Steering Angle Prediction in Cars

Mohammed Shuhad (301462898)

ENSC 813 : Final Project Report

mohammed_shuhad@sfu.ca

1 Introduction

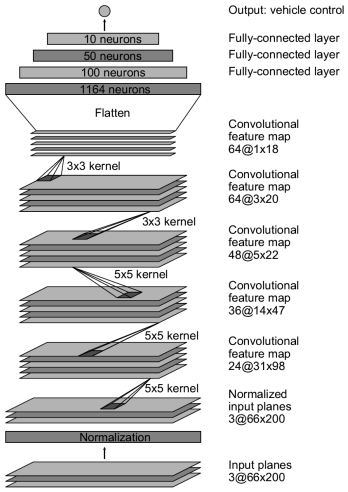
Humans have been trying to discover new ways to make their lives easier from the beginning of time. Due to the recent technological advancements there has been a lot of effort that went into automating tasks. We ve had great success in automating monotonous tasks. Having had great success in that field we have shifted our focus to automating more complex problems which usually involve human perception. One such field where we had great progress is the image processing field. Image processing problems have had an insurgence ever since the Alexnet won the ImageNet Large Scale Visual Recognition Challenge which used deep neural net models to have very high accuracy.

Currently there is widespread use of such deep models for image processing tasks in self-driving cars which are on the road today. The research of self-driving algorithms really took off from the paper proposed by NVIDIA [1] which provided a simple convolutional neural network which was able to predict steering angle from raw pixel data without having to extract any additional features from the images. Ever since then the engineering community had come together to host several competitions such as the udacity challenge and many challenges online like the Kaggle dataset . Several other research has been inspired from this model [1] to solve for more specific problems in particular conditions, a brief study of some of the works have been done on section 2 of this report.

In this project I have taken the model proposed by Bojraski *et al* [1] as the baseline model and introduced recent advancements to it, to make it a better model. Getting high accuracy from this model is a challenge due to the high diversity of possibilities this model can be exposed to. Therefore results in a controlled environment such as a simulation is only feasible here due to the constraints in computing power and availability of dataset from the real world.

2 Related Works

In [1], the authors proposed a deep neural network in which they showed the ability of CNNs to learn driving, trained from a dataset of images from a single front facing camera. They had great success in generalizing the model without having to manually decompose the different features such as "lane marking, path planning and control" [1]. The study successfully showcased the ability of a CNN Deep Neural network to learn meaningful features of the road by using steering angle as the signal for training. They have used NVIDIA DevBox and Torch 7 for training of the network. M. Agarwal et al [9] have developed a model inspired from [1] using tensorflow library and have obtained good results.

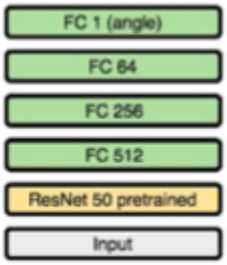


CNN architecture used in [1]

The concept of residual blocks introduced by He *et al* [2] has shown that it is easier to optimize and increase the accuracy by inserting residual blocks in deep networks. This technique was introduced to address the difficulties in optimizing the network. They have also shown that such a model generalized better for image classification.

In the experiments done in [5] it has been observed that using ELU as the activation function leads not only to faster learning, but also to significantly better generalization performance than ReLUs and Leaky ReLUs on networks with more than 5 layers. On CIFAR-100 ELUs networks significantly outperform ReLU networks with batch normalization. The concept of Batch Normalization has been explored in [3] and it has shown that it forces the activations throughout a network to take on a unit gaussian distribution at the beginning of the training.

In the study by S. Du *et al* [4] have introduced different deep learning techniques such as Transfer Learning, 3D CNN, LSTM and ResNet to a model inspired from [1] and have done an analysis of the results. They have pursued a model to capture the temporal information present in driving tasks by using 3D CNN and LSTM. But when comparing the overall results the transfer learned model(ResNet) had the best results, showing that more complex models are required to gather the temporal information in driving.



Transfer Learning Architecture used in [4]

3 Data

The dataset used is explained in the subsequent sections.

