# 1 Evaluation

In this chapter, the workflow explained in last chapter is evaluated and results are presented.

Before showing the evaluation, it is necessary to define training and testing conditions that can be easily used by others to verify the results.

We have three datasets that can be used for training and evaluation.

1. Dataset 1 - Contains 100,000 raw data. It is collected in no traffic environment, doing straight driving without any sudden turning. The data is using San Francisco map and driven during afternoon. This dataset has only data representing centre camera pointed ahead, parallel to the ground and right camera pointed to the ground at an angle $20^{\circ}$. The control commands include acceleration, throttle, braking, and steering angle values.

2. Dataset 2 - Also contains 100,000 raw data. It is, however, collect with traffic where the cars stop at signal intersections for a longer time than dataset 3. This dataset is also collect in San Francisco map and during afternoon. It contains a centre camera, right camera like dataset 1, left camera similar to right camera by pointing at an angle $20^{\circ}$ to the ground, depth camera sensors placed at centre, left and right just like RGB cameras. The control commands are same as dataset 1.

3. Dataset 3 - Contains 270,000 raw data. It is collected while driving around San Francisco. About 200,000 data is collected while driving in the afternoon. About 20,000 in different weather and light conditions. About 50,000 entries are collected in a different circular circuit map called CubeTown. In addition to RGB and depth cameras distributed just as dataset 2, a segmentation camera is kept next centre RGB camera facing forward, and a radar sensor just in front of the car near the hood also facing forward.

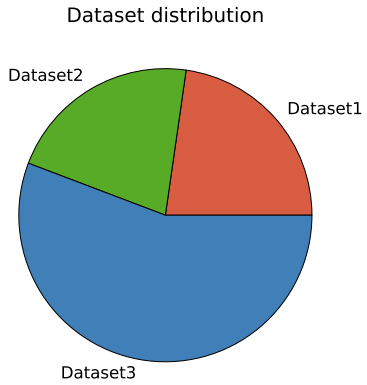



Figure 1.1: Datasets distribution

| time(in 24 hrs standard) | Morning | Afternoon | Evening |
|---|---|---|---|
| | 7:30 | 15:30 | 18:30 |

Table 1.1: Time of the day

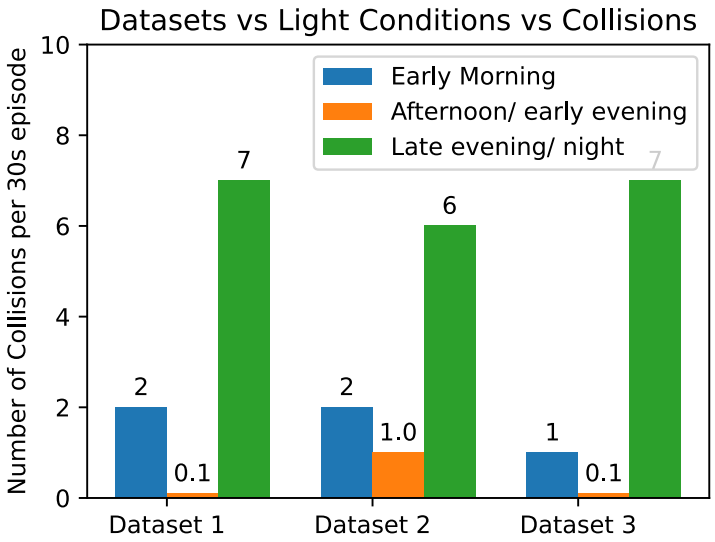



Figure 1.2: Datasets vs Light Conditions vs Number of Collisions

## Evaluation setup

While evaluating, a testing parameter *episode* is used. Each episode lasts 30 seconds. A timer is started for 30 seconds and the model is tested for collisions. If a collision happens, the time at which collision happened is noted.

