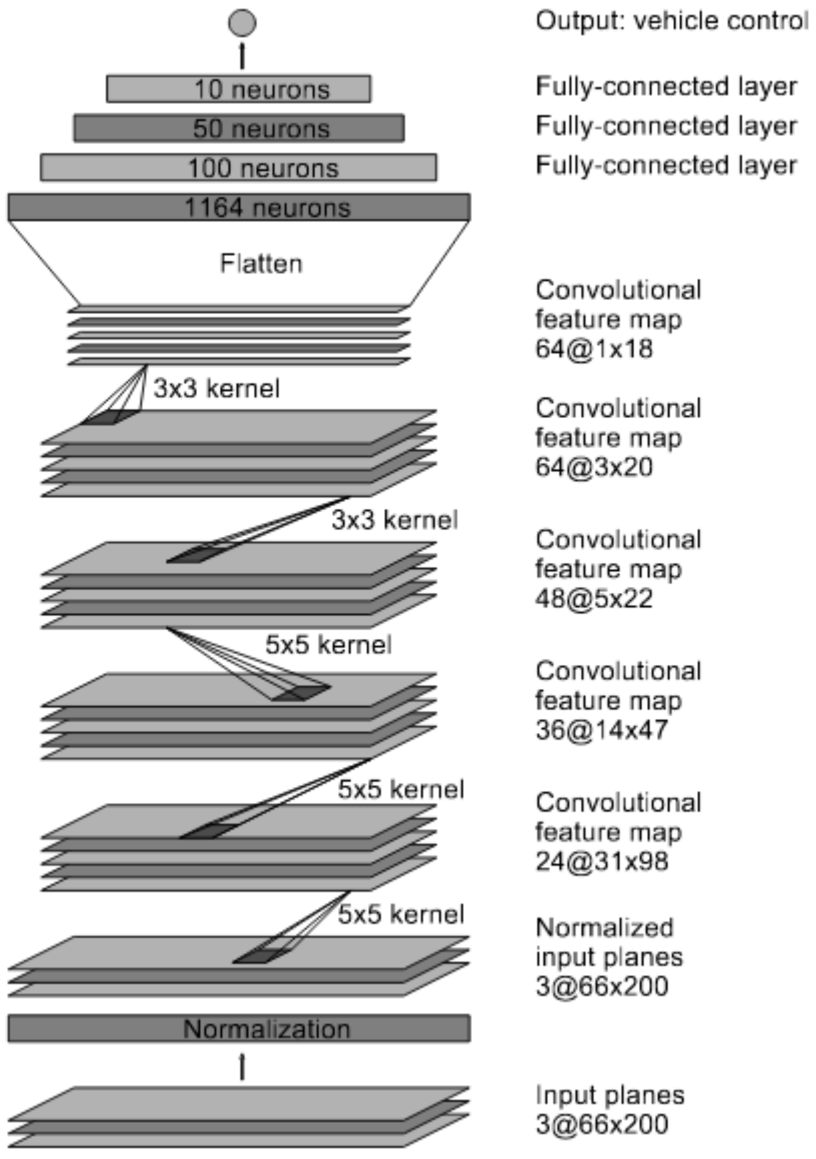
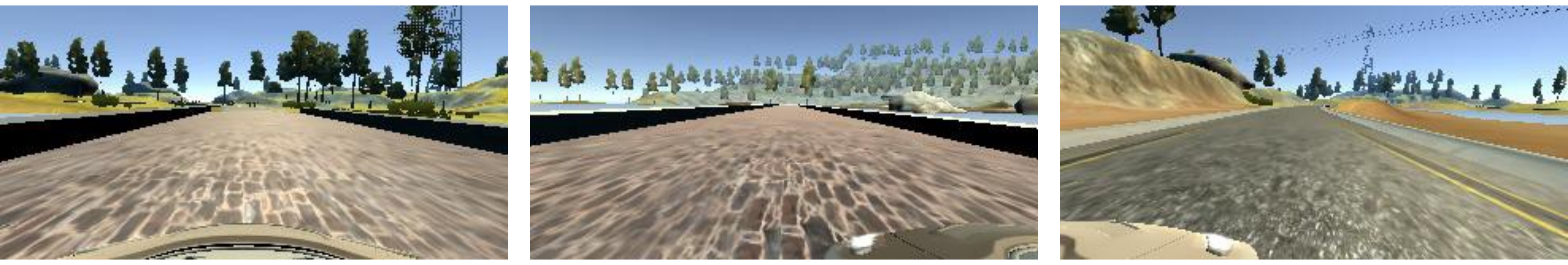
Dmytro Nasyrov
CTO @ Pharos Production Inc. — Software development outsourcing company. Give me a mes...
Aug 21 · 4 min read

