

Steering Angle Prediction using Image Processing & NN

Dhavan Raveendranath - dhavanraveendra@oakland.edu

Sunish Vadakkeveetil - svadakkeveetil@oakland.edu

GitHub Link:

https://github.com/svadakkeveetil/csi5130_steer_angle_pred



Agenda

- *Introduction*
- *Related Work*
- *Data*
- *Extraction*
- *Models*
- *Results*
- *Conclusion*

Introduction

- Levels of Autonomy
- Features required multiple sensors to perceive the environment
- 3 inputs - Brake, Throttle and Steering
- Research on Drive by wire systems to achieve L4+ autonomy
- Accurate estimation & control of steering wheel angle important for safety & stability of system



SAE J3016™ LEVELS OF DRIVING AUTOMATION™

Learn more here: [sae.org/standards/content/j3016_202104](https://www.sae.org/standards/content/j3016_202104)

Copyright © 2021 SAE International. The summary table may be freely copied and distributed AS-IS provided that SAE International is acknowledged as the source of the content.

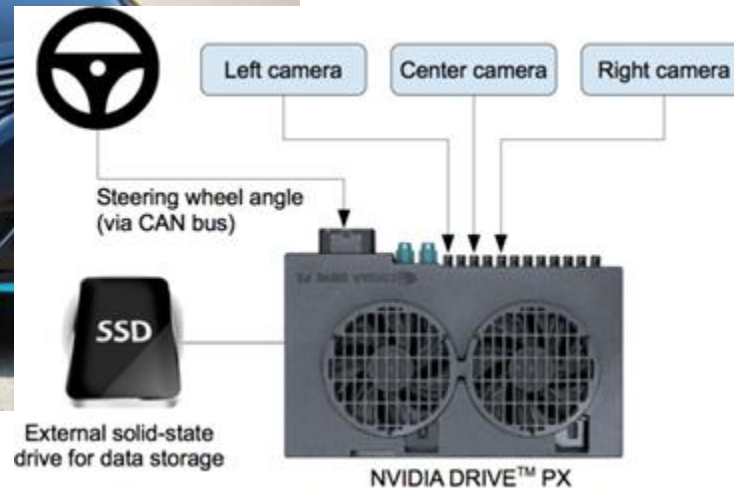
SAE LEVEL 0™	SAE LEVEL 1™	SAE LEVEL 2™	SAE LEVEL 3™	SAE LEVEL 4™	SAE LEVEL 5™
You <u>are</u> driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			You <u>are not</u> driving when these automated driving features are engaged – even if you are seated in “the driver’s seat”		
You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	

Copyright © 2021 SAE International.

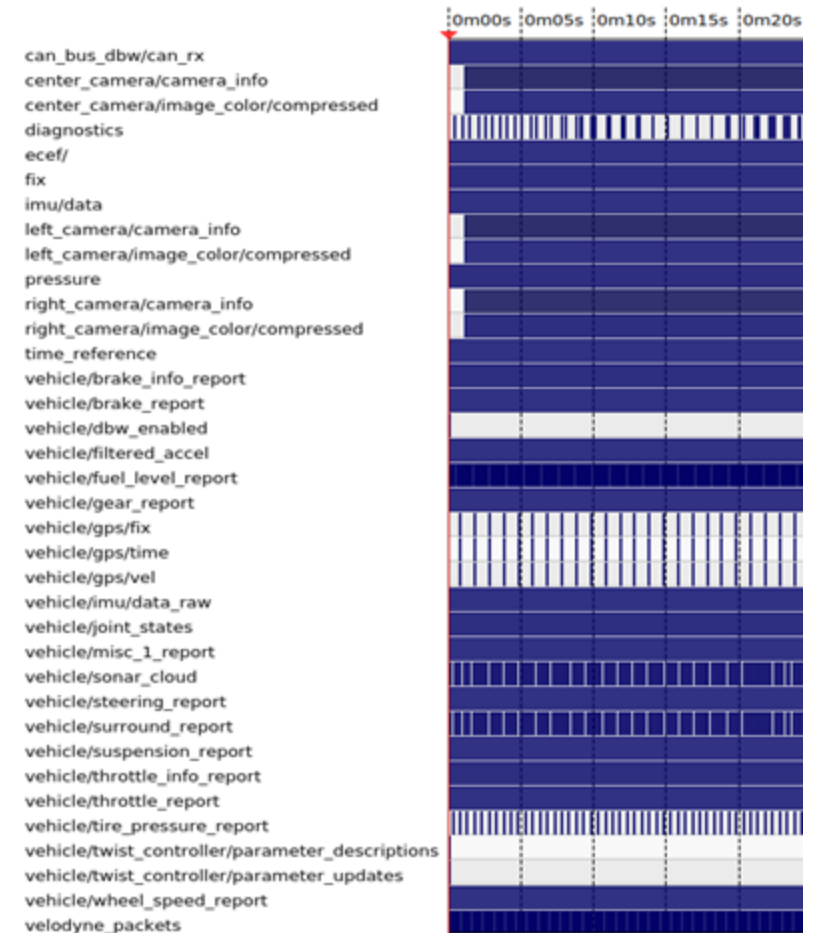
	These are driver support features			These are automated driving features	
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering OR brake/acceleration support to the driver	These features provide steering AND brake/acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met	This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering OR • adaptive cruise control	• lane centering AND • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/steering wheel may or may not be installed • same as level 4, but feature can drive everywhere in all conditions

