

机器学习纳米学位毕业项目：Deep Tesla

学员：Fei He

2017 年 11 月 11 日

目录

I. 问题的定义	4
1.1 项目概述	4
1.1.1 项目背景	4
1.1.2 数据集和输入	4
1.2 问题陈述	5
1.2.1 问题描述	5
1.2.2 问题定义	6
1.3 评价指标	6
II. 分析	6
2.1 数据的探索	6
2.2 探索性可视化	7
2.3 算法和技术	8
2.3.1 端到端	9

目录

2.3.2 卷积神经网络	9
2.3.3 迁移学习	10
2.4 基准模型	10
III. 方法	12
3.1 数据预处理	12
3.1.1 数据简化	12
3.1.2 数据增加	12
3.2 执行过程	13
3.2.1 基准模型改进	13
3.2.2 大转向角度数据添加	14
3.3 完善	15
3.3.1 构建	15
3.3.2 训练	16
IV. 结果	17
4.1 模型的评价与验证	17
4.2 合理性分析	19
V. 项目结论	19
5.1 结果可视化	19

5.2 对项目的思考	19
5.3 需要作出的改进	20
VI. 参考文献	21

I. 问题的定义

1.1 项目概述

1.1.1 项目背景

近年来，深度学习神经网络在计算机视觉领域，特别是图像识别上取得的突破性进展无疑是科技界的热点之一。诚然，神经网络并非新兴技术，从时间线上可以看出它的发展脉络：1958年康奈尔心理学家罗森布拉特推出感知机；1963年，弗拉基米尔·万普尼克在他的书《支持向量机》中首次发表反向传播算法；1986年Hinton等人发明训练多重神经网络纠错的方法；1989年LeCun用卷积神经网络识别手写体；1991年递归神经网络发明；2007年李飞飞创立ImageNet；2012年谷歌大脑识别猫脸（6月），8月谷歌将神经网络引入语音识别，10月Hinton的学生在ImageNet竞赛夺冠，成绩大幅提升；2015年12月微软ResNet图像识别准确率超越人类。在将近50年的发展历史中，其迎来爆发的根本原因就在于现今计算机强大的计算力和海量的数据。现在，这项技术已经延伸到医疗、游戏、公共服务等等和人们生活息息相关的各个层面之中。

在众多深度学习神经网络应用场景之中，无人驾驶（自动驾驶）无疑是中短期内最有前景的之一。现阶段，以摄像头作为道路图像传感器，结合人工智能和视觉处理技术实现无人驾驶（自动驾驶）在成本上体现了巨大的优势。不同的神经网络模型已经应用到车辆转向控制和速度控制中。如英伟达（NVIDIA）提出的端到端[1]（end to end）卷积神经网络模型（Convolutional Neural Network，以下简称CNN或ConvNet）；Comma.ai提出的变分自动编码器[2]（Variational Autoencoder，以下简称VAE）及基于生成式对抗网络（GAN generative adversarial network，以下简称GAN），用于实现道路视频预测的代价函数（Cost Function），在此基础上结合了递归神经网络（Recurrent Neural Networks，以下简称RNN）的转换模型（Transition Model）；最近，在UDACITY的开源挑战项目“Teaching a Machine to Steer a Car”[3]，第一名的“Team Komanda”利用了循环神经网络序列到序列模型（RNN seq2seq, Recurrent Neural Networks Sequence to Sequence），使用CNN结合长短期记忆网络（Long Short-Term Memory，以下简称LSTM）实现了一个不同长度序列输入图像到输出转向角度的映射模型。

在这些已经实现的案列中，与传统的方法相比，神经网络和深度学习展现了惊人的效率和优势。本项目就是基于MIT 6.S094这门公开课中的Tesla数据集训练深度学习模型，根据车辆的前置相机所拍摄的路况图像，实现对车辆转向角度的预测，从而对深度学习模型有一个初步的探索。

1.1.2 数据集和输入

数据集包括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。数据可以从[这里下载](#)：[百度云](#)，数据集包含10段H264/MKV格式视频，视频帧为30fps，截取画面像素为1280 * 720，视频数据格式如图“Figure 1: 视频帧图片”：

每一个视频都对应一张存储有对应时间戳（ts_micro）和帧编号（frame_index）和转向角度（wheel）的CSV



Figure 1: 视频帧图片

格式表格。表格见“Table 1: CSV 表格数据”。其中转向角度（wheel）中‘+’表示顺时针，‘-’表示逆时针。

ts_micro	frame_index	wheel
1464305394391807	0	-0.5
1464305394425141	1	-0.5
1464305394458474	2	-0.5

Table 1: CSV 表格数据

在项目中，通过裁剪图像（去掉天空部分）、缩减像素采样原始数据得到了宽、高为 $80 * 80$ 的 RGB 图像，并将图像和转向角度以 JPEG 格式和 CSV 格式保存到硬盘中作为输入数据，具体步骤将在“数据预处理”章节进行详诉。

1.2 问题陈述

1.2.1 问题描述

该项目的目的在于利用深度学习模型：端到端 NVIDIA end-to-end 卷积神经网络模型（CNN）；VGG16 + Nvidia 迁移学习（Transfer Learning）模型。根据车辆的前置相机所拍摄的路况图像，实现对车辆转向角度的预测，并评估模型的表现。

1.2.2 问题定义

假设 x_t 表示的是数据集的第 t 帧， X_t 是帧长为 n 的视频表示： $X_t = \{x_{t-n+1}, x_{t-n+2}, \dots, x_t\}$ ， Y_t 表示对应对应帧长为 n 的转向角度： $Y_t = \{y_{t-n+1}, y_{t-n+2}, \dots, y_t\}$ ，预测转向角度时定义估值函数 $F: R^{80*80*3*n} \rightarrow R^n$ ，下一帧转向角度预测结果为 $y_{t+1} = F(x_{t+1})$ 。