As supervised learning is used, the models have to be tested/validated with unknown data to determine its capability. Hence the datasets are split 80-20. Meaning 80% is train data and 20% validation data. The optimizer *Adam* takes the 20% data to test the trained model. Training data leads to training loss and test data to validation loss.

## 1.1 Determine the best lighting conditions to test the model

All three datasets are used. The test is conducted in San Francisco map without traffic option switched ON. By varying the light conditions to morning, afternoon and evening, we observe how light influences the prediction of output. Only steering angle is predicted and a steady velocity of 3 meter per second is used. An episode length of 30s is used. When a collision is observed, the time of collision and the count are noted down.

It is seen from figure 1.2 that afternoon time provides the best light conditions for all the three datasets. Dataset 1 and 3 perform equally across the three lighting conditions.

If the percentage of number of collisions, as shown in figure 1.3, is calculated, dataset 3 performs the best among the datasets for morning and afternoon part of the day.

### 1.1.1 Datasets performance during afternoon if traffic is enabled

All three datasets are again used. The time is fixed at 15:30. The traffic is toggled ON.

From figure 1.4, we can observe that all three datasets do well even in traffic. However, it is surprising

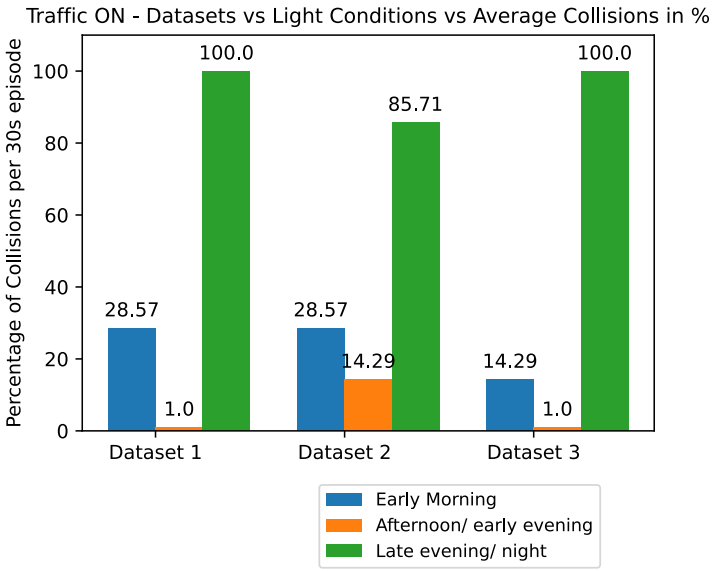



Figure 1.3: Traffic ON - Datasets vs Light Conditions vs Average number of Collisions

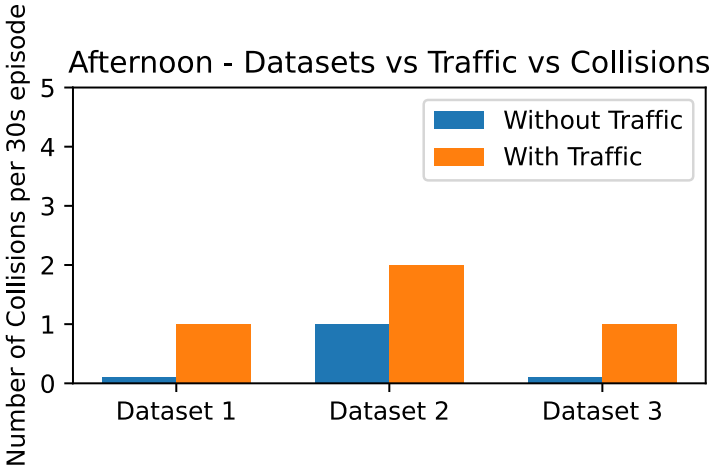



Figure 1.4: Afternoon - Datasets vs Traffic vs Number of Collisions

to see dataset 1 which had no traffic while the dataset was collected, performs remarkably well when driven in traffic. Since dataset 2 shows higher number of collisions, it is eliminated.

