Self Driving Cars Using CNN

Vishwanath Patil
Computer Software Engineering, San Jose State University
1 Washington Sq, San Jose, CA 95192
Vishwanath.patil@sjsu.edu

***Abstract*— Training a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This approach proved surprisingly powerful.This approach has very less amount of training data from humans the system learns to drive in traffic on local roads with or without lane markings and on highways. It also can drive in areas with less visual guidance such as in parking areas 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.**

***Keywords*— Convolutional neural network, Visual guidance, Visual detection.**

Introduction

CNNs have revolutionized pattern recognition.Prior to the widespread adoption of CNNs,most pattern recognition tasks were performed using an initial stage of hand-crafted feature extraction was done by classifier. The outbreak of CNN's is that features are learned automatically from training examples sets. The CNN approach is mainly built for image recognition tasks because the convolution operation captures the 2D nature of images.Less number of parameters need to be learnt by using CNN compared to other models.While CNNs with learned features have been in commercial use for over twenty years, their adoption has exploded in the last few years because of two recent developments.Labeled data sets such as the Large Scale Visual Recognition Challenge (ILSVRC) have been made available for training and validation in the recent years. Second, GPU accelerated systems are made available in the recent years.This paper describes about CNN that goes beyond pattern recognition. It learns the entire processing pipeline needed to steer an automobile. The groundwork for this project was done over 10 years ago in a Defense Advanced Research Projects Agency (DARPA) seedling project known as DARPA Autonomous Vehicle (DAVE) in which a sub-scale radio control (RC) car drove through a junk-filled alley way. DAVE was trained on hours of human driving in similar, but not identical environments.The training data included video from two cameras coupled with left and right steering commands from a human operator.In many ways, DAVE-2 was inspired by the pioneering work of Pomerleau who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system. It demonstrated that an end-to end trained neural network can indeed steer a car on public roads.

Overview of the DAVE-2 System

Figure 1 shows a simplified block diagram of the collection system for training data for DAVE-2.Three cameras are mounted behind the windshield of the data-acquisition car. 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 the system independent of the car geometry, engineers represent the steering command as 1=r where r is the turning radius in meters. using 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 earn how to recover from mistakes. Otherwise the car will slowly drift off the road. The training data is therefore augmented with additional images that show the car in different shifts from the center of the lane and rotations from the direction of the road.

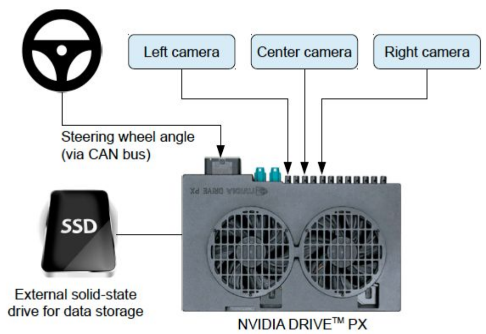

Figure 1: High-level view of the data collection system.

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. Precise viewpoint transformation requires 3D scene knowledge which are not available therefore 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.

A block diagram of the training system is shown in Figure 2.Images are fed into a CNN which then computes a proposed steering command. The proposed command is compared to the desired command for that image and the weights of the CNN are adjusted to bring the CNN output closer to the desired output. The weight adjustment is accomplished using back propagation as implemented in the Torch 7 machine learning package.

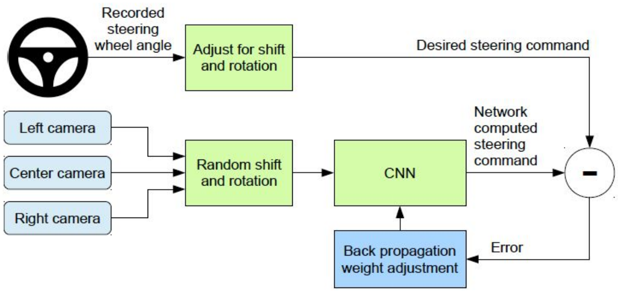

Figure 2: Training the neural network.

Once trained, the network can generate steering from the video images of a single center camera.This configuration is shown in Figure 3.

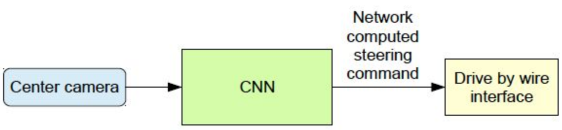

Figure 3

Data Collection

Training data was collected by driving on a wide variety of roads and in a diverse set of lighting and weather conditions. Most road data was collected in central New Jersey, although highway data was also collected from Illinois, Michigan, Pennsylvania, and New York. Other road types include two-lane roads (with and without lane markings), residential roads with parked cars, tunnels, and unpaved roads. Data was collected in clear, cloudy, foggy, snowy, and rainy weather, both day and night. In some instances, the sun was low in the sky, resulting in glare reflecting from the road surface and scattering from the windshield. Data was acquired using either our drive-by-wire test vehicle, which is a 2016 Lincoln MKZ, or using a 2013 Ford Focus with cameras placed in similar positions to those in the Lincoln. The system has no dependencies on any particular vehicle make or model. Drivers were encouraged to maintain full attentiveness, but otherwise drive as they usually do. As of March 28, 2016, about 72 hours of driving data was collected.

