无人驾驶技术入门（十七）
深度学习进阶之无人车行为克隆

陈 光

上一期的《无人驾驶技术入门（十六）| 初识深度学习之交通标志分类》我以交通标志牌的分类为例，介绍了深度学习中所涉及的有关神经网络的理论知识。包括神经网络中的参数，反向传播原理，训练集、验证集和测试集的区别，通过优化经典的图像分类网络 LeNet-5，完成了交通标志牌的分类工作。

在本次分享中，我将以优达学城(Udacity)无人驾驶工程师学位中提供的无人车行为克隆项目为例，介绍深度学习在端到端无人车控制方面的应用。所谓端到端是指图像端到控制端，即将摄像机所拍摄的图像结果输入到神经网络中，网络直接输出无人车方向盘转角、油门、刹车踏板开度等信息的技术。

使用深度学习进行“端到端”的无人车行为克隆时首先会遇到缺少数据问题。目前 CV 领域的数据集大部分是在车上采集的，并做了标记，但是对应的数据集中并不包含汽车控制相关的数据，比如方向盘转角、车速等。

幸运的是 MIT 6.S094 这门公开课提供了部分特斯拉行驶数据集，数据集包含了特斯拉前置相机的图像和对应每帧图像的方向盘转角。虽然数据有限，至少我们可以着手尝试搭建模型，实现端到端的无人车控制。@优达学城(Udacity)曾发布了一篇文章《无人驾驶技术初探——让特斯拉(Tesla)学习人类的驾驶技术》，就是使用该数据集实现了“端到端”的方向盘转角预测，数据和代码可以在这里看到。

方向盘转角预测的结果如下图右下角所示，蓝色线为汽车方向盘转角的真值，粉色线为根据当前图像给出的方向盘转角预测值。

图片出处：https://github.com/nd009/capstone/tree/master/deep_tesla

特斯拉数据集有两大不足之处。一、数据量较少，训练出的网络模型不够泛化。二、预测了方向盘转角后，仅能实现预测值和真值的简单对比，并不会作为车辆底盘的输入去控制车辆，也就是业内常说的不能够实现控制的“闭环”。

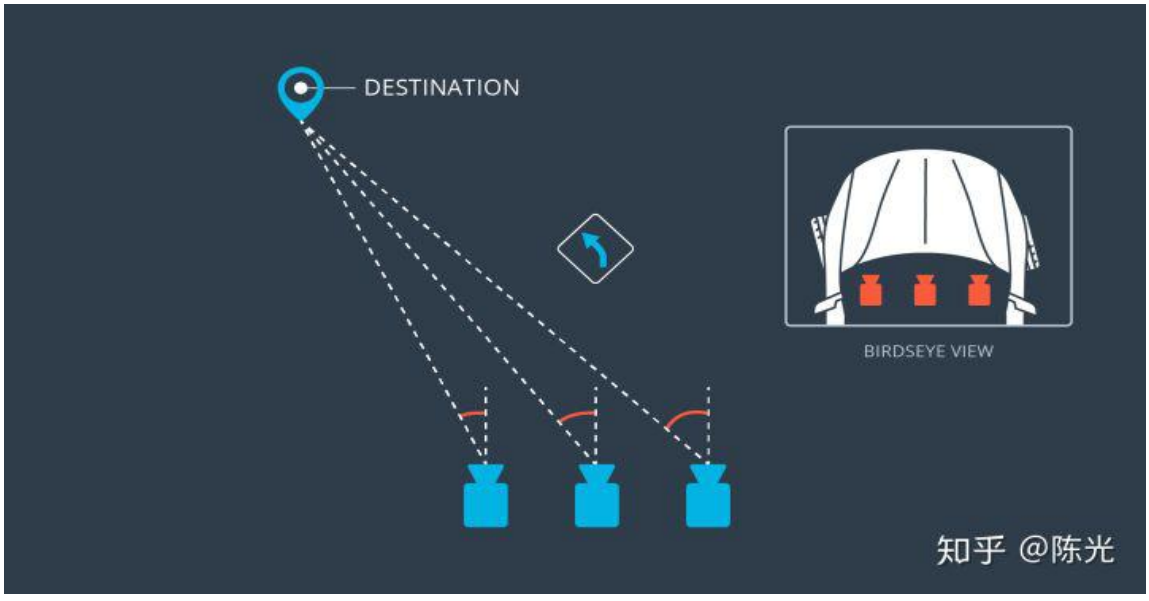
1 无人驾驶模拟器

为了解决端到端“闭环”的问题，优达学城(Udacity)无人驾驶工程师学位的老师设计了一款支持多平台(Mac、Linux、Windows)的模拟器 self-driving-car-sim，这款模拟器不仅能够实现数据的采集，还能够通过导入训练好的深度学习模型实现端到端的无人车转向控制。