## 1.2 Regression task - Determine which dataset and activation function to use

### 1.2.1 Tanh activation function

Since acceleration and steering values in LGSVL range from *-1* to *1*, *tanh* activation is chosen for the first part of regression training model. The last output dense layer predicts acceleration and steering using this activation function. As seen in figure 1.5, a continuous loss function such as mean squared error is chosen to complement tanh activation function. It is seen that both dataset 1 and 3 show similar training characteristics and some overfitting behaviours. Because of this overfitting behaviour, while evaluating, the models do well even when introduced to traffic. However, they do not show consistent performance.

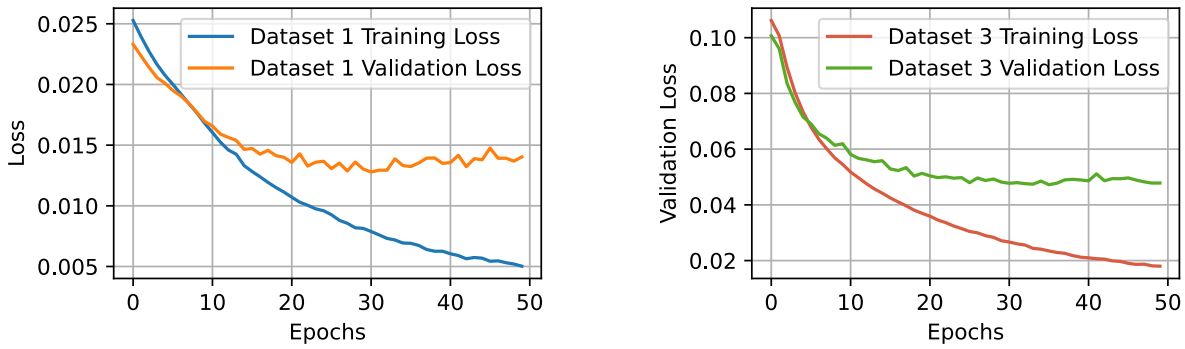



Figure 1.5: Datasets 1 vs 3 - Acceleration and Steering using Tanh activation and MSE loss functions.

| Criteria | Yes/No |
|---|---|
| Lane keeping/Drive straight | Yes |
| Gradual acceleration increase | Yes |
| Braking behaviour observed | Yes |
| Smooth steering control at high speed(10m/s) | Yes |
| Smooth steering control at turnings | Yes |
| Doesn't stop at random places | No |

Table 1.2: Dataset 1 - How the model evaluates to different criteria.

### 1.2.2 Sigmoid activation function

The acceleration values are split into positive and negative values. Instead of negative values an another variable called *braking* is introduced. Since we know negative acceleration values mean, braking is active. Using sigmoid as activation function and mean squared error as loss function, a training is conducted for both datasets 1 and 3. As seen in figure 1.6, the training loss for dataset 1 is close to zero. And dataset 3 losses are slightly higher and close its values in figure 1.5.

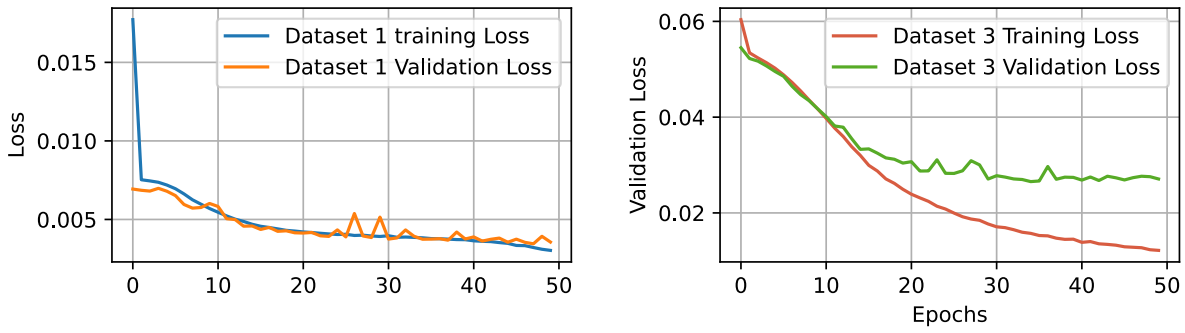



Figure 1.6: Datasets 1 vs 3 - Acceleration and Steering using Sigmoid activation and MSE loss functions.

