PROGRESS REPORT: AUTONOMOUS CAR USING CNN

Rudra Dey
Student# 1010124866
rudra.dey@mail.utoronto.ca

Pravin Kalaivannan
Student# 1010141295
pravin.kalaivannan@mail.utoronto.ca

Aadavan Vasudevan
Student# 1010101514
aadavan.vasudevan@mail.utoronto.ca

Abishan Baheerathan
Student# 1010218756
abishan.baheerathan@mail.utoronto.ca

ABSTRACT

This document presents our team's procedure to create the baseline model code for an autonomous self-driving car. ...

1 PROJECT DESCRIPTION

2 INDIVIDUAL CONTRIBUTIONS AND RESPONSIBILITIES

2.1 INDIVIDUAL CONTRIBUTIONS

3 DATA PROCESSING

Data was collected from the CARLA simulator, including images and corresponding steering values. We then normalized the images allowing us to train our CNN. Further, data augmentation was applied to increase the size and diversity of the dataset, leading to better generalization of the model.

3.1 DATA COLLECTION AND CLEANING

In CARLA, there is vehicles that have autonomous functionality, which we used to collect data. We attached an RGB Camera Sensor to the car to record 10 frames per second (FPS) as the car travels around a designated map. In addition to the images, data for the steering values and turn signals of the car was collected allowing for use as ground truth labels for training the model. This data collection was done on diffreent maps with different weather conditions to ensure a diverse dataset. We collected samples on eight maps, in which the car collected data for 30 minitues, resulting in 14400 total images.

To clean the data, we removed any images that were all black, duplicates or not the correct resolution of 320x240 leaving us with 14239 images. We then saved the data as numpy arrays, zipped and uploaded to a shared Google Colab file along, ensuring access for all members.

Procedure:

1. Spawn in a vehicle
2. Attach an RGB Camera Sensor (front-facing) to the dash of the car.
3. Collect image data (10 fps) along with steering data and turn signals.

4. Clean the data by removing corrupted images and duplicates.
5. Save the data as numpy arrays for easy access and processing.

3.2 Data Preprocessing

All collected images were resized to a resolution of 160x120 pixels, with the pixel values normalized to a range of [0, 1]. This was done to ensure consistency in the input data

3.3 Data Agumentation

3.4 Plan for Testing on Unseen Data

3.5 Challeneges Faced and Solutions

Data will be collected without traffic in the roads (i.e no pedestrians or cars on the road). The car would have a constant speed and the camera will always be mounted in the same position and angle at the front of the car. The FPS will be set to a constant 20 FPS. We will use the CARLA's autopilot to gain the steering values necessary to be used as ground truth labels. To get a diverse set of data, we will collect data in different maps and weather conditions.

4 Creating Baseline Model

This baseline model will predict steering angles from grayscale images for an autonomous vehicle simulator (which already has a autopilot built in).

The following libraries will be used:

1. numpy: For numerical operations on arrays
2. matplotlib: For visualization of data
3. scikit-learn: For machine learning components
   (a) Ridge: A regularized linear regression model
   (b) MSE: Mean Square Error for evaluation metrics
   (c) train_test_split: For data splitting

The data will be loaded using two numpy arrays: images.npy and angles.npy. The images.npy will contain N grayscale images of resolutions 160x120, and the angles will have the corresponding images steering angles (of the car) in a range from [-1,1]. This makes it so the image data is represented as 3D arrays of samples x hieght x width as the dimension. The steering angles are continuous (regression problem).

The images will be flattened from their 2D representation of 160x120 matrix to a 1D representation of 19200 vectors. Where each image becomes a single row in a feature matrix. Additionally, we will split do a 80-20 train-test split.

The baseline model will be a ridge regression model. The ridge regression model is basically, the same as a linear regression model except, it regularizes large weights. It adds an L2 penality to discourage large weights, this helps with overfitting. The same as linear regression as we try to fit: $\hat{y} = Xw + b$, but now we try to minimize the loss function: $LOSS = MSE + \alpha||\omega||_2^2$. The model will learn the weights corresponding to each pixel's contribution to steering.

We will compute two metrics, the mean square error (MSE) and the R^2 score (proportion of variance).

4.1 Building the model

4.2 Evaluating the model

5 Training Neural Netowrk

*Quantitative Results: Training Loss (MSE) and Validation Loss

5.1 Model Training Script

5.2 Logging

5.3 Outputting Results

6 Evaluating with Inference

*Qualitative Results: Using model for actual driving in CARLA

6.1 Running model in CARLA using live camera feed