Where is the data coming from?

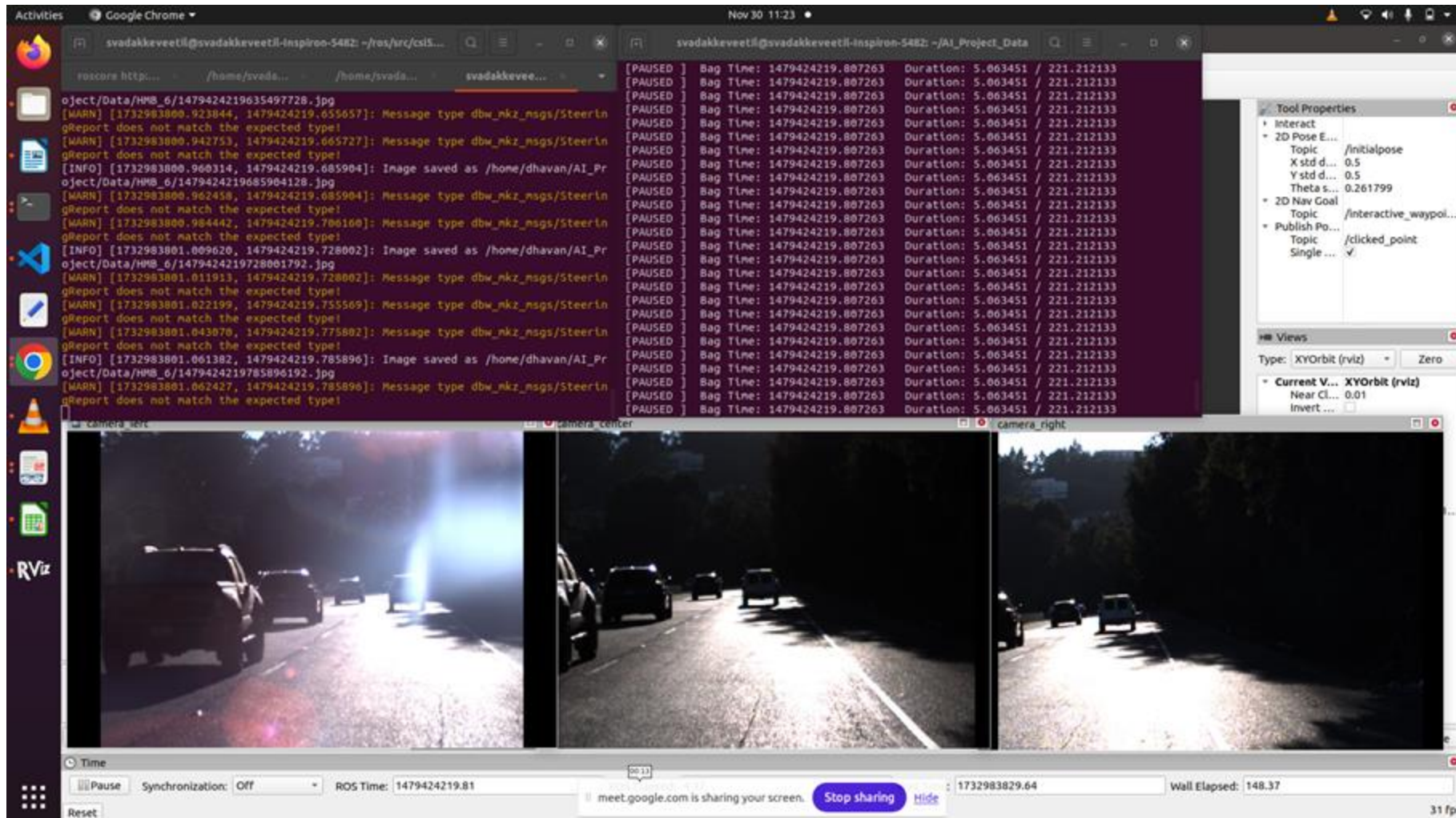
- Udacity self driving Challenge - Used NVIDIA drive to extract CAN data and camera images from cameras mounted
- Data in ROSBAG format
- Data collected from Lincoln MKZ