## 1.3 Classification task - which dataset and loss function to use

Since regression training models shows enough capability and since the acceleration are basically two discrete values, it would be worthy to try as classification task. Of course steering being continuous

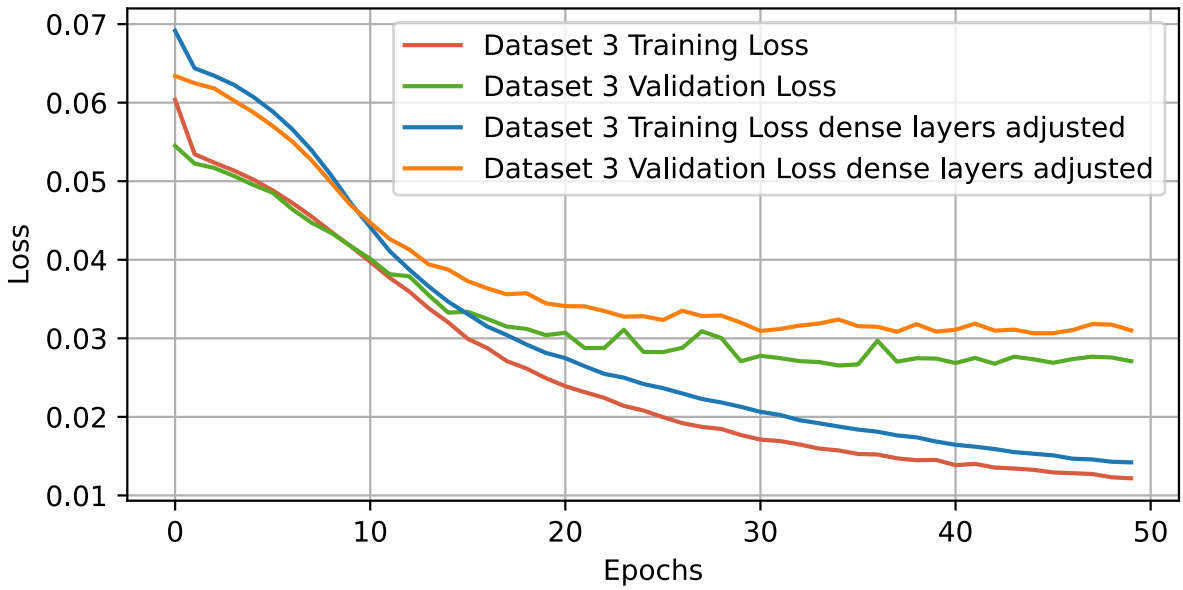



Figure 1.7: Dataset 3 - Acceleration and Steering using Sigmoid activation and MSE loss functions with adjusted dense layers.

uses MSE.

### 1.3.1 Binary crossentropy Loss function

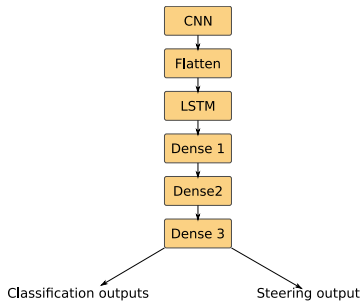
Upon using *binary crossentropy* as loss function, we can see in figures 1.8 and 1.9, exhibit higher loss and the bias above 50% in training. Sure enough when evaluating, the vehicle either stays stationary or moves forward for a few seconds and halts completely.