该项目的任务就是训练模型得到较好的估值函数 F ，从而对预测视频帧的转向角度有一个比较准确的预测。

1.3 评价指标

就整个项目来说，这是一个回归问题，均方误差（Mean Squared Error, MSE）可以作为评估指标。

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

其中， y_p 是预测的转向角度， y 是真实的转向角度，样本数量为 N ，均方误差低则模型表现优。

同时，要结合分析预测转向角度的时间序列图和人工转向角度的时间序列图的“贴合”程度（是否接近人工驾驶）来考虑哪一个模型表现更好。

II. 分析

2.1 数据的探索

在初始对数据探索阶段，分别查看了单张测试图片和 10 段道路视频文件，对不同的驾驶场景进行了截图，“Figure 2: 不同场景截图”：从图中可以看出，视频帧包括不同的时段（如正午，傍晚）、不同的天气情况（如阴晴），不同的道路情况（如桥，林间路）等。视频通过前置相机拍摄，在底部可以看到车头以及相机阴影。查看了 CSV 格式表格数据，发现数据包含 27000 个样本，并且转向角度的数值在 [-18, 15] 区间。



Figure 2: 不同场景截图

2.2 探索性可视化

目标变量 CSV 表格中的转向角度，需要通过直方图查看其平衡性：如图“Figure 3: 转向角度数据分布”：可以看出，整个转向角度数据分布基本为正态分布，车辆的控制信号中角度为 0 附近的数据比较多，意味着得到的模型可能对比较大的转弯反应“迟钝”，解决方法则可以通过复制转向角度较大的数据以增加训练数据来解决，使得模型学到有价值的转向策略（一般情况，一个好的数据集应该是顺时针、逆时针、直行的转向信号频次足够接近的数据集，即“对称”的数据集。但从逻辑上，人类开车时需要转弯的频率是要远低于直行微调方向盘的频率，增加整体训练数据也可以让模型学到转弯的策略）。

另外，需要查看转向角度在时间序列上的分布：如图“Figure 4: 转向角度的时间序列”：

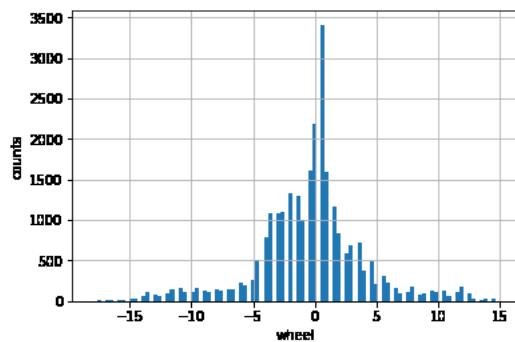


Figure 3: 转向角度数据分布

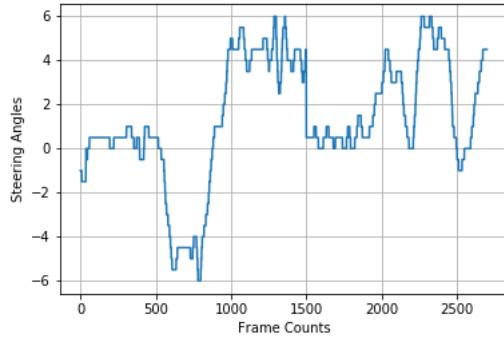


Figure 4: 转向角度的时间序列

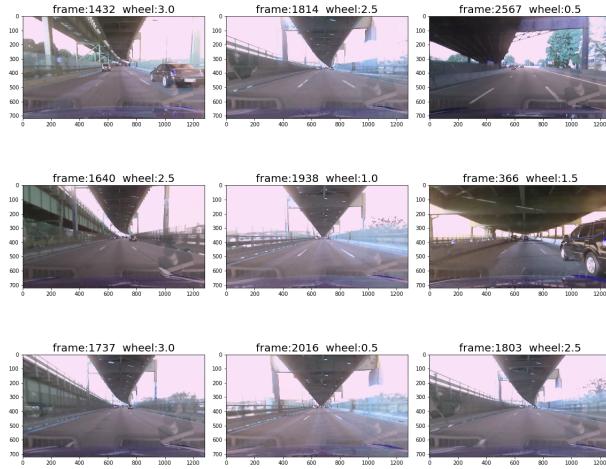


Figure 5: 随机帧与转向角度

查看了前 2700 帧（90 秒）图像的转向角度的时间序列，结合 CSV 表格可以看出，人工转向记录的数据是离散的，非连续的。但从逻辑上来说，它仍是求一个具体数值，而不是对这些角度分类，所以还是一个回归问题。

最后，随机从一段视频中选择几帧查看一下人工转向角度是否合理，如图 Figure 5。可以发现人工转向角度基本正确，幅度调整得当，说明驾驶员的驾驶策略是比较合理的，提供的数据可以用来训练。

2.3 算法和技术

在该项目中涉及到的算法和技术有：端到端技术（end-to-end）、卷积神经网络（CNN）、迁移学习技术（Transfer Learning）。现分述如下：

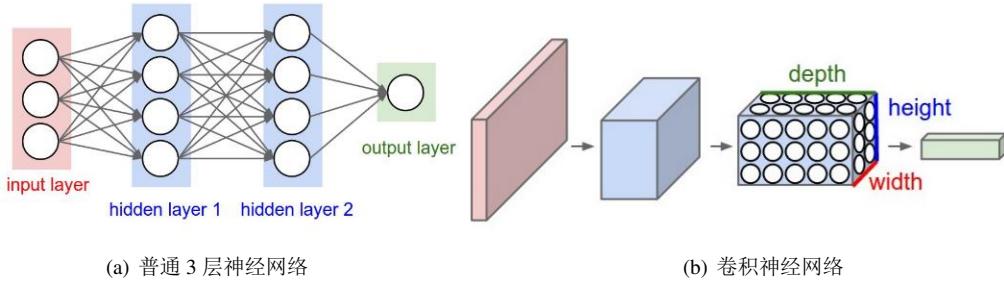


Figure 6: 普通神经网络与卷积神经网络对比

2.3.1 端到端

端到端的学习方法可以追溯到 80 年代后期[4]，其方法可以简述为神经网络输入为原始图片，神经网络输出为直接控制的命令。其特点在于，特征提取层和控制输出层的分界并不明显，网络中的每一个部分对于系统来说都起着特征提取和控制作用。