3.1 Real Camera Images

Training data for Real world scenarios have been obtained from [7]. The input images are recorded from the front dash cam of a car. The labels for these images exist as angles in degrees (-25 to 25).

Images from the front facing cam in a car [7]

3.1 Car Simulation Images

The training data has been obtained from [6], which comprises images from 3 cameras on the front(left, center and right) and labeled with steering angle, throttle, brake and speed. This dataset has been collected from the Udacity car simulator program.

Images from left, center and right cameras of the Car in Udacity simulator [6].

The steering angle in this dataset has been scaled down to the range -1 to 1. To see visible changes in the model it has been scaled back to the actual value in degrees -25 to 25, so that the MSE calculated will work as a credible metric in judging the performance of the model.
In Order to use the images in the left and right cameras a shift of -5 degree and +5 degree is applied to the steering labels.

4 Model

The CNN network computes the result and back propagates the error value based on the mean squared error matrix and updates the weights and biases of the network for optimal results and better applicability of the network. The model inspired from [1] is taken as the baseline model and its performance is inspected by analyzing the validation and training loss. The final proposed model (Model H) here is the cumulative effort of adding and changing layers to the baseline model to give the most optimum results.

4.1 Pre Processing and Augmentation

Training and Validation Images are converted into YUV format from RGB to make them more human visually appearing which has become the standard in the implementation of autonomous driving [1]. Vectorisation and Standardization have also been added inspired from [9] to normalize the input. Certain models(Model I) have been trained on the horizontally flipped images, with their steering angles also flipped to analyze if the model has learned the symmetric features of the road.

4.2 Model Architecture

Layer	Layer Parameters
Input with BatchNormalisation	Dimension = (160, 80), Channels = 3
Convolution2D with Activation and BatchNormalisation	Kernel size = (5 x 5), No. of filters = 24, Strides = 2, Activation = ELU
Convolution2D with Activation and BatchNormalisation	Kernel size = (5 x 5), No. of filters = 36, Strides = 2, Activation = ELU
Convolution2D with Activation and BatchNormalisation	Kernel size = (5 x 5), No. of filters = 48, Strides = 2, Activation = ELU
Convolution2D with Activation and BatchNormalisation	Kernel size = (3 x 3), No. of filters = 64, Strides = 1, Activation = ELU
Convolution2D with Activation and BatchNormalisation	Kernel size = (3 x 3), No. of filters = 64, Strides = 1, Activation = ELU
Dropout	0.5

Fully Connected layer with Activation	1000 neurons, Activation = ELU
Fully Connected layer with Activation	100 neurons, Activation = ELU
Fully Connected layer	1 neuron

5 Result and Analysis

Hardware specification of the computer used to train the model are as follows:

- 8 GB RAM
- Intel i5 processor

Programming language and libraries used:

- Python 3.7
- Keras and Tensorflow libraries
- Open CV

5.1 Results of the Model on Real Camera Images

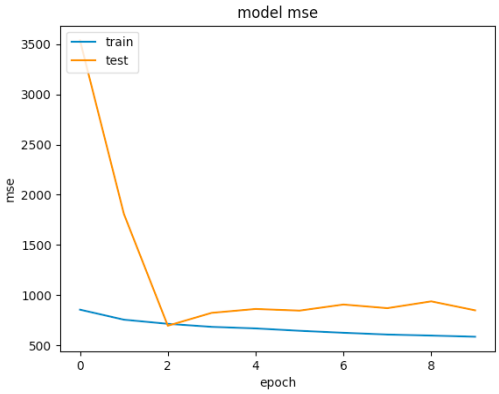
The following Results are obtained on the dataset mentioned in section 3.1. The Training parameter is as follows :

- Epochs - 10
- Batch size - 128
- Loss function used - Mean Squared Error (MSE)
- Learning Rate - 0.0001
- Optimizer - Adam
- Training size - 41k images
- Validation size - 4k images

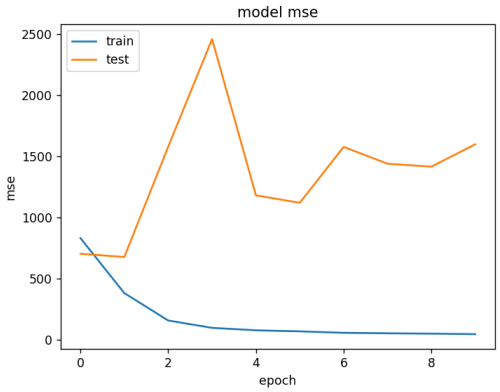
Models Trained:

- Model A : Baseline model Inspired from [1].
- Model B : Relu activation and BatchNormalisation added in between all the convolutional and fully connected layers.