#### Control commands distribution

This biased behaviour seen in binary crossentropy needs more investigation. Looking at the labelled output data a distribution piechart is plotted. In the chart 1.10, we can see that a state *no action* dominates the datasets. Dataset 1 has only a small portion for acceleration and even smaller for braking. With this in mind, dataset 3 is collected with an attempt to increase acceleration and braking values share. However, since the vehicle most times has to drive straight with *no action*, this state even dominates in dataset 3.

### 1.3.2 Categorical Crossentropy Loss function

So now we know that this dominant state *no action* needs a separate label if classification task has to be continued. After creating the label, we would then have three labels – acceleration, braking and no action. Hence, we use a new classification loss function called *Categorical crossentropy*. This loss function classifies model into each category. However, since no action dominates the dataset, we can assume that this category will dominate the training. With this in mind, dataset 3 is used. For training a model as shown in figure 1.11 is designed and its result is seen in figure 1.12. Interestingly, the training loss curve follows the classification loss as it dominates the model. Steering loss however, after epoch 32 starts to increase. This is strange behaviour as it is not any time before in training. Upon evaluation, the steering was



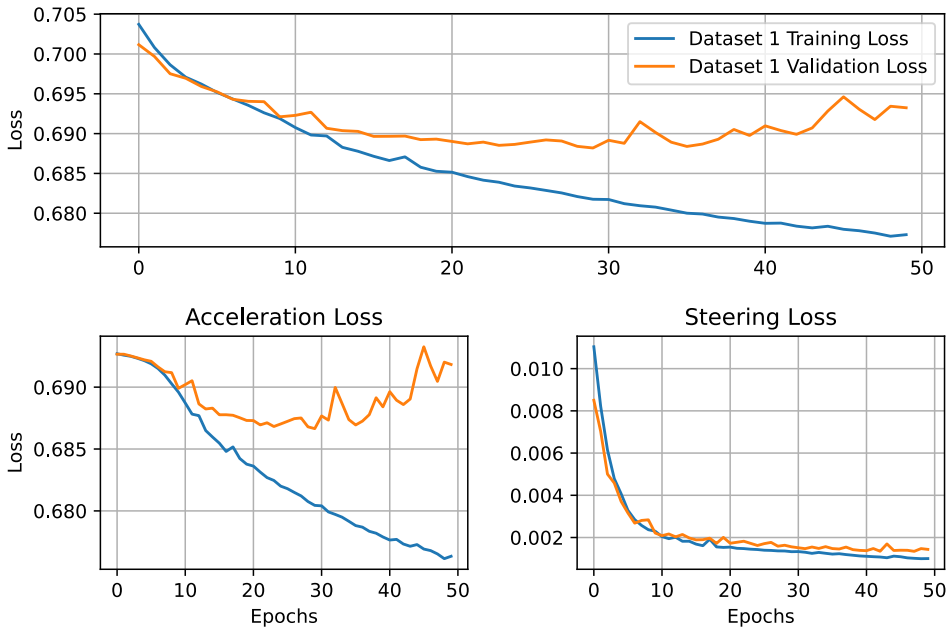



Figure 1.8: Dataset 1 - Binary Crossentropy

all over the place and acceleration was not stable at all, resulting in many collisions.

The cause for this behaviour is investigated and it is found that since both classification task outputs and steering output inherit the same dense layer, sufficient learning is not observed. The dense layer is perhaps overwhelmed. This paves way nicely to the next design iteration.

### 1.3.3 LSTM vs Non-LSTM

In the next stage of evaluation, acceleration in addition to steering is predicted. Acceleration in LGSVL contains positive values for forward throttle and negative values for braking.

Acceleration is relative to previous frame. Hence it is necessary to include past frames information while predicted it. LSTM is used for this purpose. When non-LSTM model is used to predict acceleration, it only predicts for the current frame which often results in vehicle being stationary.

