KING MONGKUT'S UNIVERSITY OF TECHNOLOGY LATKRABANG

FACULTY OF ENGINEERING

01416520 – VISION SYSTEM FOR MOBILE ROBOTICS

BEHAVIORAL CLONING

INSTRUCTED BY: BEEYING SAEUNG

NAME:

ANANCHANA LERTPHITAKSUNTORN	62011087
NATTARRUD CHAROENNITHI	62011178
PANADDA LUCKANANUKUL	62011182

DATE OF SUB: 10TH DEC 2021

ABSTRACT

Behavioral cloning is a method which human skills are captured and reproduced in a computer program. Self-driving cars are one of the main headline scopes for AI implementation of behavioral cloning. In this project, we study how to gather, train, and implement CNN model on self-driving cars. Udacity simulator is a driving simulator where we can drive and collect data simultaneously. Data is then preprocessed and augmented to improve training time. The model has an accuracy of 53% and can drive the car autonomously on both tracks successfully for an infinite number of times.

CONTENTS

1. DATASET DESCRIPTION

In order to train the model, we must get data for driving. Udacity Simulator enabled us to gather the dataset. The car in the simulator has three camera images: a center camera, a left camera, and a right camera. will be collected on a continuous basis while in training mode.

We gathered a total of **8,038 observations**. Each with 6 features and 1 target. [center, left, right, throttle, brake, speed, steering_angle]

Train data : 80%

Validation data : 10%

Test data : 10%

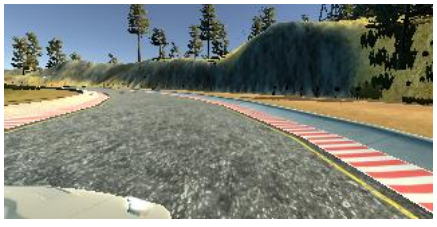
The technique used for gathering driving data for each map is

1. 1 lap clockwise
2. 1 lap counter clockwise
3. 1 lap recovery

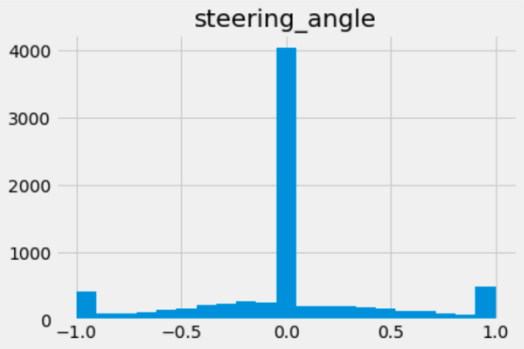
Recovery driving is used when the trained model drives out of lane at a specific location and needs to recover.

left camera

center camera



right camera



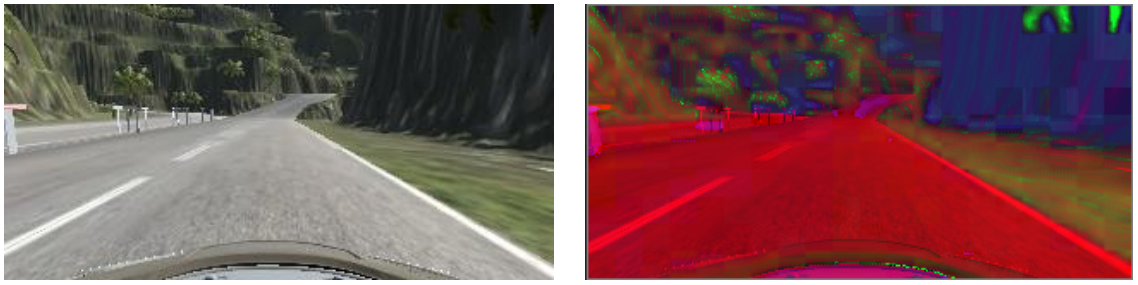
Steering angles are plotted as shown above, there are high amounts of kurtosis. Thus, the model will mostly drive straight, which is good for straight roads but bad for sharp curves. Some data augmentation is required to create better distributions.