2.3.2 卷积神经网络

为了引入端到端的技术，需要从原始图片提取相关信息（特征）。从效率和减少网络参数个数考虑，卷积神经网络（CNN）无疑是最佳选择。它的结构和普通神经网络（如图 Figure 6(a)）都一样，其主要不同之处在：CNN 假定输入的是图片，根据图片的特性对网络进行设定以达到提高效率，减少计算参数量的目的。具体做法为：各个网络层将神经元安置到三维空间（width, height, depth），每个神经元接收一些输入，前向传播进行点积运算和选择性的使用非线性运算（如激活函数），然后使用反向传播算法更新权重（weights）和偏差（bias），可见图“Figure 6(b)”。在本项目中，红色输入层是维度为 $80 * 80 * 3$ （宽、高、RGB 通道）的图片，输出则是转向角度数值。

卷积神经网络有三种层：卷积层、池化层和全连接层（Convolutional Layer, Pooling Layer, 及 Fully-Connected Layer）。卷积层由很多过滤器组成，每个过滤器都只有一小部分，每次只与原图像上的一小部分连接，每个过滤器在原图像滑动得到一个特征图（feature map），假如有 k 个过滤器则形成的特征图深度（depth）为 k 。这样，通过卷积层，就得到了新的三维的图像特征表示，通过权重共享大大提高了计算效率；池化 pooling 层，他是将卷积层得到的结果无重合的分成几部分，然后选择每一部分的最大值，或者平均值，或者 2 范数，或者其他值，如图 Figure 7。全连接层则是每一个神经元都与上一层的所有神经元相连，用来把前边提取到的特征综合起来。

通过这三种网络层的依次连接，卷积网络就完成了原始图片到转向角度的映射，为了使模型有更好的表现，需要调整的参数有：1. 对图片预处理的参数包括图像的尺寸和图像的感受区域范围；2. 超参数：训练次数（number of epochs），批量大小（batch size），优化类型（optimizer type），学习率（learning rate）；3. 网络结构：网络层的数量，网络层的类型，网络层中的参数调节（参数的初始优化，激活函数等）。

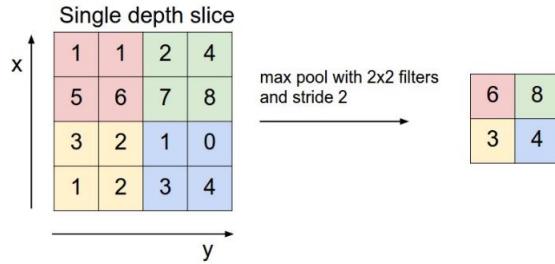


Figure 7: 池化 pooling 层

2.3.3 迁移学习

迁移学习就是用大数据集（如 ImageNet）训练过的卷积网络作为初始模型或者直接作为固定特征的提取器[5]，即是说把预先训练好的模型，应用到类似的处理任务上，达到节约训练时间和提高模型表现的目的。以下是三种迁移学习的策略：1. 训练好的卷积网络作为固定特征提取器（ConvNet as fixed feature extractor, train only the top-level of the network, the rest of the network remains fixed）。使用训练好的网络将最后一层全连接层（一般是分类用的）去掉或替换，然后将已经训练好的部分看做一个特征提取器；2. 对卷积网络进行微调（Fine-tuning the ConvNet, train the entire network end-to-end, start with pre-trained weights）。第二个策略是不仅对顶层的全连接层进行重新训练和替换，同时通过反向传播对已经训练好的卷积层参数进行微调。可以对所有网络层的参数都进行微调，也可以让前几层的参数固定（前几层一般包含通用特征的识别，如颜色、线条等），然后对高层部分进行微调（包含更多训练集的特征）；3. 预训练模型（Pretrained models, train the entire network end-to-end, start from random weights）。使用别人发布的在大数据集（如 ImageNet）训练卷积网络的 checkpoints 进行重新的参数训练。策略 2 和策略 3 的区别在于 2 使用的是在训练好的权重上进行参数微调，而 3 需要在随机的参数权重上重新开始训练。

在本项目中，作为迁移学习的卷积网络为 VGG16[6]，采用的策略为 2: Fine-tuning（模型中称为 fine-tuned，即参数微调）具体讨论见“III. 方法”章节。

在本项目中的解决方法为：使用 NVIDIA 提出的结构端到端（end-to-end），所谓端到端指的是神经网络输入为原始图片，神经网络输出为直接控制的命令。其特点在于，特征提取层和控制输出层的分界并不明显，网络中的每一个部分对于系统来说都起着特征提取和控制作用；另一种方法是利用迁移学习（Transfer Learning）VGG16 在第一种方法的基础上优化神经网络模型。通过训练数据训练模型，然后使用模型完成从测试图像到转向角度的映射。

2.4 基准模型

本项目尝试了两种不同模型：第一种是英伟达（NVIDIA）提出的端到端（end-to-end）卷积神经网络模型（convolutional neural network），该模型将作为基准模型；第二种基于迁移学习（transfer learning）的 VGG16 + Nvidia 模型，该模型作为对比模型，对比模型将在下一章讨论。本质上两种模型都是一致的，使用卷积层提取特征，在加上全连接层进行回归预测。