Model	Trainable Parameters	Training loss	Validation loss	Time taken to train (Seconds)
A	3,159,933	584.4135	847.4879	4192.4
B	3,160,405	47.688	1598.0425	2388.8



Model - A



Model - B

The Model B improved over Model A only in terms of training loss and time to train. These 2 models doesnt seem to learn the features of the road to perform the task of driving. This is mainly because of the high variation in input that could exist in a real world scenario - such as the weather, lighting, traffic etc. Nonetheless Model B seems to be performing relatively well on the data it has already seen before (Training data). Therefore a similar model can be installed in a self-driving car, if it takes the route on which it has been trained before.

5.2 Results of the Model on Simulation Images

The Training parameters is as follows:

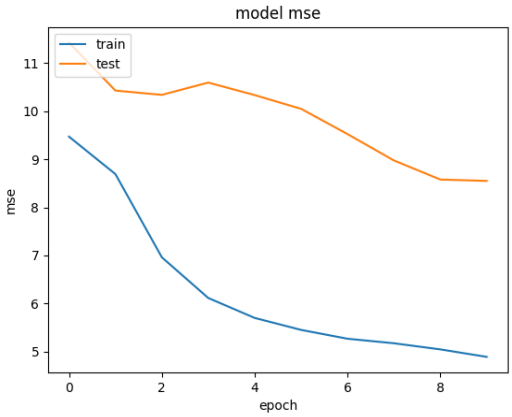
- Epochs - 10
- Batch size - 128
- Loss function used - Mean Squared Error (MSE)
- Learning Rate - 0.0001
- Optimizer - Adam
- Training size - 8k images
- Validation size - 1k images

Models

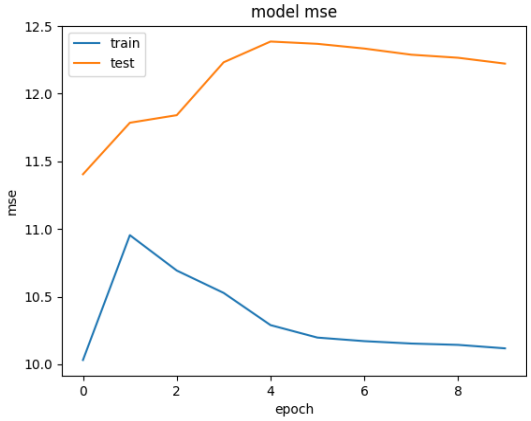
- Model C : Baseline model Inspired from [1] (same as Model A in Section 5.1).
- Model D : Relu activation and BatchNormalisation added in between all the convolutional and fully connected layers of Model C.
- Model E : L2 Norm Penalty($\lambda = 0.0001$) added to the convolution layers of Model C Inspired from [8].
- Model F : Replacing the activation function of Model E from RELU to ELU [5].
- Model G : Transfer Learned Model using ResNet50 as the non trainable layer and 4 fully connected trainable layers, Inspired from [4]. (1000, 512, 256, 64)

Model	Trainable Parameters	Training loss	Validation loss
C	3,159,933	3.34	11.8

D	3,160,405	3.97	11.9
E	3,160,405	3.4	8.7
F	3,160,405	4.88	8.55
G	2,709,353	10.11	12.22



Model F - Plot of Loss vs epochs



Model G - Plot of Loss vs epochs

Even though the Training loss of Model F is a little higher than the other models. F had the best generalization (difference between the training error and validation error was minimal). Therefore the best model from this experiment is Model F. The Transfer learning model (Model G) showed good trends in training but the Loss is too high to consider it.