模拟器 self-driving-car-sim 包含两种模式——训练模式(Training Mode)和自动驾驶模式(Autonomous Mode)。

训练模式也可以理解为数据采集模式。点击图中的记录按钮，使用上下左右按键即可实现汽车的行驶，控制汽车行驶一段路，再次点击后，模拟器就会将刚才人工驾驶的这段路的数据进行记录，并存储。

存储的数据包含一个表格，表格中每一行记录了车载相机所拍摄到图像的存储路径和拍摄时刻的车辆的控制信号。车上有三个车载相机，分别安装在汽车的左侧、正中心和右侧，如下图所示。



图片出处：优达学城(Udacity)无人驾驶工程师学位

每一组（帧）数据将包含三个视角的图像、方向盘转角和车速。如下图所示为三个视角拍摄到的道路图像，可以通过汽车引擎盖的位置判断摄像头的位置。

2 数据收集与预处理

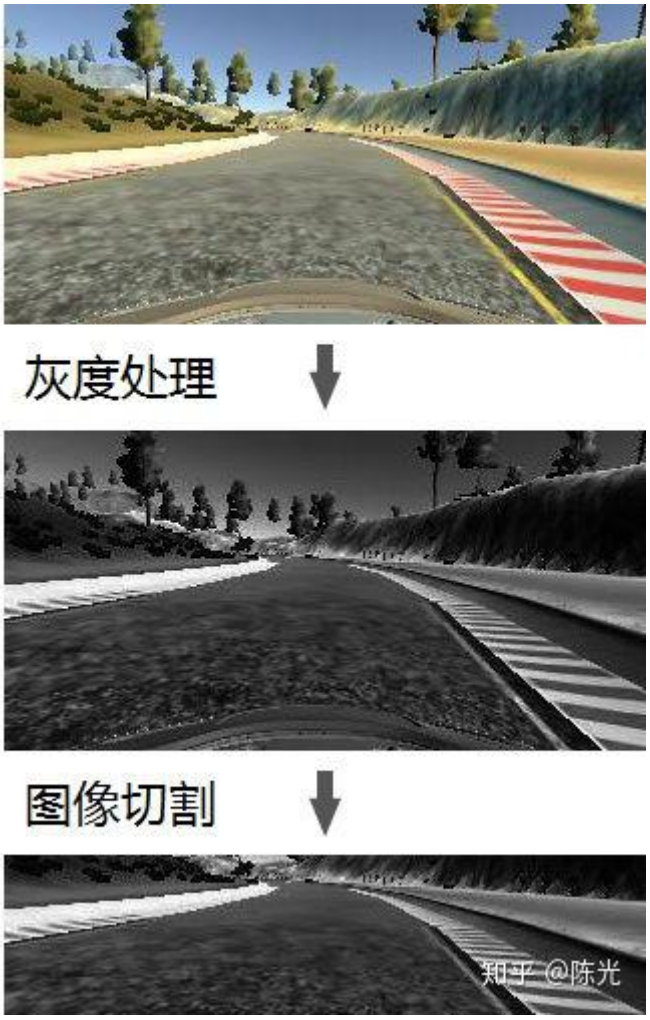
模拟器提供的路线是一个圈，控制车辆行驶时，大部分的路段都是朝左转向的。如果只录制朝一个方向转向的数据就会导致训练出的模型遇到任何路况都会倾向于向左转弯，显然这种模型并不是我们想要的。为了避免出现上述情况，使训练出的模型更加泛化，需要同时记录顺时针和逆时针的行驶数据。

除了同时记录顺时针和逆时针的方式外，还可以使用上一篇提到的数据增广技术增加数据的丰富度。在这个项目中最直接的数据增广方式就是将图像左右对称，随后将图像所对应的方向盘转角加一个负号。这样就能够将数据量增倍。

