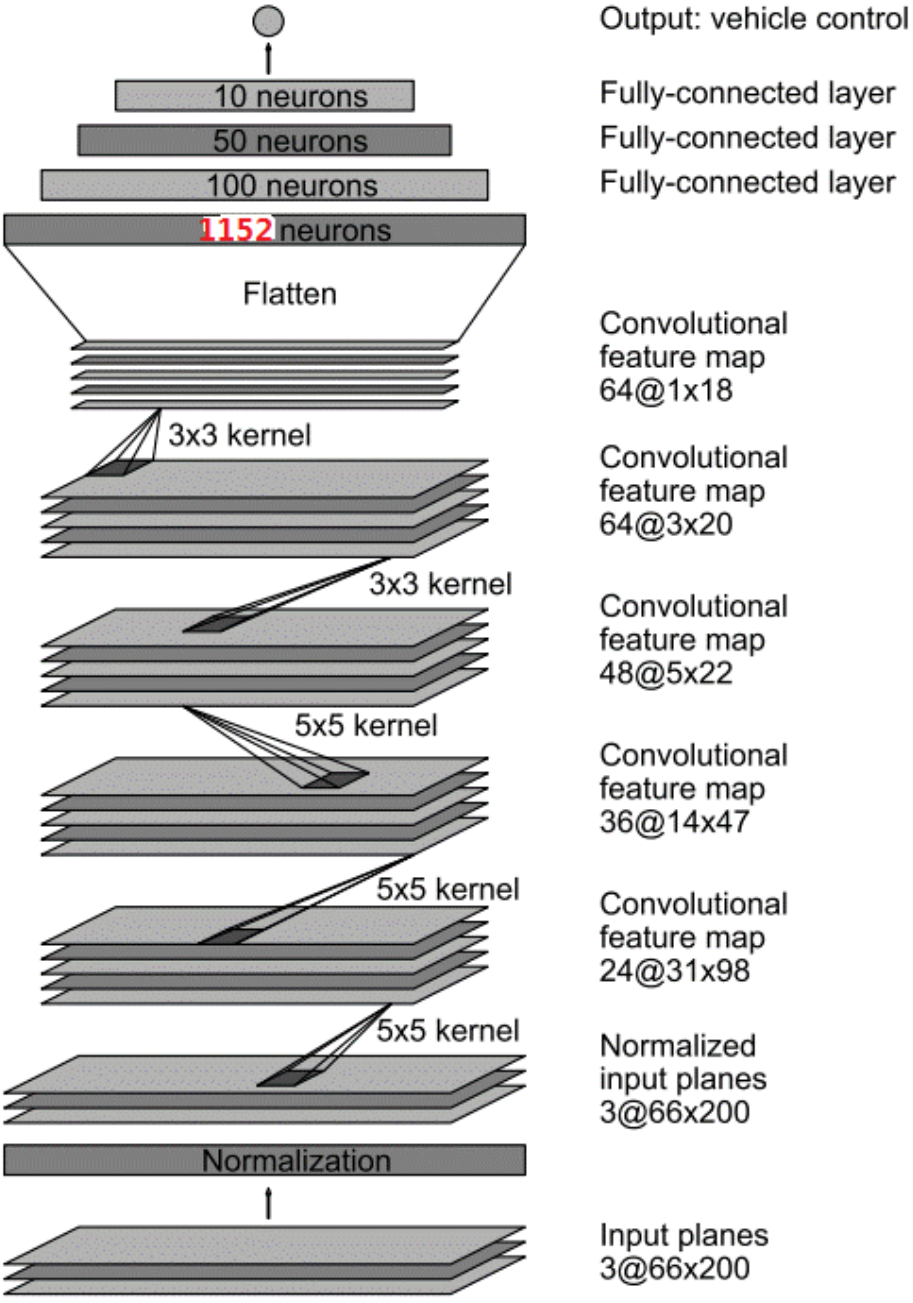
Assignment 1

一、 实现一个 CNN 模型

1. 学习 CNN

2. 使用 Tensorflow 建立如下 CNN 模型



1) 色彩空间转换

load_data.py 中的 LoadTrainBatch() 函数和 LoadValBatch() 函数实现了对数据的预处理并按需返回相应 batch。阅读 paper "*End to End Learning for Self-Driving Cars*" 的过程中发现有一处地方其并未实现：

"*The input image is split into YUV planes and passed to the network.*"

意思是将图像的 RGB 空间转化到 YUV 空间，故调用 opencv 库函数实现。

添加色彩空间转换语句：

```
image = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)
```

2) 直方图均衡化

paper 的 Network Architecture 中第一层为 normalization 层，查阅资料后了解了一般的 CNN 模型中 normalization 层理解为 batch normalization 层，用于批数据的归一化，可加快收敛速度。一开始是这么理解的，但后来了解到其实每一层的激活函数前都可以有一个 batch normalization 层，此处特意标注出来想必是不一样的用意。

