Supervised Learning for Simulated Autonomous Vehicle using Convolutional Neural Network

Haotian Xu
College of Engineering
University of California, Berkeley
Berkeley, CA 94720
Email: haotianxu@berkeley.edu

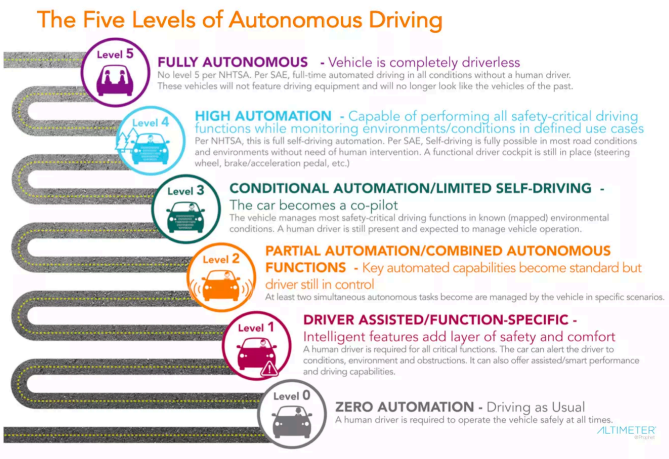
***Abstract*—Autonomous vehicles research is currently one of the most prominent areas of research within the fields of deep learning and computer vision. Many companies around the world that are involved in the automotive industry have joined the race to achieve full, level five autonomy. This project seeks to address this issue by creating and training a convolutional neural network (CNN) using three simulated front-facing RGB cameras paired with human-input steering commands. Once trained, this system is able to map raw pixel data from a single front-facing RGB camera directly to steering commands. Nvidia's autonomous vehicles research was used as a starting point for designing the CNN, and Udacity's self-driving car simulator was used to collect driving data and evaluate the model's performance. This method proved to be surpisingly effective, even with very limited training data. After some data preprocessing and hyperparameter tuning, the model was able to drive around Udacity's "Lake" simulated driving course without crashing or leaving the track. With additional research, this and similar endeavors will help to realize fully-autonomous, driverless vehicles.**

I. Introduction

The need for reliable and safe self-driving systems for autonomous driving is reflected in the emergence of self-driving AI endeavors. Uber, Lyft, Waymo, Tesla, MobilEye, Ford and many other companies have invested millions into self-driving car research in order to compete with one another in the race for a technology that will revolutionize transportation. A self-driving system with level five autonomy will enable humans to live more productively, increase the efficacy of any industry that is involved with the use of automotive transportation, and decrease the number of accidents, automobile related deaths, and general costs of automotive operation.

Many organizations have implemented their own versions of self-driving artificial intelligence. Most major automobile companies have their own R&D teams working on autonomous vehicles research, and this emerging technology will be unavoidably pertinent in the future. While much progress has been made, most developments are protected under the confidentiality of patents and intellectual property. As of now, no entity has been able to successfully implement level five automation.

One of the issues facing autonomous vehicles research is that training often relies on very large datasets and long training times to converge. This makes conducting research in this field inaccessible to anyone who does not have access to extensive datasets and computational power. This project seeks to explore this issue by attempting to train a one-car model with a very limited dataset to perform road following within a driving simulator.



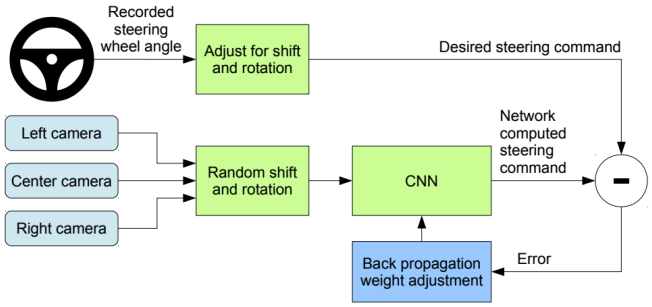
1. The five levels of autonomous driving.

II. Discussion of Related Work

A 2016 study by the Nvidia Corporation trained a CNN to map raw pixels from a single front-facing camera directly to steering commands [1]. Using minimum training data from a variety of road, lighting, and weather conditions, the system learned to drive in traffic on local roads with or without lane markers, on highways, and in areas with unclear visual guidance such as parking lots and unpaved roads.

This system automatically learned internal representations of all processing steps, such as detecting the important features of the road, with only human steering angles as training signal. It was able to optimize all processing steps simultaneously, without explicit decomposition of the problem.

Nvidia's research empirically 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 72 hours of human driving was sufficient to train the car to operate in diverse conditions; such as on highways, local roads, and residential roads; and in sunny, cloudy, and rainy conditions. The CNN was also able to learn meaningful road features from a very sparse training signal of steering alone.