以上都是针对安装与车辆正中间摄像头的数据收集，其实位于两侧的摄像头的数据也能够提供很多帮助。其一，使用左右两侧的数据可以将数据量再增加两倍；其二，由于两侧的数据更靠近道路外侧，它们可以教会网络在车辆开始向路边缘偏移时如何转向以回到道路中央，我们只需要在使用时，给对应的方向盘转角加上或减去一个修正量，就可以使用这部分数据了。

经过以上一系列的处理，数据集就变为了最初的 6 倍。接下来我们对数据进行一定的预处理操作。

观察图像可以看出，图像的颜色信息并不会对汽车的转向造成很大影响。即使是灰度图，无人车依旧可以正常驾驶。因此，我们将彩色图像数据进行灰度处理，以此来降低模型输入图像的复杂度。影响车辆方向盘转角的是道路形状，因此图像中的天空、树木部分并不是模型关注的重点，可以将该天空、树木部分进行切割，使图像尽可能地被道路占据。图像的预处理操作如下图所示：



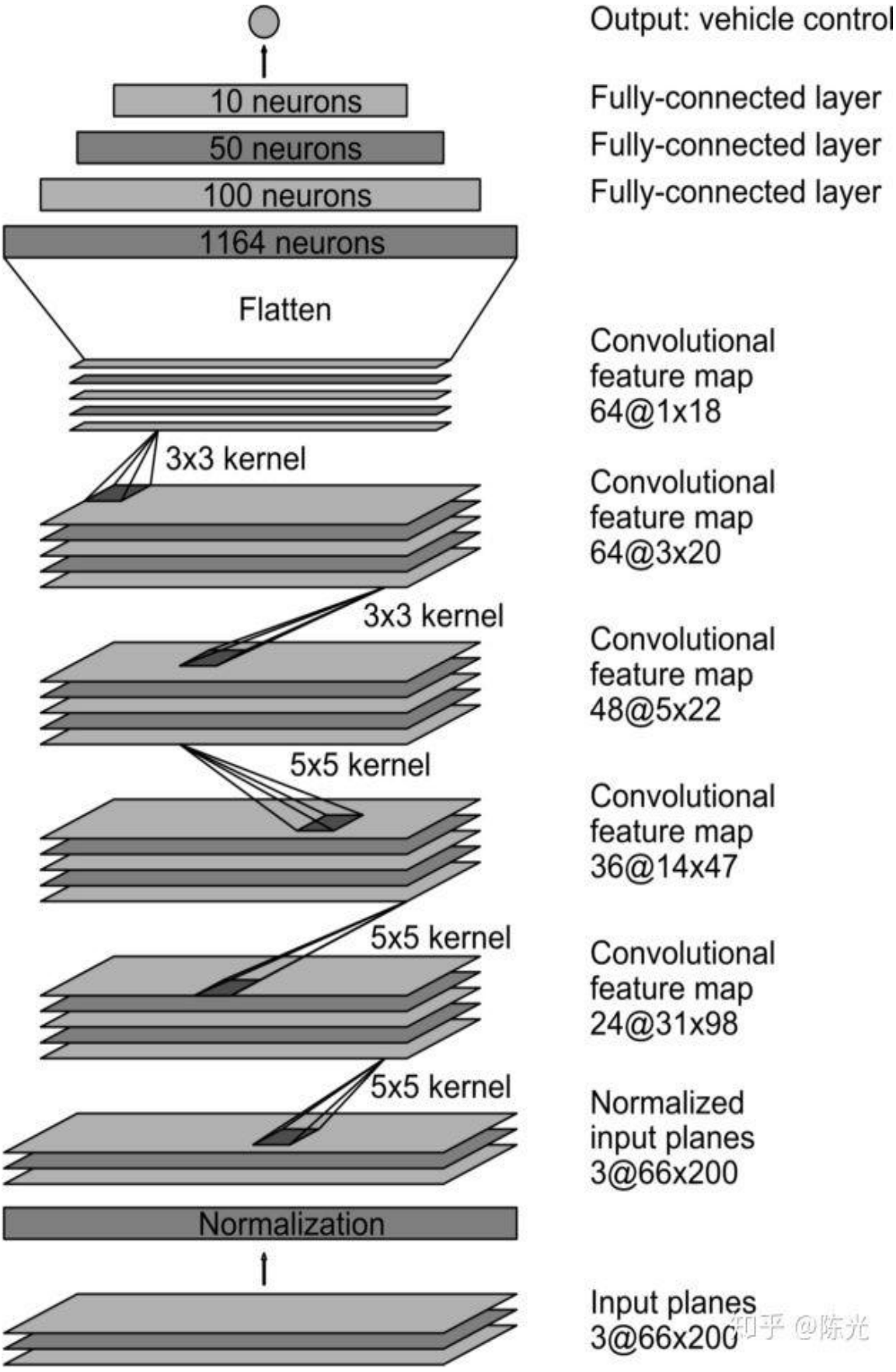
完成数据收集和预处理工作后，就开始搭建端到端的网络模型吧。

3 端到端网络模型

在深度学习领域，大部分情况下，研究人员不是从头开始搭建神经网络的。从头搭建神经网络，不仅需要设计网络的架构，还需要进行大量的训练、测试以证明网络有效，这会非常耗费精力和时间。为了加快进度研究人员通常会从已经训练过的网络着手，对其进行修正和微调。这种对成熟网络的改造被称作”迁移学习“。

用通俗的话说就是借鉴他人的网络来解决类似的问题。

英伟达的无人车团队曾在 2016 年 8 月提出过一个专门用于实现端到端无人驾驶的网络，这个网络也是他们进行实车训练时所使用的网络，这样的网络正是我们可以借鉴的。模型结构如下所示：



图片出处：https://devblogs.nvidia.com/deep-learning-self-driving-cars/

从下往上，网络包含一个归一化层，归一化后接了 5 个卷积层，再连接了 4 个全连接层，最终输出了车辆的控制量，这个控制量可以是方向盘转角，也可以是车速。

使用 tensorflow 的 keras 库能够很快地搭建出这个网络，如下所示：

```
from keras.models import Sequential, Model
from keras.layers import Lambda, Input, Flatten, Dense, Cropping2D, Convolution2D

model = Sequential()
# normalize image
model.add(Lambda(lambda x: x / 127.5 - 1.0, input_shape=(66, 200, 3)))
# convolution layer
model.add(Convolution2D(24,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(36,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(48,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(64,3,3, activation="relu"))
