Behavioral Cloning of Autonomous Vehicles

Vatsal Sanghavi
vats@umd.edu

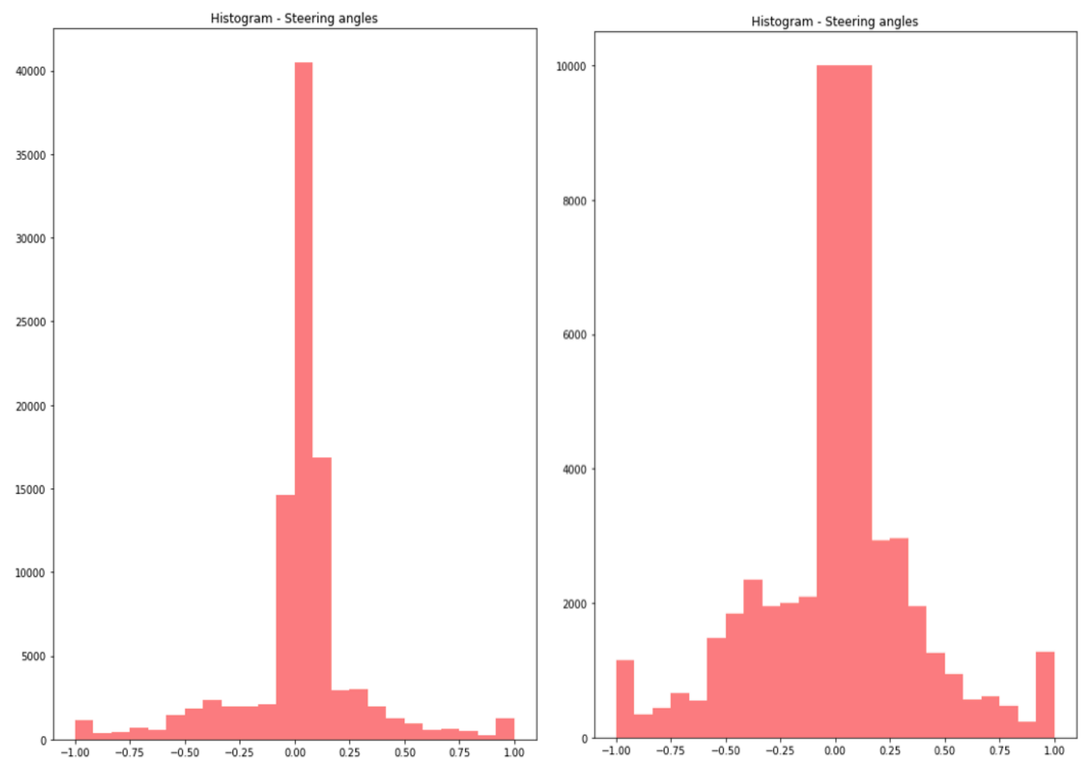
Bhavin Kothari
bhavin57@umd.edu

Viraj Shah
viraj97@umd.edu

Introduction

The primary purpose of this project was to learn how to autonomously drive a car by leveraging deep learning for behavioral cloning. We used the video game GTA 5 for generating frames while driving around the roads of the game and then trained our model on that dataset which included steering angle and throttle as outputs.

Methodology



Unbalanced steer angle targets Balanced steer angle targets

For data pre-processing, we started with data balancing to ensure that our neural network was not biased to the most frequent input - which was when the steering angle was at 0 degrees. This was balanced out by randomly filtering some of these inputs. We then performed data augmentation wherein images were flipped. Changing steering angles to maintain a balance between left and right steering targets was also implemented to ensure a more complete training environment for our model. Unwanted details from our images, including trees, mountains and sky were also removed. This was implemented to reduce the memory required to store and run the model. We also zoomed in on images to get a better view of the roads – hence ensuring better performance. Following this, we panned randomly selected images to shift focus onto the vehicle by blurring out the backgrounds. Finally, the image brightness was artificially changed to augment the test set. This was done to ensure better model performance during all times of the day, including adverse lighting conditions. The underlying aim of this pipeline was to improve model generalization capability and eliminate any bias whatsoever.