2. Nvidia's system for training the neural network.

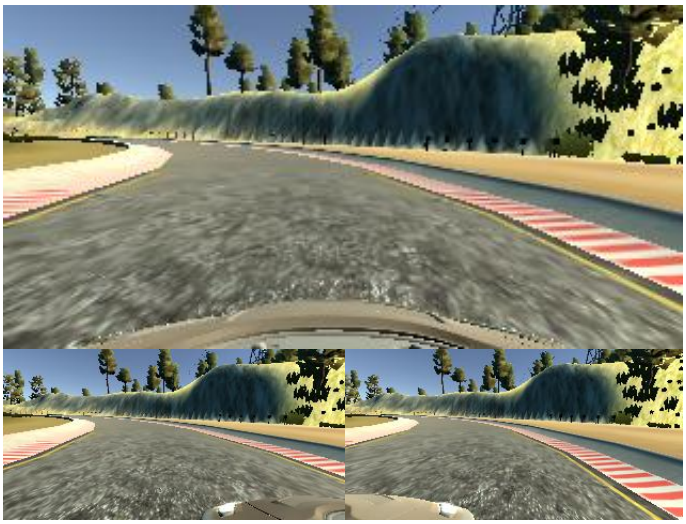
III. METHODS

A. Tools and Dataset

The project was written in Python, using the Keras framework with Tensorflow back-end. The dataset was generated using Udacity's self-driving car simulator. The simulator allows users to record human input in the form of keystroke logging ("keylogging", the steering controls based on arrow keys and the duration of presses), in addition to frame-by-frame image data from three front-facing RGB cameras.

The car was driven four times around the "Lake" course in Udacity's simulator, or about 5.5 minutes of driving. In order to best maintain the quality of the human-input driving data, the car was kept in the center of the lane as best as possible. Frame-by-frame image data for the four laps was captured from center, left, and right front-facing cameras attached to the vehicle within the simulator.

The final image dataset consisted of 4,422 frames with center, left, and right camera angles of RGB images of 160x360 resolution. This was represented in a numpy array of 4,422 (total frames) x 3 (camera angles) x 160 (height in pixels) x 360 (width in pixels) x 3 (RGB values).



3. Images taken from center, left, and right camera angles (cropped).

B. Data Preprocessing

In order to reduce the dimensionality of the dataset, the images were first cropped from 160x360 pixels to 66x200 pixels. This crops the dimensions of the input images to match the dimensionality of the images from Nvidia's research, as well as eliminates most of the sky and the extreme edges from the images.

Next, the dataset was augmented by flipping the image over the x-axis. This was appended to the dataset to double the size of the original dataset. The images from the left and right camera angles were also appended to the dataset with steering angle corrections to help training for sharp turns.

Additionally, the input images, which were in RGB color space, was mapped to the YUV color space, which helps emphasize certain important features of the road. Finally, the dataset was shuffled, in order to randomize the images that will be allotted into training data and validation data.

The preprocessed dataset consisted of 26,532 images of 66x220 resolution and YUV color space, split into original and flipped images with center, left, and right camera angles. This was represented in a numpy array of 26,532 (total images) x 66 (height in pixels) x 220 (width in pixels) x 3 (YUV values).

C. Network Architecture

Nvidia's CNN architecture consisted of 9 layers, including a normalization layer, 5 convolutional layers, and 3 fully-connected layers. Because their CNN performed so well under conditions of limited training data and sparse training signal, the network architecture of this project was modeled after Nvidia's design. The final network architecture consisted of 10 layers, including a normalization layer, 5 convolutional layers, a dropout layer, and 3 fully-connected layers.

The network takes as input YUV images of 66x200 resolution represented as a numpy array with dimensions 3x66x200. After passing this data through the network, the CNN returns an output control value as the steering angle.