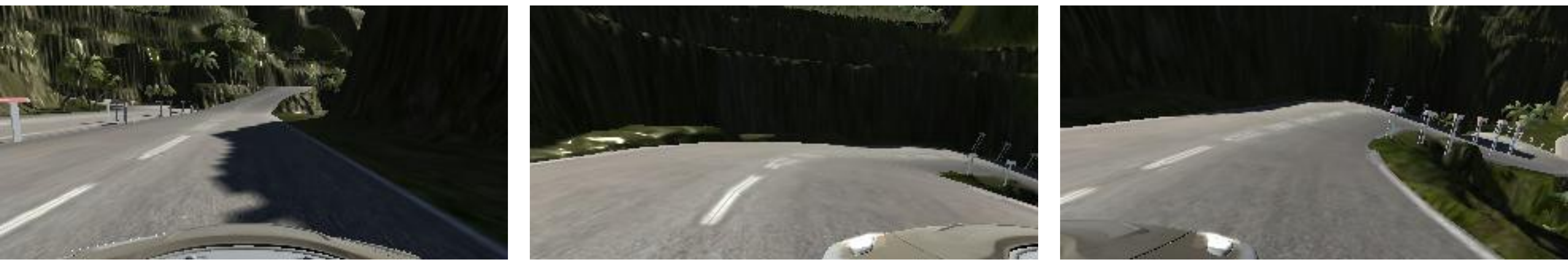
Behavioral Cloning. NVidia Neural Network in Action.



NVidia Convolutional Neural Network

This is a writeup on Project 3 from Udacity course Self Driving Car Engineer. This time we will talk about Behavioral Cloning. Our goal is to use manually collected image data to teach the car to stearing left and right based on a conditions around. We have a simulator created with Unity, we can drive a car on two different tracks like in Need for Speed in 1999. Car has 3 cameras on board—left, right and center camera. Cameras snapshoting images of the road. We will use this images to train our neural network.


center | left | right cameras on track 1


center | left | right cameras on track 2

To collect more data from a single track we have drive the car in both directions of the track. This should generalize the prediction of the model. Also we can add image augmentation to simulate shadows and bright highlights—different environment—but in future.

. . .

We have 3 options for the network. We can create it from the scratch and pray to make it work, we can use NVidia neural network (see image above), and we can use Comma.ai neural network. Our first approach was to try to make a neural network by ourself. That approach sucked after 2 weeks of tries. We have chosen Nvidia's solution.

You can find much more about this DNN architecture here:

End-to-End Deep Learning for Self-Driving Cars

In a new automotive application, we have used convolutional neural networks (CNNs) to map the raw pixels from a front…

devblogs.nvidia.com

All about comma.ai you can find here:

https://github.com/commaai/research

Input is a 3 channels image with 200 width and 66 height. Images from camera have a different resolution. So we need to prepare them to make it work. First we crop them to the road range to avoid learning from the sky and trees. We can blur image just a little to make pixelated road lane smoother. Also let's convert the image to YUV from RGB. Also we need to analyze and prepare the data to avoid biased result, because we have a lot of straight drive. We will use data from both tracks of the simulator.

Original image

Scaled to 200x66

Blurred

In YUV colorspace

. . .

For the framework we choose Keras to simplify our life with a Tensorflow backend. First layer is a normalization to -0.5–0.5 from 0–255. Network scheme is presented above, for the activation layer we will use ELU to make prediction smoother. Before the flatten layer we add dropout. Then we have a flatten layer and 3 fully connected layers. To save RAM we will use batch generator. That’s all! Now we will run training for tens epochs and check the result.

What we can improve here? Probably it’s a good idea to play with different color spaces combinations and use convolutional blur instead of plain Gaussian. Also we need to collect more data from track 2 to make it less sticked to track’s environment. Also it should be cool to try comma.ai’s network structure instead of nvidia and to compare both of them.

You can find full source code here.