model.add(Convolution2D(64,3,3, activation="relu"))
# flatten layer
model.add(Flatten())
model.add(Dense(100))
model.add(Dense(50))
model.add(Dense(10))
model.add(Dense(1))
```

由于我们的图像大小与模型所使用的图像大小不同，同时对图像数据进行了预处理（灰度处理和切割），因此需要对模型进行一些修改。修改后的模型如下：

```
from keras.models import Sequential, Model
from keras.layers import Lambda, Input, Flatten, Dense, Cropping2D, Convolution2D

input_shape = (160, 320, 3)
ouput_shape = (160, 320, 1)
normalized_shape = (80, 320, 1)

model = Sequential()
# user defined grayscale function
model.add(Lambda(color2gray, input_shape = input_shape, output_shape= ouput_shape))
# cropping
model.add(Cropping2D(cropping=((50,30), (0,0))))
# normalize image
model.add(Lambda(lambda x: x / 127.5 - 1.0, input_shape=normalized_shape))
# convolution layer
model.add(Convolution2D(24,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(36,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(48,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(64,3,3, activation="relu"))
```

```
model.add(Convolution2D(64,3,3, activation="relu"))
# flatten layer
model.add(Flatten())
model.add(Dense(100))
model.add(Dense(50))
model.add(Dense(10))
model.add(Dense(1))
```

搭建好模型后，我们将之前的数据选出 80%作为训练集，另外 20%作为验证集开始训练模型。

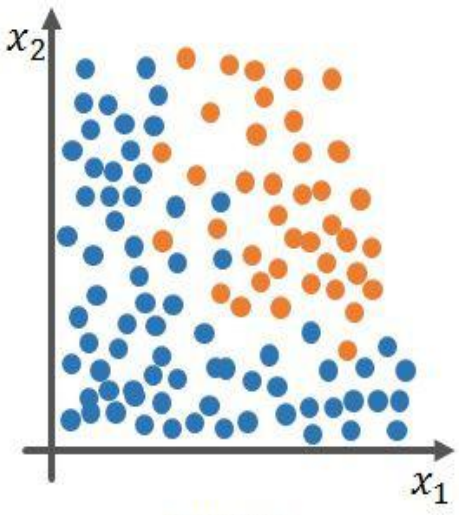
4 过拟合问题

使用以上数据和网络训练模型后，将模型应用到模拟器中让汽车自动驾驶时会发现，汽车在运动一段时间后会冲出了跑道。此时，我们遇到了一个深度学习领域经常遇到的问题——过拟合。