Information of ROSBAG



Preparing the Data - Extracting Images



ROSBAG Files



Center Camera



Steering Angle



Creates .jpg - Images



Creates .csv Data File

PreProcessing Data

Load Data



Image Preprocessing



X_train.npy Files Save
Image Data



Y_train.npy Files Save
Steering Angle

Models

- Multi CNN layer with single FC
- ReLu vs Batch Normalization
- NVIDIA model
- Large FC layers

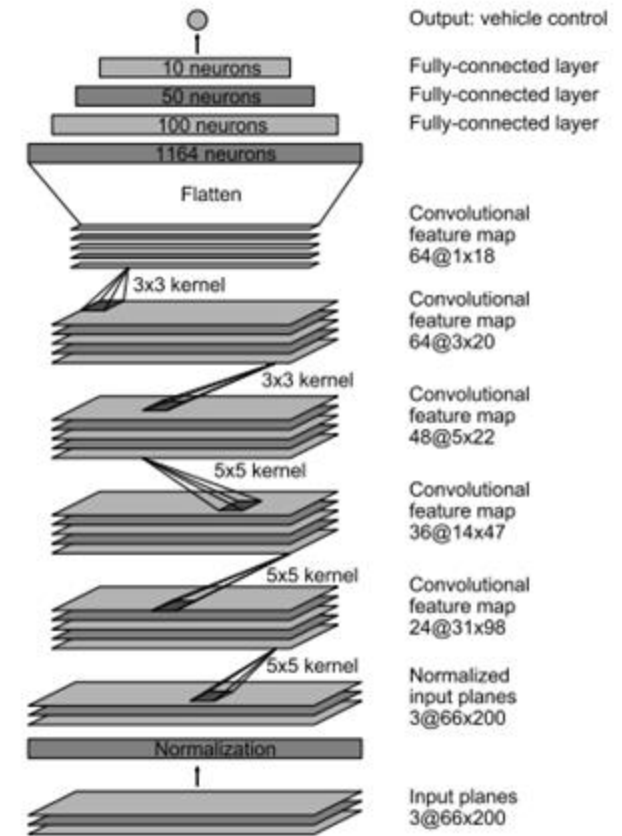


Figure 4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.

Model Training

The image displays a model training workflow across three windows:

- Terminal Window (Left):** Shows the execution of a training script. It reports data paths, shapes, and the creation of a 'sequential' model. A table lists the model's layers, their output shapes, and the number of parameters.
- File Explorer (Middle):** Displays the 'models' directory, showing three saved HDF5 model files with names like 'weights_hsv_gray_diff_ch4_comma_prelu-01-0.00955.hdf5'.
- Code Editor (Right):** Shows the Python script 'train.py' with configuration options for different model names and the training loop logic.