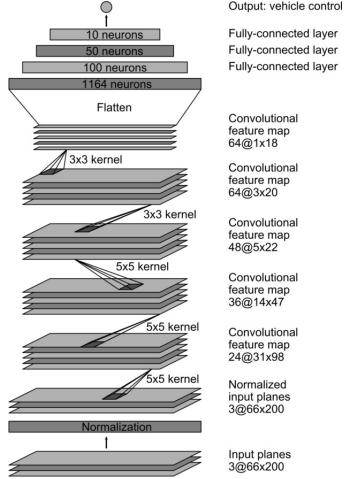


Figure 8: Nvidia 原始模型架构

基准模型 Nvidia model 参考了 NVIDIA 发表的一篇论文[1]提出的模型架构，该论文介绍如何利用卷积网络（CNNs）实现端到端的无人驾驶，具体模型见图如图 Figure 8。简单分析一下该模型的结构：输入层接收是宽、高（200, 66）并且有 3 个颜色通道的图片，在项目中将调整为（80, 80）的 RGB 图片；网络的第一层归一化层（Normalization Layer）采用了归一化操作，归一化作为输入卷积层之前的预处理操作，进行归一化操作有助于适配不同类型的网络结构，并且加速模型在 GPU 上的训练，在项目中省略了归一化层（把归一化操作放在数据预处理阶段，具体见下一章）；接下来的紧跟着五个卷积层（Convolutional Layer），前三个卷积层选择了 5x5 的 kernel 和 2x2 的 strides，后两层卷积层选择了 3x3 的 kernel 并且没有 strides，卷积核数量逐层增多（24→36→48→64→64）是有一定的解释的，卷积神经网络的特点是“局部视野”+“权值共享”，通过选取图像中部分区域（通过卷积核的大小决定）并选择多个卷积核（权值共享）来将提取到的局部特征应用到全局图片，卷积核的数量和层数越多可以提取到的特征越“通用”；卷积层之后增加了 3 个全连接层（Full Connected，简称 FC），FC 层用来对卷积层提取到的特征进行训练并最终输出转向控制信号。

注意的是卷积层与 FC 层之间通过 Flatten 连接，实现了将卷积层输出的多维向量变成一维向量，用于输入 FC 层。模型构建好了以后，需要编译并选择训练优化算法，选择的是深度学习中常见的算法 Adam optimizer[7]。同时，在除最后输出层外，其他各层使用 RELU 激活函数[5]，RELU 输出所有负值为 0，正值为本身，ReLU 的优点在于它不会饱和，收敛快（即能快速找到代价函数的极值点），而且计算简单（函数形式很简单）；但是收敛快也使得 ReLU 比较脆弱，如果梯度的更新太快，还没有找到最佳值，就进入小于 0 的函数段，这会使得梯度变为 0，无法更新梯度直接挂机了。所以，对于 ReLU 神经元，控制学习率（learning rate）十分重要。此外，它的输出也不是均值为零 0 的。

之所以要选择该模型为基准模型，是因为模型网络层不多且清晰，并且基于已被验证的假设（在实际道路测试中有良好效果，见论文[1]）。在经过 10 次 epochs 之后（经过简单测试，发现 5 次 epochs 训练不充分，20 次 epochs 易过拟合，10 次 epochs 训练合适，其它模型都将设置为 10 次以对比），生成器的 batch size 为设置为 128（数据预处理见下一章），基准模型的‘training loss’（MSE）值为 0.5361，‘validation loss’（MSE）值为 0.5013，在测试集上 MSE 值为 8.8617。从损失值看，基准模型学会了一定的驾驶策略，但存在过拟合（训练集和验证集损失值低，测试集高）。具体训练结果见图 Figure 9。

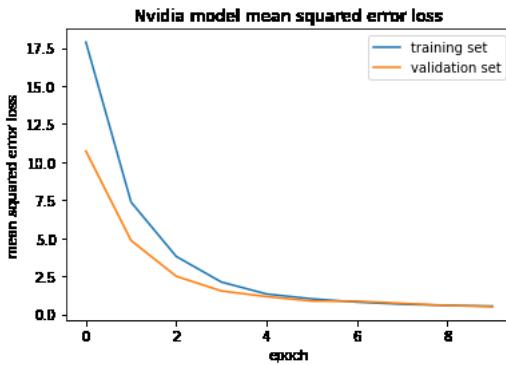


Figure 9: Nvidia 基准模型训练结果图

III. 方法

3.1 数据预处理

数据预处理策略为：数据简化、数据增加。

3.1.1 数据简化

原视频文件如果直接转换为图片读入内存中，会导致内存溢出（图片太大），所以需要对其进行简化，具体步骤如下：

1. 将所有视频文件转化为图片（RGB 颜色通道，JPG 格式）。
2. 在转换过程中，对每一张图片进行裁剪（将图片中天空部分剪掉），然后重设图片尺寸为宽、高为（80, 80）的 RGB 颜色图片。
3. 将前 9 段视频转换为训练图片保存在硬盘中，将第 10 段视频转换为测试图片保存在硬盘中（经过裁剪和重设尺寸后可以将所有图片直接保存内存，在后面只对测试集使用此方式）。同时将对应的转向角度和图片路径保存为 CSV 文件。具体代码见‘preprocess_data.py’文件中的‘load_data()’函数。

3.1.2 数据增加

深度学习只有在大量的训练集上才能体现出它的威力，过小的数据集会导致模型训练不充分。在本项目中，用于训练模型的数据集大小为 24300，这些数据对于真实世界的路况而言太少了，需要增加数据。

另外，对于大型图像的处理任务，图片不可能一次性加载入内存，在单机上又无法进行分布式处理，因此有必要采用生成器（python data generator）来进行数据的分批次导入。虽然本项目经过数据简化后可以全部读入本机内存，但出于数据处理方式的一致性和学习目的考虑，对训练用的数据使用生成器来生成训练集和验证集（训练用样本 80% 划分为训练集，20% 划分为验证集）。对于测试集则直接读入内存。在使用生成器分批次从硬盘导入数据时，要对数据进行增加处理（具体代码见‘preprocess_data.py’文件中的‘generator()’函数）：