Data Pre-processing Pipeline

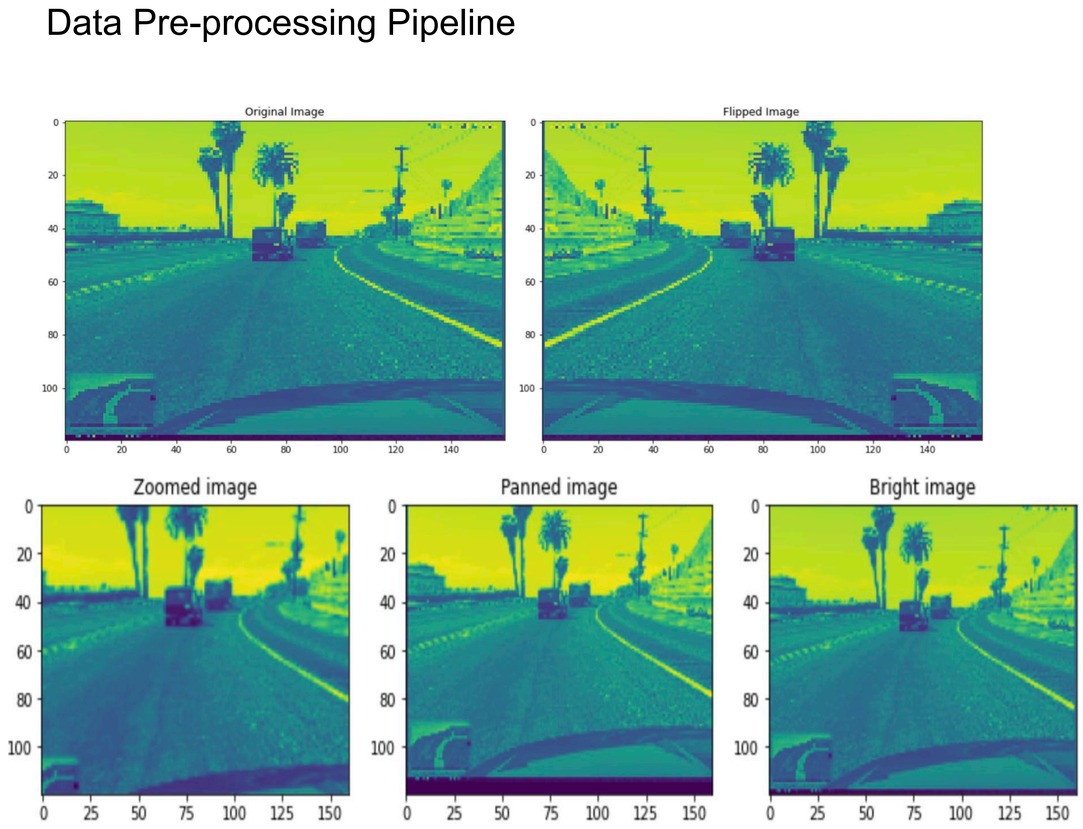


Image transformations – Data Augmentation

Neural Network based model training

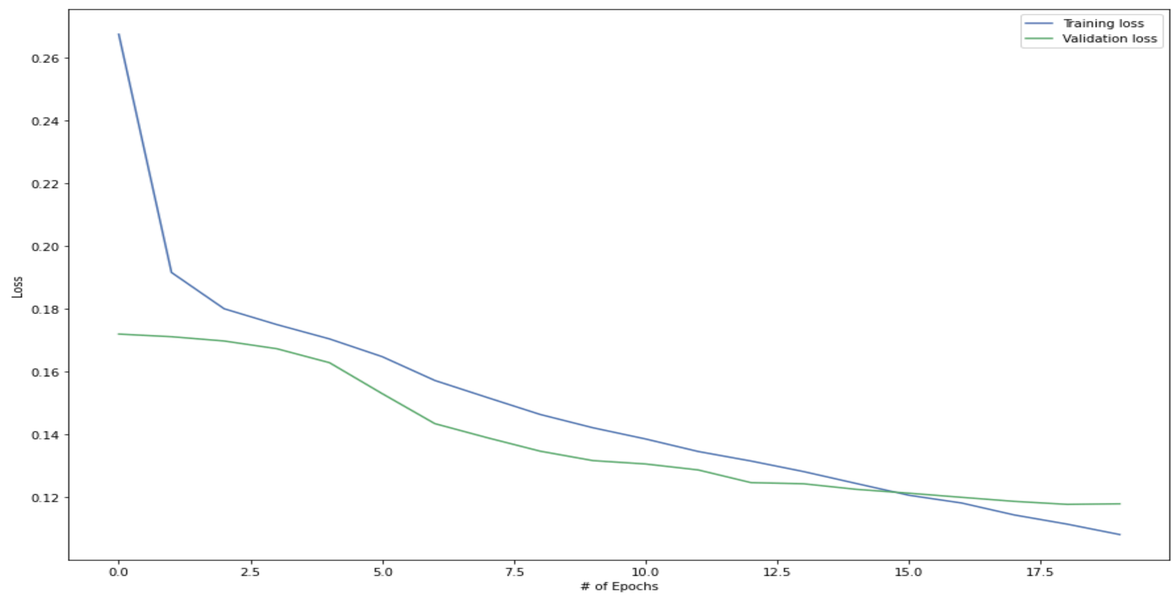
In our final model, we had 9 trainable layers which consisted of 5 convolutional layers and 4 fully connected feed forward layers. For speeding up the learning process and classification, we used a nonlinear ELU function instead of RELU. No pooling layers were used as it invariantly shifts the output to some extent which we do not want for our vehicle on roads. Dropouts were made after every fully connected feedforward layer so that validation loss would reduce, and the model would not overfit itself.

Convolution: 5x5, # of filters: 24, strides: 2x2, activation: ELU
Convolution: 5x5, # of filters: 36, strides: 2x2, activation: ELU
Convolution: 5x5, # of filters: 48, strides: 2x2, activation: ELU
Convolution: 3x3, # of filters: 64, strides: 1x1, activation: ELU
Convolution: 3x3, # of filters: 64, strides: 1x1, activation: ELU
Flattening
Fully connected: neurons:100, activation: ELU
Dropout (0.5)
Fully connected: neurons:50, activation: ELU
Dropout (0.5)
Fully connected: neurons:10, activation: ELU
Dropout (0.5)
Fully connected: neurons:2 (output)

Final Model Architecture

Results