注意到此处的用语为 image normalization，而非 batch normalization。查阅维基百科得知 image normalization 是图像处理中的工作，类似于直方图均衡化。所以再次调用 opencv 库函数实现 image normalization。

在色彩空间转换语句前添加三通道的直方图均衡化操作：

```
for i in range(0, 3):
    image[:,:,i] = cv2.equalizeHist(image[:,:,i])
```

3) model.py 实现

参数 W 均从均值为 1，方差为 0.1 的高斯分布中随机获取，而参数 b 则设为固定常数 0.1。经测试效果良好。

每一层的激活函数均使用 relu，即函数 f(x) = max(0, x)，此举为在线性组合中增加非线性因素。全连接层中添加 keep_prob 占位符，在传递时以某一指定概率置其权值为 0，即 drop_out，以防止过拟合。

除最后一层输出层外，每一层激活函数之前均采用归一化操作 batch normalization，以加快收敛速度。

观察数据集范围，99.5%的数据在[-180, 180]之间，故将输出映射至此范围内可加快收敛速度。而 load_data.py 中将从 data.txt 中读取的数据*PI/180 作为对应的 Y 值，此时值域为[-PI, PI]，而此神经网络的输出理论上为整个实数域，可取有界函数 arctan，其值域为[-PI/2, PI/2]，将其乘 2 即为理想值域范围。

二、 训练一个 CNN 模型

1. 损失函数(loss function: mean square loss)

损失函数定义为 loss = mean square loss + 所有训练参数 * 自定义系数 lambda

2. 选择优化方法、确定学习率(1e-4)

优化方法选择 adam 优化器，查阅资料得此优化器的平均性能最好。
学习率取 1e-4。

3. 保存训练模型

通过 tf.train.Saver 保存训练模型的 checkpoint，通过 tf.summary.FileWriter 保存 logs。之后可通过 tensorboard 查看存储的模型信息。

三、 讨论与思考

1. 实验结果(参数选择，loss value 曲线图)

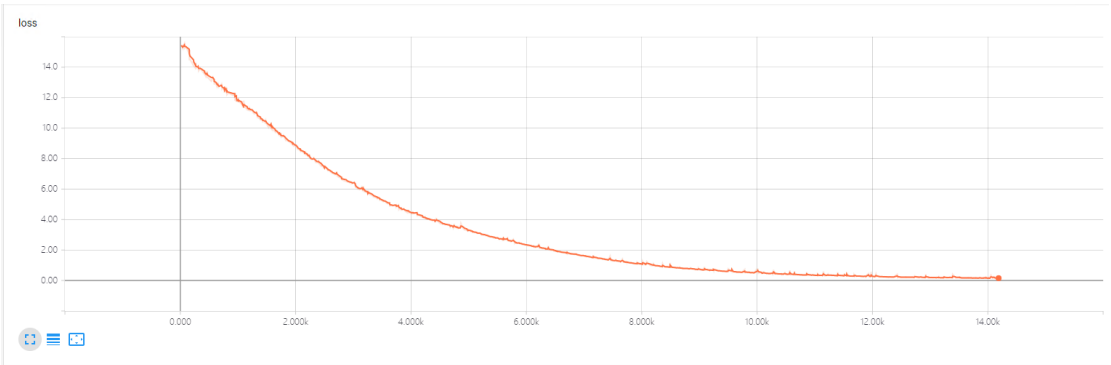
1) 参数选择

损失函数中的自定义参数 lambda 的大小与训练的参数个数有关，此处网络层次较多，相对参数也多，故选取一个较小值，取 0.01，与均方误差求和以取得平衡，使得参数绝对值较小，以防过拟合。

epoch 取 50，batch_size 取 128（取 2 的幂次方有助于提供并行效率，此为显存可容纳最大 2 的幂次方数）

一开始以为每个 epoch 开始前均需打乱数据，以增强训练的随机性，但后来发现给定的 load_data.py 中巧妙地利用了取模的方式来循环取 batch，以增强每个 batch 的随机性：只要 data 个数不是 batch_size 的整数倍，就会产生与之前 epoch 不一样的 batch。

2) loss value 曲线图



四、 Reference

[1]. Bojarski M, Del Testa D, Dworakowski D, et al. End to end learning for self-driving cars[J]. arXiv preprint arXiv:1604.07316, 2016.

[2]. https://cs231n.github.io/convolutional-networks
[3]. http://colah.github.io/posts/2014-07-Conv-Nets-Modular/
[4]. https://github.com/SullyChen/Autopilot-TensorFlow
[5]. https://en.wikipedia.org/wiki/Normalization_(image_processing)
[6]. http://blog.csdn.net/cxmscb/article/details/71023576