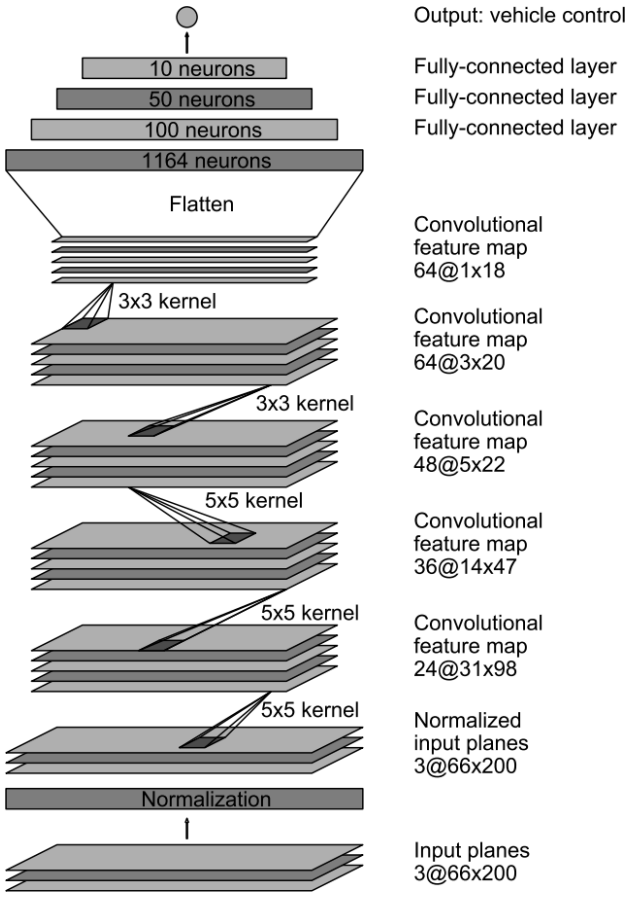
The normalization layer performs image normalization. The normalizer is hard-coded, and is not adjusted during training. Performing image normalization in the neural network allows the normalization scheme to be altered with the network architecture, as well as to be accelerated using GPU processing.

The five convolutional layers are designed to perform feature extraction. The first three of the five convolutional layers use striped convolutions with a 2x2 stride and a 5x5 kernel. The last two convolutional layers use non-strided convolutions with a 3x3 kernel. Nvidia empirically determined these hyperparameters through a series of experiments that varied the layer configurations. All convolutional layers used the exponential linear unit ("elu") activation function to prevent the vanishing gradient problem.

The dropout layer is designed to perform dropout regularization. After evaluations of initial models showed signs of overfitting, this dropout layer with dropout of 0.5 was introduced to reduce the overfitting.

The three fully connected layers are designed to function as a controller for steering. However, because this system is trained end-to-end, it is not possible to make a clean distinction

between the parts of the network that serve as the feature extractor and the parts that serve as the controller.



4. Nvidia's CNN architecture.

D. Training, Evaluation, and Optimization

The model was trained using the Adaptive Moment Estimation (Adam) optimization algorithm. The Adam optimizer is similar to classical stochastic gradient descent, but preserves momentum by altering the learning rate based on the first and second moments of the gradient.

The data was trained over ten epochs, and the model was saved at the end of each epoch. The performance of each model was then evaluated visually based on its performance in a driving test, and the results were recorded. Because the model is trained on static images without a concept of velocity, the velocity of the vehicle during evaluations was manually set to a constant 10mph.

To optimize the validation split, the model performance for several values between 0.01 and 0.5 was evaluated. The one that yielded the best result was a 0.3 validation split. This meant that the last 30% of the dataset (roughly 1.2 laps around the track) was being used as validation data. However, this introduces a significant bias, since the human driving input is different for each lap.

To resolve this issue, the numpy array is randomly shuffled during preprocessing. This scrambles the order of the images, and causes the dataset to be randomly split between training and validation data.

After these optimizations, the model was able to successfully drive the vehicle around Udacity's "Lake" simulated driving course without crashing or leaving the track.

IV. Conclusion

This project has empirically demonstrated that CNNs are able to learn the entire task of road following, even with an extremely limited dataset of 5.5 minutes of human driving input. Like in the findings of the Nvidia Corporation, the CNN was able to perform this task without explicit manual decomposition of the processing steps when trained and evaluated within a simulator. The CNN was also able to learn meaningful road features from a very sparse signal of steering angles.

V. Improvements and Future Research

Further research is needed to incorporate velocity data into the training process, so that the network would be able to take full control of the driving process. Research is also needed to improve the robustness of the network to operate outside the specific conditions of the simulator. These will most likely require a larger dataset than was used during this project, but should greatly improve the performance of the network.

Another area to explore is the effects of training the model using deep reinforcement learning. One of the problems this project encountered is the consistency of the human driving input. Supervised learning fundamentally assumes that the training data is sampled from examples of great performance. However, given the inconsistency of human ability, this assumption will often times not hold. As a result, supervised learning will often perform worse than deep reinforcement learning (see AlphaGo Master vs AlphaGo Zero).

Therefore, the next step would be to train a second model using deep reinforcement learning, and compare the results of that experiment with the results of training with supervised learning. This comparison will hopefully shed some insight into the strengths and limitations of supervised learning versus deep reinforcement learning.

References

1. NVIDIA Corporation, "End to End Learning for Self-Driving Cars," April 2016. https://arxiv.org/pdf/1604.07316.pdf.

Data

1. https://drive.google.com/open?id=1inxGfkKfqNE2V-zzFe2zs39FegHMFe6I