For our setup, we choose a $timestep = 15$. That means acceleration of current time frame is predicted using previous 14 time frames.

#### Determining the optimal LSTM output units

In the table 1.3, we can see that for 100 LSTM output units, though the trainable parameters(parameters that can be trained during backpropagation) is quite high compared to other units, the training time is the least. Moreover on evaluation, it is observed that acceleration prediction is relatively good compared to other units' models. Hence, a LSTM unit of 100 is chosen.

#### Categorical Crossentropy - splitting at the dense layers

To alleviate some of the burden the second dense layer is split into two separate dense layers;one for classification outputs and other for steering as show in figure

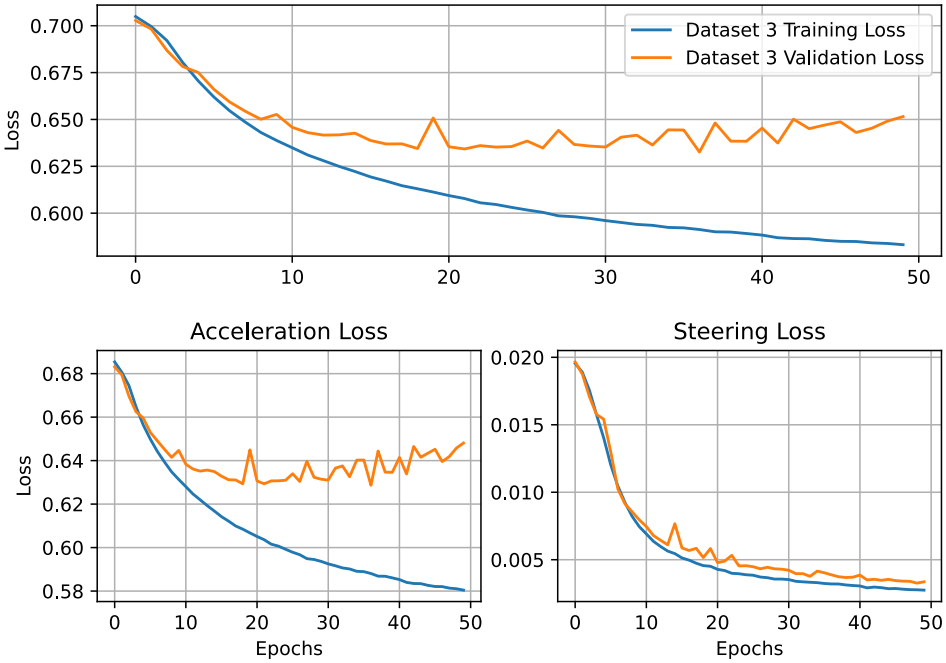



Figure 1.9: Dataset 3 - Binary Crossentropy

| LSTM Output Units | Trainable Parameters | Processing time needed |
|---|---|---|
| 20 | 20000 | 1hr 44m |
| 60 | 61000 | 1hr 42m |
| 100 | 434000 | 1hr 40m |

Table 1.3: LSTM Output Units vs Trainable Parameters vs Training time

1.13. The result 1.14, stops the strange steering loss increase. Upon evaluation, this model shows better steering control but still acceleration is not stable and consistent.

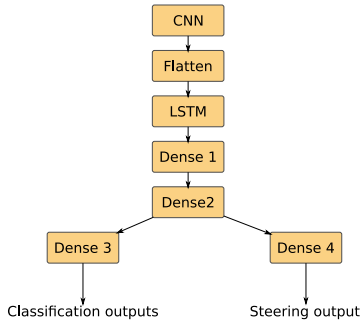



Figure 1.13: Split at the second dense layer

**Categorical Crossentropy - splitting at the LSTM layers**

Now what happens if the model is split further at LSTM layer as shown in figure 1.15. The main aim here is to see if steering control improves and is stable for considerable acceleration prediction. From figure 1.16, the steering loss gets a marginal gain. Still at evaluation, the trained models stops at random places, and steering control not stable at higher acceleration predicted values. Hence the predicted acceleration value is reduced by 50-70% and fed to the controller. Sure enough the vehicle exhibits stable movements.

