Advanced Self Driving – Lane Detection

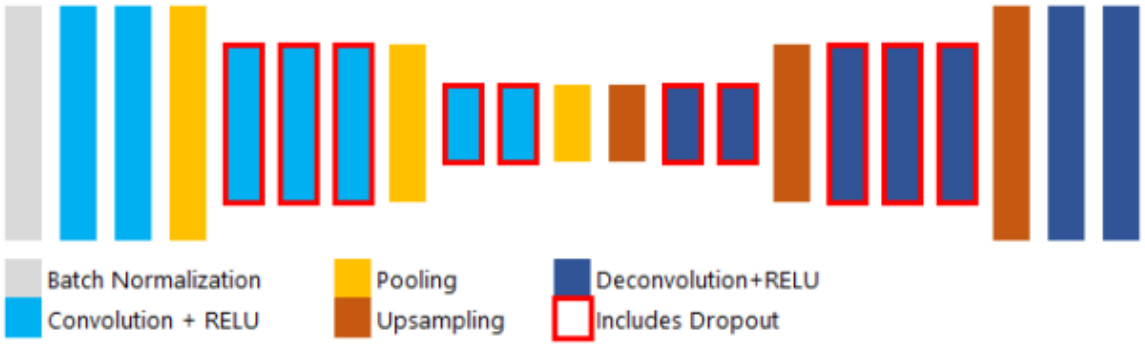
Self-driving cars have piqued human interest for centuries. Leonardo Da Vinci sketched out the plans for a hypothetical self-driving cart in the late 1400s, and mechanical autopilots for airplanes emerged in the 1930s. In the 1960s an autonomous vehicle was developed as a possible moon rover for the Apollo astronauts. A true self-driving car has remained elusive until recently. Technological advancements in global positioning (GPS), digital mapping, computing power, and sensor systems have finally made it a reality.

In this project, we explore a crucial component of a self-driving car, that is, lane detection. One of the first components of a self-driving cars' pipeline is lane detection. Lane detection is concerned with finding lanes on the road so that each car knows where exactly it should be driving. It can also be useful for finding other information such as road curvature, if and when a car is getting out of the lane, and other important features.

In this project, we'll train a neural network that paints the lane part of the road as green. To do so, there are two common approaches. First, to detect the lane lines, fit a polynomial to these lanes, and, finally, to just paint the area between these two polynomials as green. This is not the scope of this project however. So, we'll train a convolutional neural network that learns how to generate an image with this area painted. You can see video [baseline.mp4] for a clear understanding. You'll be provided with a test video to test your pipeline. You should use your model to detect lanes in this video (use the helper script)

To train a neural network that could generate an image, you should explore transposed convolutional layer (deconvolutional layer)

The architecture of your neural network should be something similar to this.



You are expected to do hyperparameter tuning on the architecture of the neural network to get the best performance. You should also get a performance that is better than what is given in the baseline video.

After the training is done, you should call draw_lanes.py script, and pass it the saved model as an argument to convert the test video using the provided script. You do analyse the results, identify the weak points in the baseline video, solve some of these issues, and, finally, make a good write up on how you improved this method.

Provided Files

- Model-Training.ipynb: used for training the model (you just have to fill in the "# TO DO")
- Draw_lanes.py: used for generating the video
- Baseline.mp4: Baseline video. You are expected to identify the problems in this video and get a better performance.

Dataset

You can download the full training set of images used here and the full set of 'labels' (which are just the 'G' channel from an RGB image of a re-drawn lane with an extra dimension added to make use in Keras easier) here (157 MB).