1. 首先对读入图片进行归一化（Normalization），将图片数据大小范围缩小到 [0.1, 0.9] 区间。
2. 对图片进行水平翻转，增加一倍数据。
3. 对图片加入随机噪音，增加一倍数据（在这里先归一化，再加噪音，否则会产生没有图像图片）。
4. 经过处理后，总的数据就是原始数据的 3 倍。

注意，本项目训练和验证都使用增加后的数据集（目的不是对比没有增加数据模型和增加数据模型的表现，在后面是要对比增加转向角度大的数据和没有增加转向角度大的数据模型之间的对比）。在前面“探索性可视化部分”提到，转向角度数据分布中转向角度大的数据较少，在具体执行过程中将随机增加这些数据（原样本大小为 24300，增加后为 40000），再使用生成器来分批次从硬盘导入数据生成训练集和验证集（总样本将达到 120000），以观察模型表现是否提升。

3.2 执行过程

3.2.1 基准模型改进

在基准模型 Nvidia model 表现的基础上，首先对其进行改进。在原始模型中，使用的激活函数为 RELU，在改进模型中，将使用 ELU[5]进行替换，因为 ReLU 的输出值没有负值，所以输出的均值会大于 0，当激活值的均值非 0 时，就会对下一层造成一个 bias，如果激活值之间不会相互抵消（即均值非 0），会导致下一层的激活单元有 bias shift。如此叠加，单元越多时，bias shift 就会越大。相比 RELU，ELU 可以取到负值，这让单元激活均值可以更接近 0，ELU 具有软饱和的特性，提升了对噪声的鲁棒性；在卷积层后增加 BatchNormalization 层和 MaxPooling2D 层。根据 keras 文档[8]，可知，BatchNormalization 层在每个 batch 上将前一层的激活值重新规范化，即使得其输出数据的均值接近 0，其标准差接近 1，MaxPooling2D 层为空域信号施加最大值池化。增加这两层的目的是避免过拟合和加快训练收敛速度；对每一层参数初始化方法改为‘he_normal’，根据 keras 文档[8]，可知，He 正态分布初始化方法，参数由 0 均值，标准差为 $\sqrt{2 / \text{fan_in}}$ 的正态分布产生，其中 fan_in 为权重张量的输入；在全连接层后使用 Dropout 层，根据 keras 文档[8]，可知，Dropout 将在训练过程中每次更新参数时按一定概率（rate）随机断开输入神经元，Dropout 层用于防止过拟合。修改后，模型结构如图 Figure 10：

构建好改进模型后，采用基准模型的训练方法（优化算法为 Adam optimizer），经过 10 次 epochs（设置生成

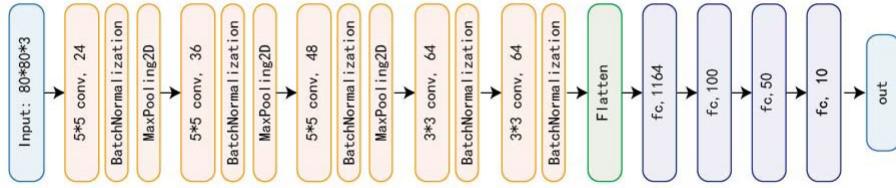


Figure 10: Nvidia 基准模型改进网络结构图

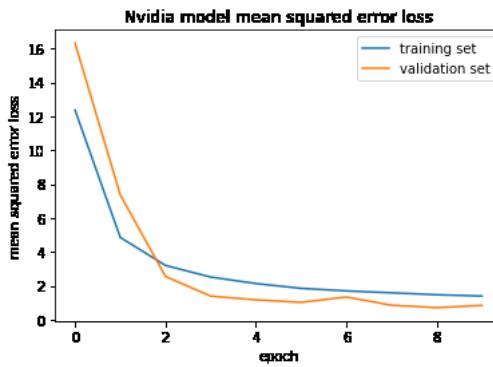


Figure 11: Nvidia 基准模型改进训练结果图

器的 batch size 为 128) 之后, 基准模型的 ‘training loss’ (MSE) 值为 1.4000, ‘validation loss’ (MSE) 值为 0.8484, 在测试集上 MSE 值为 4.4797。从损失值看, 相比基准模型, 训练集和验证集损失值增加, 测试集损失值降低, 说明模型的性能有所提升, 模型人工转向偏差相较基准模型更小。具体训练结果见图 Figure 11。

3.2.2 大转向角度数据添加

在前面 “探索性可视化部分” 提到, 转向角度数据分布中转向角度大的数据较少, 为了避免因数据少而使得模型对比较大的转弯反应 “迟钝”, 将随机增加转向角度小于 -5 或大于 5 的样本, 以对比增加前后增加后模型的表现是否有所提升。具体做法为: 首先读取样本的 CSV 文件, 然后提取转向角度小于 -5 或大于 5 的样本, 最后随机增加转向幅度大的样本到样本集中。这样, 总样本数从原来的 24300 增加到 40000 (增加了 15700 个转向角度大的样本)。划分训练集和验证集后 (20% 为验证集, 80% 为数据集), 使用生成器 (generator) 批次导入训练图片和验证图片, 期间使用数据增加技术将批次增加为原数据 3 倍。完成后, 总的训练用图片数达到 96000, 验证用图片数达到 24000, 数据量还是比较充分。

完成大转向角度数据添加后, 采用改进后的模型进行训练, 优化算法为 Adam optimizer, 经过 10 次 epochs (设置生成器的 batch size 为 128) 之后, 基准模型的 ‘training loss’ (MSE) 值为 1.8166, ‘validation loss’ (MSE) 值为 0.5717, 在测试集上 MSE 值为 3.2823。从损失值看, 训练集和验证集的损失值相较没有添加大转向角度数据的模型变化不大, 测试集损失值降低, 说明增加后对模型是有帮助的, 同时还需要结合时序图来分析对比预测转向和人工转向的差别。具体训练结果见图 Figure 12。