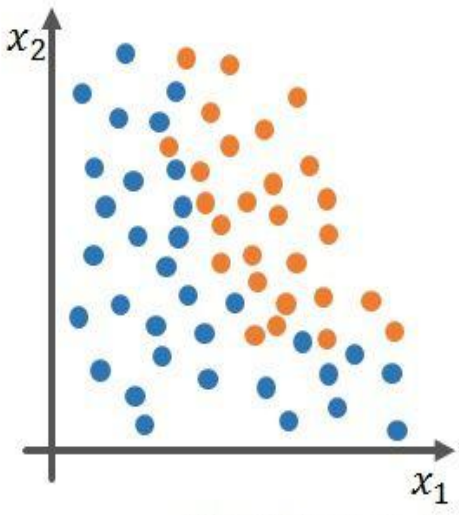
过拟合的模型最明显的特征是它在训练集上表现地很完美，但在验证集上表现地很差。如果模型在验证集上的表现很差，那么将模型应用于实际场景中时，也不会有太好的表现。

接下来，我用一个简单的例子来解释过拟合现象，并谈一下面对模型过拟合时该如何解决。

假设我们的训练集和验证集如下蓝色的点和橙色的点，需要训练一个能够区分不同颜色点的模型。

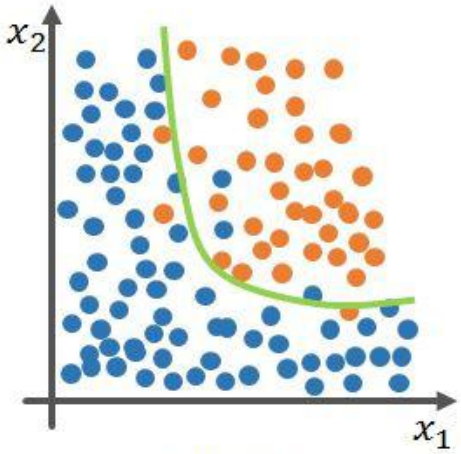


训练集

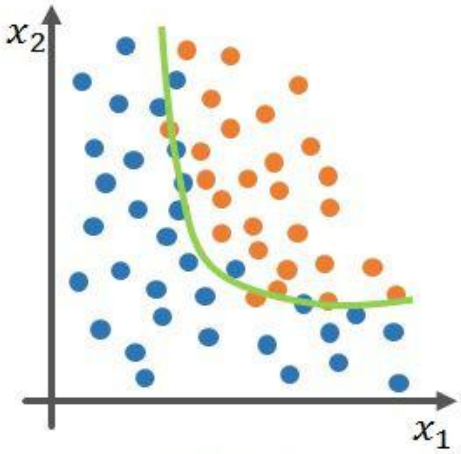


验证集

使用训练集训练出的理想模型应该既能在训练集中实现较高准确率的分类效果，同样应用于验证集时，也有不错的表现，如下图的绿线所示。

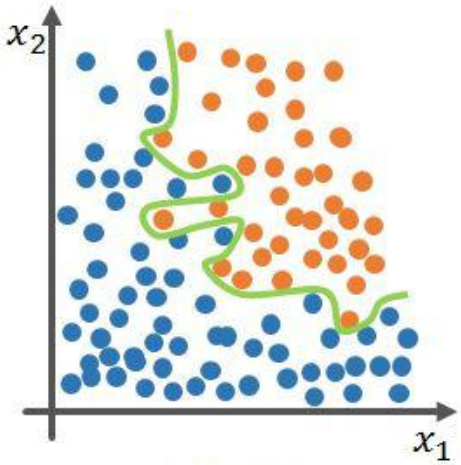


训练集

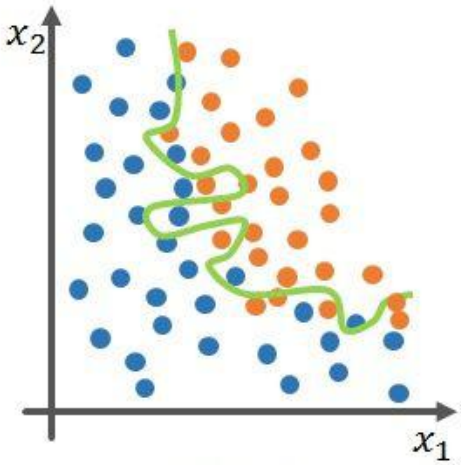


验证集

而过拟合的模型的会有一个明显的现象，那就是在训练集中能够得到极高的准确率，但一旦将该模型应用于验证集时，效果就很差，如下图绿线所示，就是一个典型的过拟合模型。过拟合的模型在训练集中表现很好，但这种表现不够泛化，导致了应用到验证集中准确率低的问题。



训练集



验证集

针对端到端的无人车控制模型的过拟合问题，优达学城的老师给出了四种解决方案：

1. 在网络中加入 Dropout 层
2. 在网络中加入 Pooling 层
3. 使用更少的卷积或全连接层

4. 收集更多数据或使用数据增广丰富数据集

为了避免模型修改后与原模型差异太大，我并未修改卷积部分的网络，而是在网络中加入 Dropout 层和微调全连接层的方案以减小过拟合。

经过多次尝试，找到了一种能够满足项目要求的网络模型。修改后的模型如下：

```
from keras.models import Sequential, Model
from keras.layers import Lambda, Input, Flatten, Dense, Cropping2D, Convolution2D

input_shape = (160, 320, 3)
ouput_shape = (160, 320, 1)
normalized_shape = (80, 320, 1)

model = Sequential()
# user defined grayscale function
model.add(Lambda(color2gray, input_shape = input_shape, output_shape= ouput_shape))
# cropping
model.add(Cropping2D(cropping=((50,30), (0,0))))