Model Architecture Table:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 64, 16)	4112
p_re_lu (PReLU)	(None, 48, 64, 16)	49152
conv2d_1 (Conv2D)	(None, 24, 32, 32)	12832
p_re_lu_1 (PReLU)	(None, 24, 32, 32)	24576
conv2d_2 (Conv2D)	(None, 12, 16, 64)	51264
flatten (Flatten)	(None, 12288)	0
p_re_lu_2 (PReLU)	(None, 12288)	12288
dense (Dense)	(None, 512)	6291968
p_re_lu_3 (PReLU)	(None, 512)	512
dense_1 (Dense)	(None, 1)	513

Training Summary:

- Total params: 6447217 (24.59 MB)
- Trainable params: 6447217 (24.59 MB)
- Non-trainable params: 0 (0.00 Byte)

Code Editor Snippets:

```
elif config.model_name == "comma3":  
    model = create_comma_model3()  
elif config.model_name == "comma4":  
    model = create_comma_model4()  
elif config.model_name == "comma5":  
    model = create_comma_model5()  
elif config.model_name == "comma6":  
    model = create_comma_model6()  
elif config.model_name == "comma_large":  
    model = create_comma_model_large()  
elif config.model_name == "comma_large_dropout":  
    model = create_comma_model_large_dropout()  
elif config.model_name == "nvidia1":  
    model = create_nvidia_model1()
```

```
Epoch 1/10  
31820/31820 [=====] - ETA: 0s - loss: 0.0465  
Epoch 1: val_loss improved from inf to 0.00955, saving model to /home/dhavan/AI_Project/Data/models/weights_hsv_gray_diff_ch4_comma_prelu-01-0.00955.hdf5  
/home/dhavan/.local/lib/python3.8/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via 'model.save()'. This file format is considered legacy. We recomme  
nd using instead the native Keras format, e.g. 'model.save('my_model.keras')'.  
    saving api.save_model(  
31820/31820 [=====] - 663s 21ms/step - loss: 0.0465 - val_loss: 0.0096  
Epoch 2/10  
31820/31820 [=====] - ETA: 0s - loss: 0.0243  
Epoch 2: val_loss improved from 0.00955 to 0.00949, saving model to /home/dhavan/AI_Project/Data/models/weights_hsv_gray_diff_ch4_comma_prelu-02-0.00949.hdf5  
31820/31820 [=====] - 627s 20ms/step - loss: 0.0243 - val_loss: 0.0095  
Epoch 3/10  
8853/31820 [=====] - ETA: 7:23 - loss: 0.0326CTraceback (most recent call last):
```