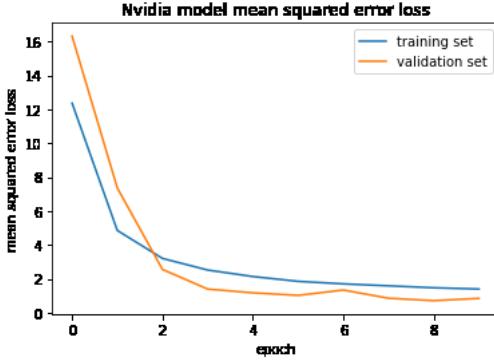


Figure 12: Nvidia 基准模型改进 + 大转向数据训练结果图

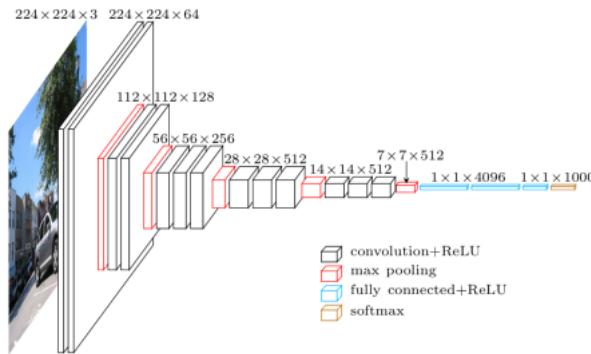


Figure 13: Vgg16 原始模型架构图

3.3 完善

3.3.1 构建

在完成 Nvidia model 模型（基准、改进、完善数据集）之后，将进一步使用迁移学习来完善整个项目。这里将选用 VGG16 + Nvidia 模型来作为 Nvidia model 的对比模型。

剑桥大学和 Google DeepMind 的研究人员在 2014 年论文[6]中提出的一种用于大规模图像识别的深度卷积神经网络，大体思想是用较小的卷积核（ 3×3 ）和较深的卷积层（16–19 层）对图像通用特征进行提取，VGG16 的网络结构图见 Figure 13，从图中可以看出，黑色的卷积层加上红色的池化层为一个 block，所以共有 5 个 block 作为特征提取，最后加上 3 个 FC 层。在实践中训练 VGG16 的数据集主要来自于 ImageNet。VGG16 可以在 ImageNet 这种涵盖多种类物体图像的大规模数据集上进行训练，对 ImageNet 中包括的景物有较强的识别能力，我们可以借鉴迁移学习的思想，利用 VGG16 来构建对比模型，设计的基本思想是使用 VGG16 来提取图像特征，然后加上 FC 全连接层来进行回归预测。

正如前面“算法和技术”章节提到将选用 fine-tuned 策略，fine-tuned 有两个使用方法：一是新的数据集比较大，而且与 pre-train model 的训练数据集比较接近，则直接使用 pre-train model 的参数权重是比较合理的，因为此时网络结构和设置不会在新的数据集上过拟合，而且采用原有网络的参数权重可以加速训练收敛；二



Figure 14: VGG16+Nvidia 模型结构图

是新的数据集比较小，而且与 pre-train model 的训练数据集差距比较大，此时应仅用顶部几层的结构和权重，最后几层提取的特征可能过于局限在原数据集。这两种方法在项目中都将尝试。另外，VGG16 初始模型来源于链接[9]，训练好的模型权重参数保存为.h5 文件中，分为 notop 和 withtop 两种，其中 notop 不包含原 VGG 结构的 3 个 FC 层，可以广泛应用在特征提取中，keras 使用的 VGG16 notop 的权重来源于链接[10]。最终 VGG16 + Nvidia 模型如图 Figure 14。

模型说明：在模型建构中使用了 GlobalAveragePooling2D 层代替了 Flatten 层，因为 $(80, 80, 3)$ 的输入图片在 block5 层输出后维度为 $(2, 2, 512)$ ，这里为空域信号施加全局平均值池化，可以输出一维向量，在实际训练中发现效果更好。另外参考基准模型，将中间 4 层 FC 层简化为 3 层，加快训练速度，同时在除最后输出层外其他 FC 层后加上 Dropout 层防止过拟合。

3.3.2 训练

VGG16 + Nvidia 模型训练按照前一小节提到的 fine-tuned 策略分为两种：一是 notop + noblocks。即直接使用预训练好的权重，之训练 FC 层权重参数；二是 notop + top2 blocks。即固定前 3 个 block 的参数权重，只训练微调 4, 5 个 block 的参数权重已经 FC 层权重参数。然后再加上转向角度大的数据对表现较好的方法模型进行训练。所以，总共是 3 种方法训练模型。结果如下：

1.notop + noblocks，采用和基准模型相同的训练方法和数据集（优化算法为 Adam optimizer, 10 次 epochs，原生成器生成的训练集验证集），将生成器的 batch size 设置为 64。训练结果发现模型的训练集和验证集的损失值都没有收敛，说明 tesla 的数据集和 VGG16 pre-train model 的训练数据集相差较大，所以这种方法不可取。

2.notop + top2 blocks，采用和基准模型相同的训练方法和数据集（优化算法为 Adam optimizer, 10 次 epochs，生成器生成的训练集验证集），将生成器的 batch size 设置为 64。模型的‘training loss’（MSE）值为 5.3162，‘validation loss’（MSE）值为 4.5624，在测试集上 MSE 值为 2.7826。具体训练结果见图 Figure15，虽然训练集和验证集损失值比较高，但在测试集上表现比 Nvidia 模型更好，说明模型学会了驾驶转向的一般