**Categorical Crossentropy - Using two different NN for acceleration and Steering**

It can be deduced that optimised weights have an important role on how it influences steering and acceleration prediction. The model is now split as two different neural networks(NN) as seen in figure 1.17. Though the result 1.18 looks similar to 1.16, the model predicts stable, consistent acceleration values.

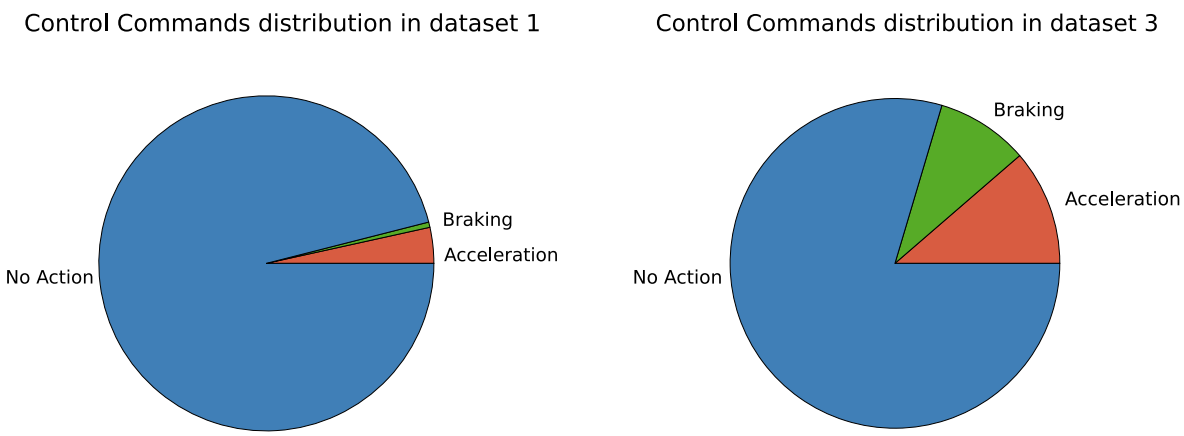



Figure 1.10: Datasets 1 vs 3 Control commands distribution

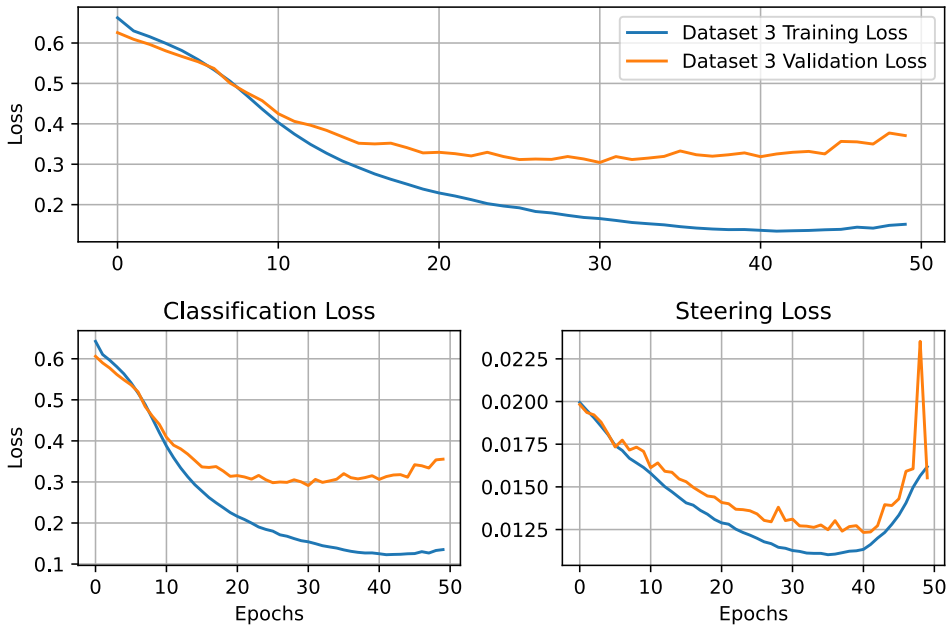



Figure 1.12: Dataset 3 - Categorical Crossentropy - Basic model