5.3 Results of the Model on Simulation Images (All 3 Cameras)

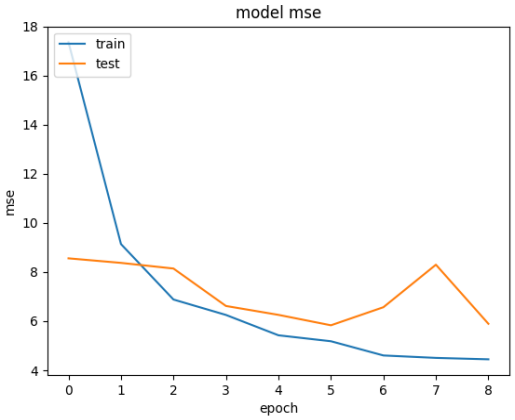
The Training size is increased here, since left, center and right cameras are considered here. The Training parameters is as follows:

- Epochs - 10
- Batch size - 128
- Loss function used - Mean Squared Error (MSE)
- Learning Rate - 0.0001
- Optimizer - Adam
- Training size - 24k images
- Validation size - 3k images

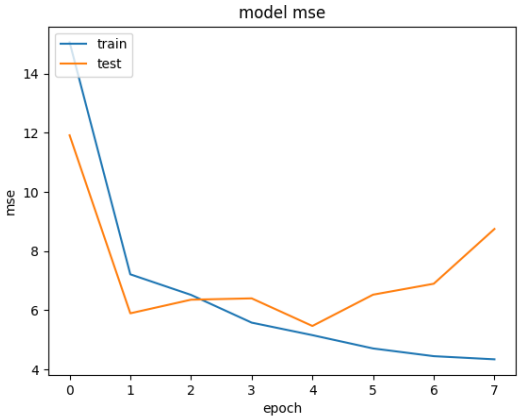
Models

- Model H : Channel Standardization applied to all the inputs on top of Model F.
- Model I : Channel Standardization and Data augmentation (images and labels flipped)

Model	Trainable Parameters	Training loss	Validation loss
H	3,160,405	4.44	5.89
I	3,160,405	4.34	8.7



Model H - Plot of Loss vs epochs



Model I - Plot of Loss vs epochs

Out of all the Models Tested, The Proposed Model (Model H) generalized the most and also showed relatively small loss.

6 Conclusion

6.1 Findings of the Work

After conducting the experiments it has been found that the proposed model generalizes well and has relatively low training loss. On observing the decreasing trend in training loss it can be inferred that the model would perform well in the environment where it has been trained - such as the particular track in the Udacity car simulator. The model fails to capture the diversities present in real world scenarios and doesn't generalize well.

6.2 Future Scope

To Analyze more Regularization techniques such as Bootstrapping and Bagging. And also to train the model with a more diverse dataset from real world images.

6.3 The Code

The code used to train and save the models - train.py
The code used to predict the results from a saved model and to check the error - predict.py
The code developed to generate video and show the results of a model in real time - video_predict.py

References

[1] M. Bojarski et al. "End to end learning for self-driving cars." In *arXiv preprint arXiv:1604.07316*. 2016.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

[3] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". *arXiv preprint arXiv:1502.03167*, 2015.

[4] S. Du, H. Guo and A. Simpson. "Self-driving car steering angle prediction based on image recognition." In *arXiv preprint arXiv:1912.05440.* 2019.

[5] D. A. Clevert, T. Unterthiner and S.Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." In *arXiv preprint arXiv:1511.07289*. 2015.

[6] Kaggle, “Training Set: Self Driving Cars”, 2019, [Online]. Available: https://www.kaggle.com/datasets/roydatascience/training-car. [Accessed: April 3 , 2022].

[7] S. Chen, “driving-dataset” 2021. [Online].
Available: https://github.com/SullyChen/driving-datasets. [Accessed April 3, 2022].

[8] I. Bajic. ENSC 813. Class Lecture, Topic: “Regularization”, School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada, Feb. 17, 2022.

[9] M. Agarwal and S. Dasgupta, “Building a Self-Driving Car Using Deep Learning and Python” 2020. [Online].
Available: https://github.com/MalayAgr/SteeringWheelAnglePredictor. [Accessed April 3, 2022].