# normalize image
model.add(Lambda(lambda x: x / 127.5 - 1.0, input_shape=normalized_shape))
# convolution layer
model.add(Convolution2D(24,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(36,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(48,5,5,strides=(2,2), activation="relu"))
model.add(Convolution2D(64,3,3, activation="relu"))
model.add(Convolution2D(64,3,3, activation="relu"))
```

```
# flatten layer
model.add(Flatten())
model.add(Dropout(0.5)) # reduce overfitting
model.add(Dense(180, activation="relu"))
model.add(Dense(60))
model.add(Dense(10, activation="relu"))
model.add(Dense(1))
```

5 效果视频

使用数据增广后的训练集及修改后的网络训练出最终的模型后，按照无人车行为克隆项目中介绍的方法运行模型，并进入模拟器的自动驾驶模式。

来看一下最终的无人车端到端自动驾驶的效果。

视频地址见原文：https://zhuanlan.zhihu.com/p/60625133

从最终的“闭环”效果可以看出，无人车确实学会了端到端的自动驾驶。有些地方会有比较明显的转向延迟，可能需要更多的数据或更优的模型来解决这些问题。

6 总结

以上就是《无人驾驶技术入门（十七）——深度学习进阶之无人车行为克隆》的全部内容。在这次分享中，我介绍了 Udacity 提供的无人车模拟器 self-driving-car-sim，使用模拟器完成了数据采集，随后对采集的数据做增广和预处理；在对英伟达无人车团队的提出的端到端网络进行了修改后训练出模型完成了项目要求。

通过这次分享，可以看出，在拥有足够多的数据后，深度学习不仅能够解决交通标志牌的分类问题，还可以用于处理无人车行为克隆这类场景更为复杂的问题。

不过话说回来，端到端的行为克隆技术目前还仅仅停留在实验室阶段，实际路测中应用还不是很广泛。这是因为，任何涉及无人车控制层的决策必须是可追溯原因的，而深度学习作为不可追溯的“黑盒”即使在实验室环境下表现良好，工程师们也不敢贸然在路测中使用。

好了\(^o^)/~，这篇分享就到这啦，我们下期见~

本文原载：知乎号“陈光”，作者授权转载。

临菲信息技术港

临菲信息技术港公众号

临菲学堂