





**Abstract**

We will train a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach has been proven surprisingly powerful. With minimum training data from humans the system learns to drive in trafﬁc on local roads with or without lane markings and on highways. It also operates in areas with unclear visual guidance such as in parking lots and on unpaved roads. The system automatically learns internal representations of the necessary processing steps such as detecting useful road features with only the human steering angle as the training signal. We never explicitly trained it to detect, for example, the outline of roads.

We will use an NVIDIA DevBox and Torch 7 for training and an NVIDIA DRIVETM PX self-driving car computer also running Torch 7 for determining where to drive. The system operates at 30 frames per second (FPS).

In this paper, we describe a CNN that goes beyond pattern recognition. It learns the entire processing pipeline needed to steer an automobile.

**Overview of the entire system**

Three cameras are mounted behind the windshield of the data-acquisition e-rickshaw. Time-stamped video from the cameras is captured simultaneously with the steering angle applied by the human driver. This steering command is obtained by tapping into the vehicle’s Controller Area Network (CAN) bus. In order to make our system independent of the car geometry, we represent the steering command as 1/r where r is the turning radius in meters. We use 1/r instead of r to prevent a singularity when driving straight (the turning radius for driving straight is inﬁnity). 1/r smoothly transitions through zero from left turns (negative values) to right turns (positive values). Training data contains single images sampled from the video, paired with the corresponding steering command (1/r). Training with data from only the human driver is not sufﬁcient. The network must learn how to recover from mistakes. Otherwise the e-rickshaw will slowly drift off the road. The training data is therefore augmented with additional images that show the e-rickshaw in different shifts from the center of the lane and rotations from the direction of the road.

Images for two speciﬁc off-center shifts can be obtained from the left and the right camera. Additional shifts between the cameras and all rotations are simulated by viewpoint transformation of the image from the nearest camera. We approximate the transformation by assuming all points below the horizon are on ﬂat ground and all points above the horizon are inﬁnitely far away. This works ﬁne for ﬂat terrain but it introduces distortions for objects that stick above the ground, such as cars, poles, trees, and buildings. Fortunately these distortions don’t pose a big problem for network training. The steering label for transformed images is adjusted to one that would steer the vehicle back to the desired location and orientation in two seconds.

**Network Architecture**

The network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers. The input image is split into YUV planes and passed to the network.

The ﬁrst layer of the network performs image normalization. The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer conﬁgurations. We follow the ﬁve convolutional layers with three fully connected layers leading to an output control value which is the inverse turning radius. The fully connected layers are designed to function as a controller for steering.

**Training Details:**

To train a CNN to do lane following we only select data where the driver was staying in a lane and discard the rest. We then sample that video at 10 FPS. A higher sampling rate would result in including images that are highly similar and thus not provide much useful information. To remove a bias towards driving straight the training data includes a higher proportion of frames that represent road curves.

After selecting the ﬁnal set of frames we augment the data by adding artiﬁcial shifts and rotations to teach the network how to recover from a poor position or orientation. The magnitude of these perturbations is chosen randomly from a normal distribution. The distribution has zero mean, and the standard deviation is twice the standard deviation that we measured with human drivers.