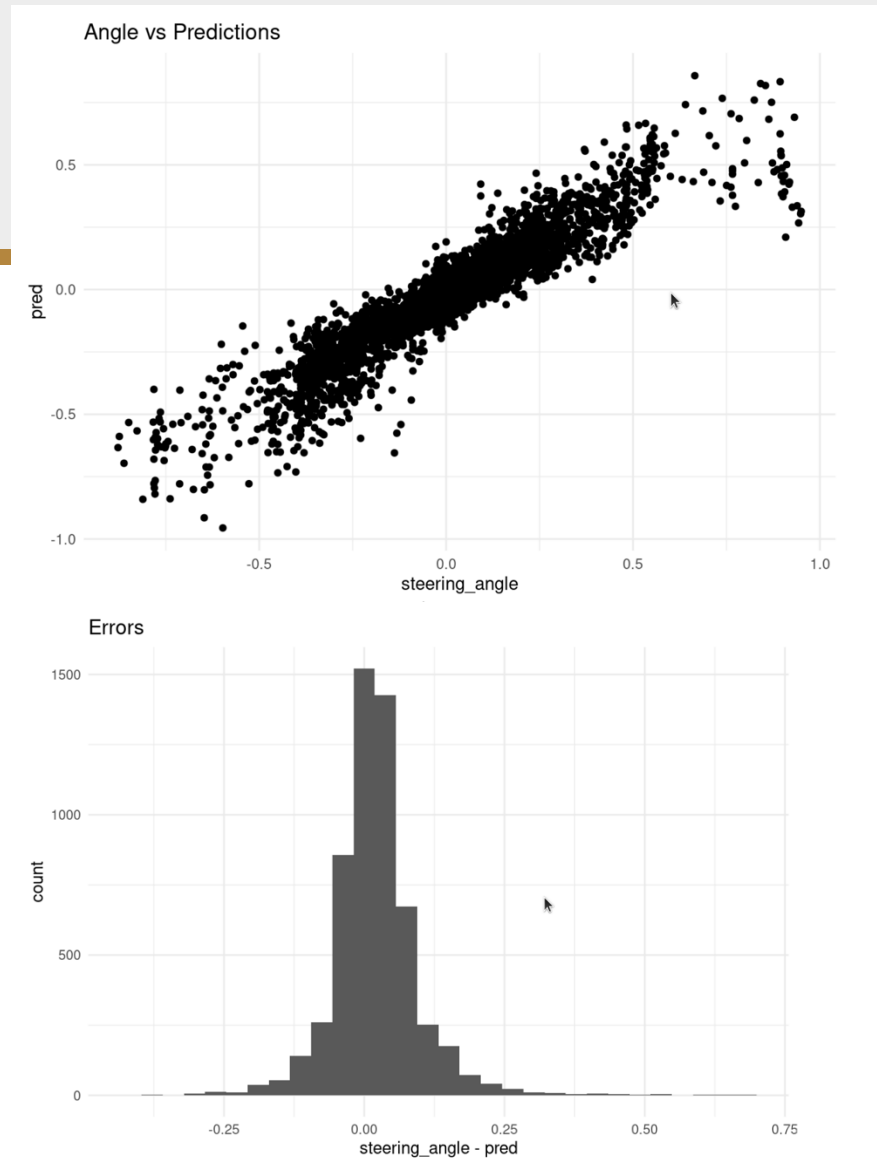

Different Models

Model	CNN Layers	Activation Layer	Fully Connected Layer/ Dense Layer	Issues
Comma Model Prelu	3	PReLU	1 (512 units)	No
Comma model Irelu	3	LReLU	1 (512 units)	No
Comma Model Batch Normalization	3	BN & ReLu	1 (512 units)	No
Comma Model ReLu	2	ReLu	1 (256 units)	No
Comma Model Prelu LSTM	3	PReLU & 1 LSTM	1 (512 units)	No
Comma Model Large	2	ReLu	1 (1024 units)	Yes (Same predictions for test set)
Comma Model Large Dropouts	2	ReLu	1 (1024 units) with dropouts 0.5	No
NVIDIA	5	ReLu	3 (100, 50, 10 units)	Yes (Same predictions for test set)

Results

Model	MSE	RMSE
Comma Model PRelu	0.00864	0.09295160031
Comma model LReLU	0.03348	0.1829754082
Comma Model Batch Normalization	0.05733	0.2394368393
Comma Model 6	0.00662	0.08136338243
Comma Model Prelu LSTM	0.01642	0.1281405478
Comma Model Prelu Large with Drop out	0.01307	0.1143241007
Rambo	0.00324	0.05692099788

Comma Model with Prelu as activation layer performed the best comparing the RMSE values



Visualization

- Model 6 with multiple CNN and a single dense layer and ReLu activation performed the best
- Output visualization of estimated vs actual was created using PyGame



Conclusion

- Steering wheel angle recognition using image processing and neural networks.
- NN models based on CNN & dense layers
- ReLu & Batch normalization used as activation layers
- 6 model evaluated using MSE and RMSE

Next Steps

- Analyze the NVIDIA and using large dense layers
- Hyperparameter tuning for different filters and batch size
- Data augmentation to include the effect of shadow and daylight savings
- Evaluate the performance of using transfer learning and LSTM

Thank You
Any Questions?