做了几种尝试，使用maxpool和dropout，替换激活函数为elu，最终得到的模型较为理想，在行驶过程中并无明显错误发生。

对项目的思考

整个项目中，从拿到课题开始，先是进行了对数据的探索，并对数据进行了一些初步的预处理，然后找到了业界较为理想的方案进行了复现，并且对其尝试优化，整个过程下来，对深度学习的应用有了较为深入的了解。项目中比较困难但同时也是有意思的地方在于，并没有一个很明显的标准来判断最终生成的结果是否是合理的，只能通过实际行驶情况来判断，但实际行驶情况多种多样，可以说是不可能穷举完的，因此在这个地方还值得我们进行更多的深入思考，到底最终以一个什么样的标准来判断该自动驾驶是安全可靠的。

需要作出的改进

为了得到更好的性能未来可以尝试从以下方面进行优化

1.在数据增强方面可以做更多的尝试；

2.由于目前数据量较少，未来可尝试使用GAN[4]来生成更多的训练数据；

3.目前只基于英伟达的模型进行的尝试，即使用了CNN模型，未来可探索RNN[5]和LSTM[6]在自动驾驶方面的应用。

引用

[1]新智元,"深度学习驱动的自动驾驶新主流框架盘点",http://www.weixinnu.com/article/57d45cd90695bce9092ea794,2016.

[2]jiandong,"Deep Learning for Self-Driving Cars : DeepTesla",https://github.com/nd009/capstone/tree/master/deep_tesla,2017

[3]M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang et al., “End to end learning for self-driving cars,” arXiv preprint arXiv:1604.07316, 2016.

[4]Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Joshua (2014). "Generative Adversarial Networks". arXiv:1406.2661

[5]Stelios Timotheou "The Random Neural Network: A Survey", Comput. J. 53 (3): 251–267, 2010.

[6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.