Self-driving cars program - project 4: Behavioural Cloning

Francesco Boi

Contents

1 Content of the project

Here is the content of the project:

- *writeup.pdf* (this file): report of the project;
- *writeup.tex*: source tex file;
- *model.py*: Python file to train the Convolutional Neural Network;
- *model.h5*: trained Keras model;
- *P4Data.tar.gz*: archive containing data to train the model;
- *P4OtherData.tar.gz*: archive containing additional training data;
- *drive.py*: Python script to autonomously drive the vehicle on the simulator using the trained neural network and to generate the images for the output video;
- *run1.tar.gz*: archive containing the images output by *drive.py* to generate the output file;
- *video.py*: Python script to generate the output video required by the project using the images in *run1.tar.gz*;

- *video.mp4*: output video of the project;
- *merge_data.py*: Python script to merge dataset recorded in different training sessions of the simulator;
- *README.md*: file giving a general description of the project and instructions to set a virtual environment and how to run the script.

2 Project goals

The steps of this project are the following:

- use the simulator to collect data of good driving behaviour;
- build, a convolution neural network in Keras that predicts steering angles from images;
- train and validate the model with a training and validation set;
- test that the model successfully drives around track one without leaving the road;
- summarize the results with a written report.

3 Instruction

Instructions to run the scripts are given in the README.md file.

4 Data collection

Initially both tracks were used to train the model, driving the simulator both clockwise and anti-clockwise. Furthermore, the simulator was driven to recover the centre of the track when offroad. Unfortunately, the huge amount of data meant the training was extremely slow, especially due to the generator use. For this reason, new data were collected by driving the car around the track for three laps only.

5 Model architecture

The model architecture is very similar to the Nvidia model with some changes. Particularly, dropout layers were added to the dense layers with 100 and 50 neurons with a dropping probability of 0.25. No dropout layer was added to the last but one layer with 10 neurons since it has much less units than the previous layers. Unlike the model in the lectures, all the dense layer except the output one have a ReLU activation function: having multiple dense layers without an activation is equivalent to having a single dense layer due to the linear operations

Neural network layers					
Layer	Input	Output	Stride	Pad	Act.
Lambda	(160, 320, 3)	(160, 320, 3)	/	/	/
Crop(50, 20, 0, 0)	(160, 320, 3)	(90, 320, 3)	/	/	/
Conv2D(5, 5, 24)	(90, 320, 3)	(43, 158, 24)	(2, 2)	def.	relu
Conv2D(5, 5, 36)	(43, 158, 24)	(20, 77, 36)	(2, 2)	def.	relu
Conv2D(5, 5, 48)	(20, 77, 36)	(8,37,48)	(2, 2)	def.	relu
Conv2D (3, 3, 64)	(8, 37, 48)	(6, 35, 64)	(1, 1)	def.	relu
Conv2D (3, 3, 64)	(6, 35, 64)	(4,33,64)	(1, 1)	def.	relu
Flatten	(4,33,64)	(8448)	/	/	relu
Dense(100)	(8448)	(100)	/	/	relu
Dropout(0.25)	(100)	(100)	/	/	/
Dense(50)	(100)	(50)	/	/	relu
Dropout(0.25)	(50)	(50)	/	/	/
Dense(10)	(50)	(10)	/	/	relu
Dense(1)	(10)	(1)	/	/	none

involved. The output layer has no activation: since the steering is between -1 and 1, some experiments were carried out with a *tanh* but it did not improve the situation.

Here is the final architecture: Figure 1 produced with `keras.utils.plot_model` shows the architecture graphically.

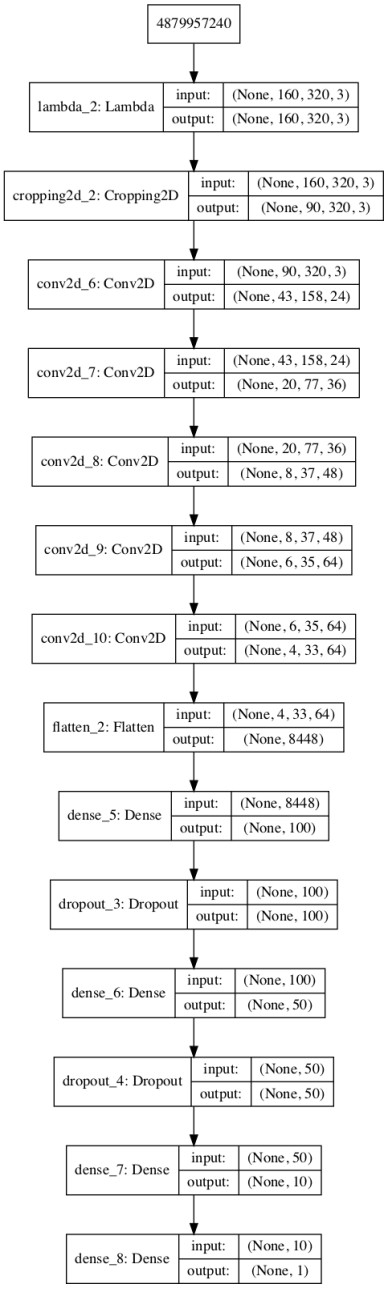

Figure 1: Graphical representation of the model with input and output shapes.