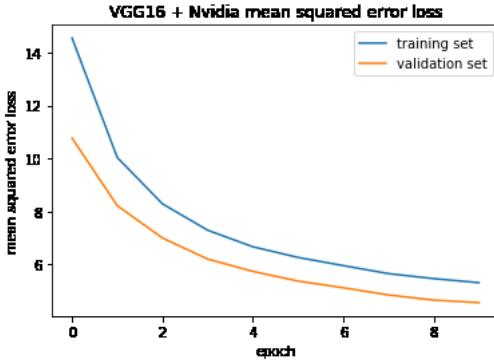


Figure 15: VGG16+Nvidia (notop+top2 bolocks) 训练结果图

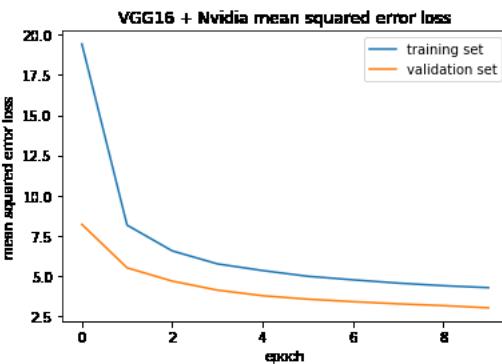


Figure 16: VGG16+Nvidia (notop+top2 bolocks+ 大转向角度数据增加) 训练结果图

策略。

3.notop + top2 bolocks + 大转向角度数据增加, 采用和“大转向角度数据添加”小节相同的训练方法和数据集(优化算法为 Adam optimizer, 10 次 epochs, 增加转向角度数据后的生成器生成的训练集验证集), 将生成器的 batch size 设置为 128。模型的‘training loss’(MSE) 值为 4.2552, ‘validation loss’(MSE) 值为 2.9988, 在测试集上 MSE 值为 3.5445。增加转向幅度大的 sample 后, 模型在验证集上有所提升, 但测试集损失值反而增加了。在这里并没有看出增加后对模型有帮助, 但还需要结合时序图来分析对比预测转向和人工转向的差别。具体训练结果见图 Figure 16。

IV. 结果

4.1 模型的评价与验证

在对上面基模型、基模型改进、基模型改进 + 大转向数据、VGG16+Nvidia (notop + top2 bolocks) 和 VGG16+Nviida (notop + top2 bolocks + 大转向角度数据增加) 经过多次训练后发现, VGG16 + Nvidia model 的训练结果曲线明显比 Nvidia model 的更平滑; 另外, 每次因对初始权重的随机设置, 训练集、验证集和测

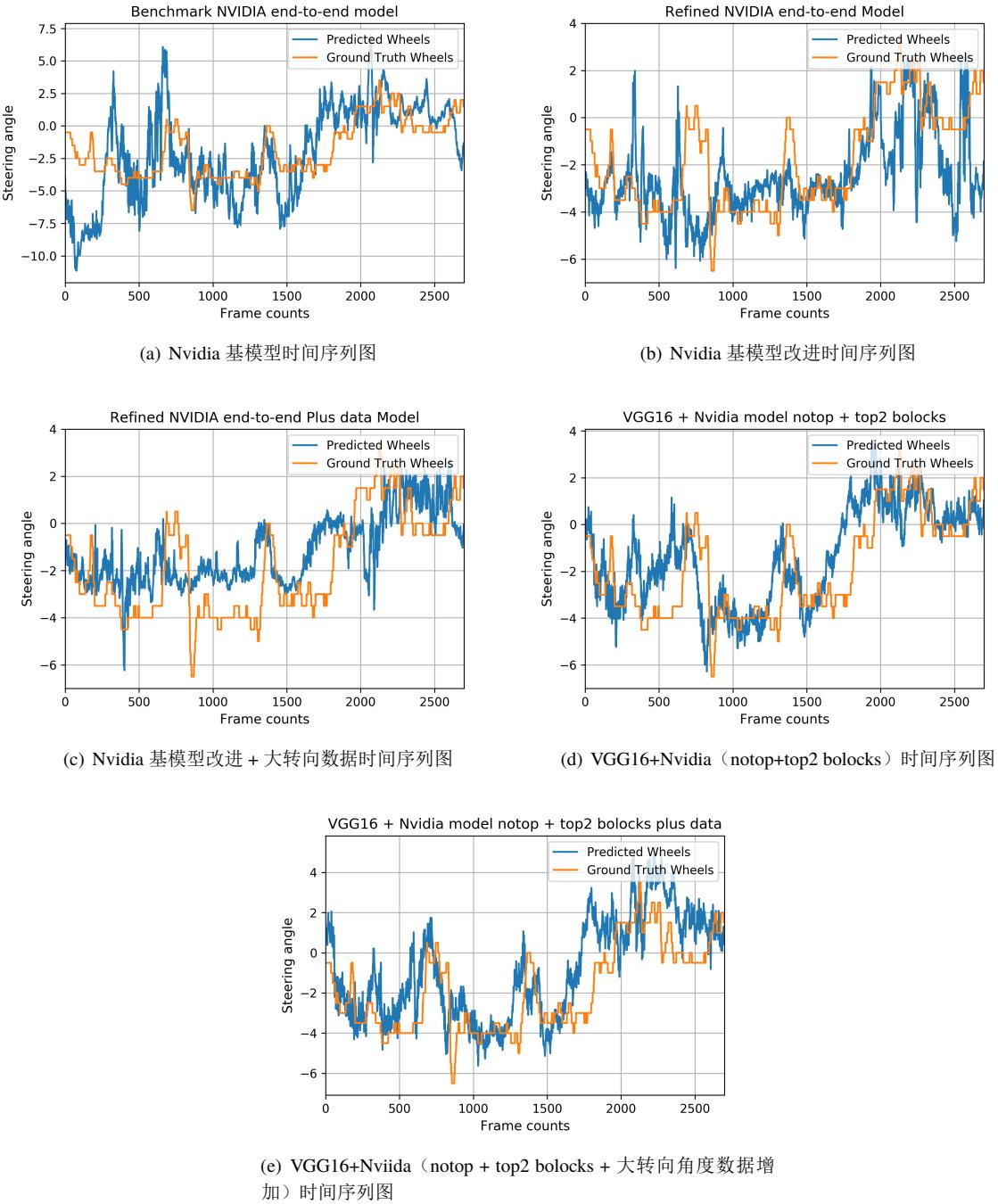


Figure 17: 各模型时间序列图对比

试集结果会有不同（但一般总体 MSE 值的差别不超过 5），大体来说 VGG16 + Nvidia model 表现是普遍好于 Nvidia model，增加大转向角度后模型表现是普遍好于不增加训练的数据。但这也从侧面说明一个问题，仅凭 MSE 作为模型表现判断标准是不完善的，因为一个智能的好的模型应该是总体预测走势接近于人工的，而有时一个模型 MSE 值小可能是因为对大的转向不敏感，或有时普遍准确的预测掩盖了某些时刻比较严重的错误。所以，在模型评价和最优模型的选择上，还要进行各个模型转向角度时间序列图的比较。如图 Figure 17。