Network Architecture

Train the weights of the network to minimize the mean squared error between the steering command output by the network and the command of either the human driver, or the adjusted steering command for off-center and rotated images (see Section 5.2). 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 normalizer is hard-coded and is not adjusted in the learning process. Performing normalization in the network allows the normalization scheme to be altered with the network architecture and to be accelerated via GPU processing. The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer configurations. We use strided convolutions in the first three convolutional layers with a 22 stride and a 55 kernel and a non-strided convolution with a 33 kernel size in the last two convolutional layers. 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, but we note that by training the system end-to-end, it is not possible to make a clean break between which parts of the network function primarily as feature extractor and which serve as controller.

Training Details

Data Selection

The first step to training a neural network is selecting the frames to use. Collected data is labeled with road type, weather condition, and the driver's activity (staying in a lane, switching lanes, turning, and so forth). 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.

Augmentation

After selecting the final set of frames engineers 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.

Simulation

Before road-testing a trained CNN, first evaluate the networks performance in simulation. A simplified block diagram of the simulation system is shown in Figure 5.The simulator takes pre-recorded videos from a forward-facing on-board camera on a human-driven data-collection vehicle and generates images that approximate what would appear if the CNN were, instead, steering the vehicle. These test videos are time-synchronized with recorded steering commands generated by the human driver.Since human drivers might not be driving in the center of the lane all the time, we manually calibrate the lane center associated with each frame in the video used by the simulator. They call this position the "ground truth". The simulator transforms the original images to account for departures from the ground truth. Note that this transformation also includes any discrepancy between the human driven path and the ground truth. The transformation is accomplished by the same methods described in Section 2. The simulator accesses the recorded test video along with the synchronized steering commands that occurred when the video was captured. The simulator sends the first frame of the chosen test video, adjusted for any departures from the ground truth, to the input of the trained CNN. The CNN then returns a steering command for that frame. The CNN steering commands as well as the recorded human-driver commands are fed into the dynamic model [8] of the vehicle to update the position and orientation of the simulated vehicle. The simulator then modifies the next frame in the test video so that the image appears as if the vehicle were at the position that resulted by following steering commands from the CNN. This new image is then fed to the CNN and the process repeats. The simulator records the off-center distance (distance from the car to the lane center), the yaw, and the distance traveled by the virtual car. When the off-center distance exceeds one meter, a virtual human intervention is triggered, and the virtual vehicle position and orientation is reset to match the ground truth of the corresponding frame of the original test video.

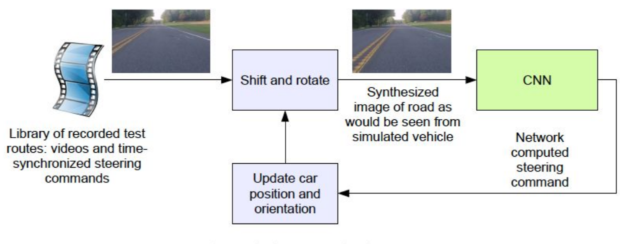

Figure 5: Block-diagram of the drive simulator.

Evaluation

Evaluating of networks is done in two steps, first in simulation, and then in on-road tests.In simulation system has the networks provide steering commands in our simulator to an ensemble of prerecorded test routes that correspond to about a total of three hours and 100 miles of driving in Monmouth County, NJ. The test data was taken in diverse lighting and weather conditions and includes highways, local roads, and residential streets.

Conclusion

Empirically it has been demonstrated that CNNs are able to learn the entire task of lane and road following without manual decomposition into road or lane marking detection, semantic abstraction, path planning, and control. A small amount of training data from less than a hundred hours of driving was sufficient to train the car to operate in diverse conditions, on highways, local and residential roads in sunny, cloudy, and rainy conditions. The CNN is able to learn meaningful road features from a very sparse training signal (steering alone). The system learns for example to detect the outline of a road without the need of explicit labels during training. More work is needed to improve the robustness of the network, to find methods to verify the robustness,and to improve visualization of the network-internal processing steps.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, Winter 1989. URL: http://yann.lecun.org/exdb/publis/pdf/lecun-89e.pdf.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[3] L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., , and D Zuckert. Optical character recognition for self-service banking. AT&T Technical Journal, 74(1):16–24, 1995.

[4] Large scale visual recognition challenge (ILSVRC). URL: http://www.image-net.org/ challenges/LSVRC/.

[5] Net-Scale Technologies, Inc. Autonomous off-road vehicle control using end-to-end learning, July 2004.Final technical report. URL: http://net-scale.com/doc/net-scale-dave-report.pdf.

[6] Dean A. Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report,Carnegie Mellon University,1989.

[7]https://www.researchgate.net/publication/301648615_End_to_End_Learning_for_Self-Driving_Cars