2. TRANING SETTING

- DATA AUGMENTATION

Image flipping

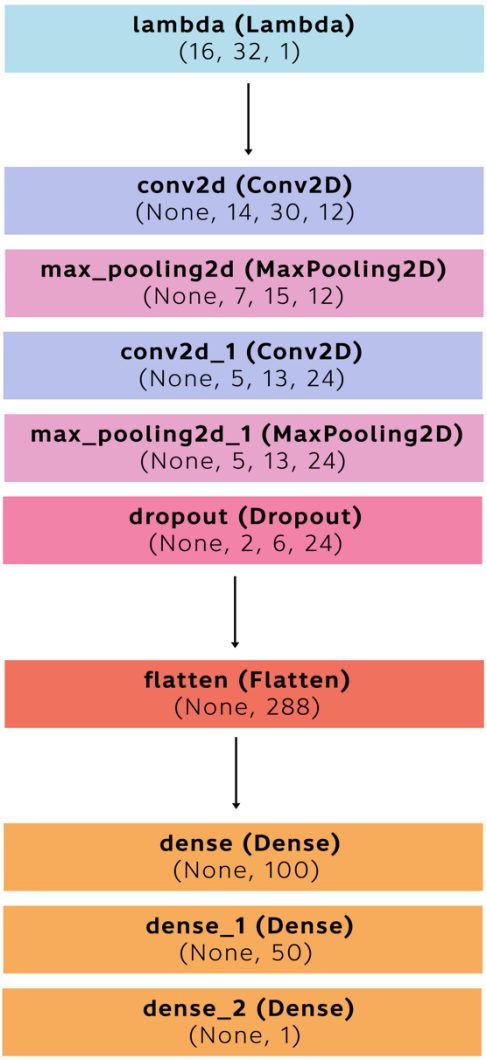
To normalize the steering angle distribution, the images were flipped and multiplied their steering angle by -1. This technique doubles the size of our dataset.



RGB to HSV color space conversion

We reduced the sizes of the images by a factor of 10 and converted them to HSV colorspace. This reduced the number of parameters and dramatically increased training times. All pixel data is normalized and standardized in the first layer of the CNN via a Lambda function.

- MODEL DESIGN



Total trainable parameters: 36,737

- HYPERPARAMETERS

optimizer = adam

epochs = 60

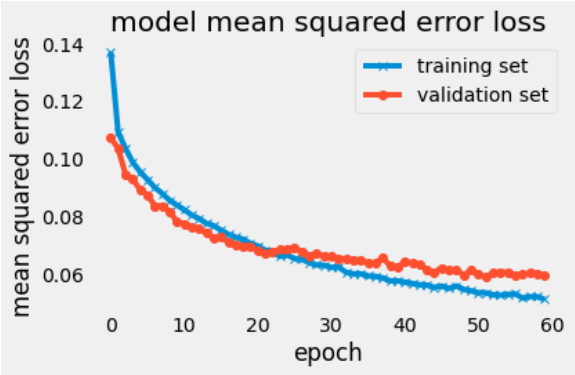
learning rate = 0.0007

batch size = 128

3. LOSS FUNCTION

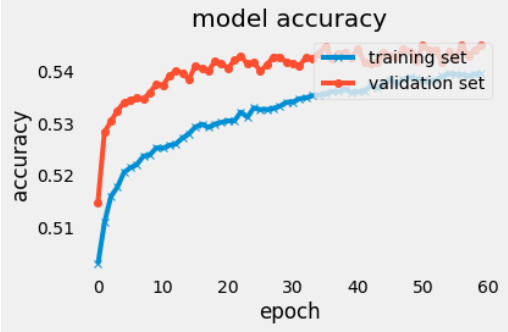
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

We use mean squared error to evaluate our model at every epoch. Mean squared error uses y true and y predicted to estimate the error and how close the model is to the real output. The closer the loss function is to 0, the better is our model at learning the dataset.



Here, we can see that the loss function decreases for each epoch. After 60 epochs, the loss value starts stabilizing so we stop the training. The final validation loss value after 60 epochs is 0.06.

4. RESULT



The model used on the test set has an accuracy of 53%. The model can be further improved by gathering more training data. Even though the accuracy seems low, the model performs well in a real driving environment.

The model can perform on both lake and jungle tracks successfully and indefinitely.

DEMO VIDEO

Lake track: https://youtu.be/EfAdbgRNFuc

Jungle track: https://youtu.be/qProL0ACKqQ

5. CONCLUSION AND FUTURE WORK

From the result, the model can drive successfully around two maps. There are still some errors as the car can steer out of the lane. To fix this issue is gaining the recovery dataset which improves the performance of the model.

As a future work, We'll concentrate on including object detection features so that the vehicle can operate in a real-world setting with suitable control and speed variation dependent on objects and surroundings.

REFERENCES

1. *KERAS*. (n.d.). Retrieved from https://keras.io
2. *tf.keras*. (2021, 5 11). Retrieved from TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/
3. *self-driving-car-sim*. (n.d.). Retrieved from GitHub: https://github.com/udacity/self-driving-car-sim