Adam Optimizer was used to minimize the MSE and a learning rate of 0.001 produced quality results. Gradient Descent (with momentum) technique used for back-propagation. With an increase in number of epochs, there was a smooth decrease observed in the validation loss which reaches a value as low as 0.11 at the final 20th epoch. A batch size of 500 was considered.



Model Performance – MSE Loss

On deployment of the trained model for real-time game predictions using external dependencies, following are the model outputs to steer the car automatically.

```
COUNTDOWN begins...
Please switch to GTA 5 window.
4
3
2
1
Running...
Prediction:  tensor([[ 0.0366, -0.1258]], grad_fn=<AddmmBackward>)
Prediction:  tensor([[ 0.0904, -0.1490]], grad_fn=<AddmmBackward>)
Prediction:  tensor([[ 0.0007, -0.1668]], grad_fn=<AddmmBackward>)
```

Console Output Window - Testing

Conclusion

With an in-depth knowledge of mini-batch, activation function, adaptive learning rates, gradient descent with momentum to pick the best function, we were successfully able to output a working model to be used during real time game play. The compute-intensive model training was executed over Google Collab's GPU runtime and helped us test various architecture designs for MSE Loss as the performance metric. The process of deploying trained model to self-steer the car in GTA-V gaming environment was done during a night mode to demonstrate performance in adverse lighting conditions.

This can be improved by using a larger training dataset and CNN-LSTM can be applied instead.