从上面各模型的时间序列图可以看出，后两个 VGG16+Nvidia model 明显比前三个 Nvidia model 输出的角度更平滑和贴合原始方向角度，同时从前面各模型测试集的 MSE 值也可以看出后两个模型更出色。所以应该选用 VGG16+Nvidia model，虽然‘VGG16 + Nvidia model: notop + top2 bolocks’比‘VGG16 + Nvidia model: notop + top2 bolocks + 转向角度数据增加’输出的‘test loss’要小，但从图中可以看出，后者明显比前者的图形走势更接近于原始的数据走势，所以最终的模型应该为最后一个模型‘VGG16 + Nvidia model: notop + top2 bolocks + 转向角度数据增加’。

4.2 合理性分析

综上，将基准模型和最佳模型对比可以发现，基准模型在测试集的 MSE 值（8.8617）和转向角度时间序列图（预测值和真实值对比）上都比最终模型 MSE 值（3.5445）和时间序列图的表现要差。特别是在时间序列图上可以看出，基准模型在预测上是相当不准确的，而最终模型在预测上已经比较接近人工驾驶。所以，在项目开始时所要讨论的问题（比较准确的预测转向角度）得到了解决。

V. 项目结论

5.1 结果可视化

最后，将使用最终模型在第 10 段视频上生成转向角度预测视频，如图 Figure 18。这实际上就是将时间序列图（人工和模型预测的角度对比）和驾驶视频结合起来，前面已经讨论了时间序列图的相关结果（注意：图中红线表示预测值，蓝线表示真实值，这和上面时间序列图有所不同），这里不再赘述。

5.2 对项目的思考

本项目大致可以分为四个阶段：首先、进行了简单的数据探索（视频和 CSV 文件）；二、数据预处理。包括所有视频转换为图片保存到硬盘，使用生成器（generator）从硬盘读取图片生成训练集和验证集（测试集保存到内存），期间使用数据简化（裁剪、缩小图片尺寸）和数据增加（水平翻转、图片增加噪音）技术；三、利用 Keras 训练模型并导出模型参数。完成了基模型（Nvidia model）、基模型改进（NVIDIA refined model）、基模型改进 + 大转向数据、VGG16+Nvidia（notop + top2 bolocks）和 VGG16+Nviida（notop + top2 bolocks + 大转向角度数据增加）5 个模型的训练。然后讨论了各模型的 MSE 值和时间序列图表现。四、选择最佳模型并生成最终视频。选择 VGG16+Nviida（notop + top2 bolocks + 大转向角度数据增加）模型为最终模型，并使用最终模型在第 10 段视频上生成了结果视频。

另外，在项目中完成了两个选做模型。一个是 VAE + GAN model 生成训练用图片，另一个是 CNN+RNN seq2seq model 来预测转向角度。这两个选做项目难度较大而且完成度还有待提高，每一个设计的技术和方法



Figure 18: 最终测试视频生成截图

都可以单独用一篇文章进行总结，所以在这里并没有对它们进行讨论，留待后续探讨。

项目中模型架构，超参数的调试都能体会到难度和兴趣点（为了使整个项目报告简洁，项目中很多参数调试过程并没有呈现：如不同 epochs 训练下模型表现、个别网络层添加删除后效果、不同学习率调试的效果，不同优化算法的尝试等。呈现的都是最终调试好的结果）。选做项目是整个项目最难的地方，但收获也比较多。总体来说，最终模型和结果达到预期目标，其中的涉及到的技术和算法在类似的场景中也有一定的通用性。

5.3 需要作出的改进

需要改进的地方有：

1. 在数据预处理阶段，尝试更多的图片处理技术（如平移、加阴影等）。
2. 在迁移学习中可以使用其它更好的原始模型，如 ResNet 50。
3. 对选做项目中涉及的模型还可以进一步优化（使用更完善的 LSTM 神经网络）。
4. 使用更多的训练集（如百度阿波罗平台的数据）对模型进行训练。

VII. 参考文献

[1]: 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.

[2]: E. Santana and G. Hotz, “Learning a driving simulator,” arXiv preprint arXiv:1608.01230, 2016.

[3]: Udacity. An open source self-driving car, 2017: <https://medium.com/udacity/teaching-a-machine-to-steer-a-car-d73217f2492c>

[4]: D. A. Pomerleau, “Alvinn, an autonomous land vehicle in a neural network,” Carnegie Mellon University, Computer Science Department, Tech. Rep., 1989.

[5]: A. Karpathy, “Cs231n convolutional neural networks for visual recognition,” <http://cs231n.github.io/convolutional-networks/>, 2016.

[6]: K Simonyan, A Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv preprint arXiv:1409.1556, 2015.

[7]: Diederik P. Kingma, Jimmy Ba, “Adam: A Method for Stochastic Optimization,” arXiv:1412.6980, 2014.

[8]: Keras 中文文档: <http://keras-cn.readthedocs.io/en/latest/>

[9]: VGG16 初始模型的.py 文件: <https://github.com/fchollet/deep-learning-models/blob/master/vgg16.py>

[10]: WEIGHTS PATH NO TOP: https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5