It even exhibits occasional turning behaviours at junctions. When it is exposed to traffic, the model does well to navigate, brake, and accelerate. It is not perfect as one would hope. Sunlight still plays a major role. Sunlight makes the model do random, strange crashes. Shadows and low-light however, seems like to not influence much in a few number of trials.

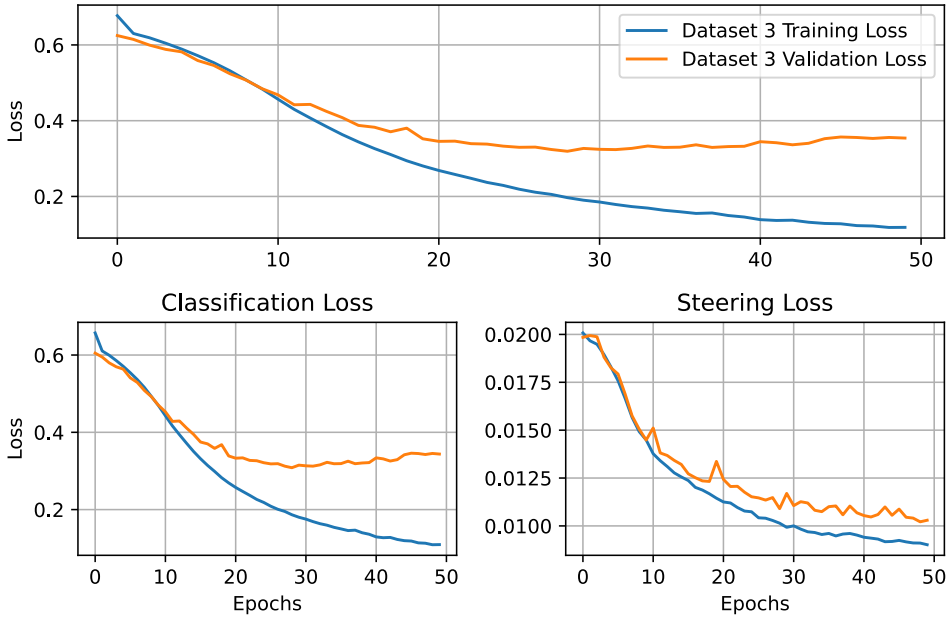



Figure 1.14: Dataset 3 - Categorical Crossentropy - Separate dense layers for classification and steering

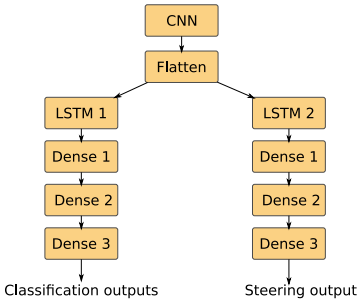



Figure 1.15: Split at the LSTM layer

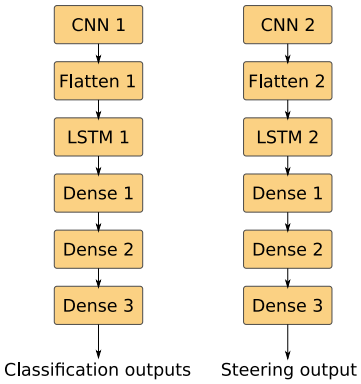



Figure 1.17: Separate NN training model

An overview of steering losses across different NN changes is shown in figure 1.19.

### 1.3.4 Velocity

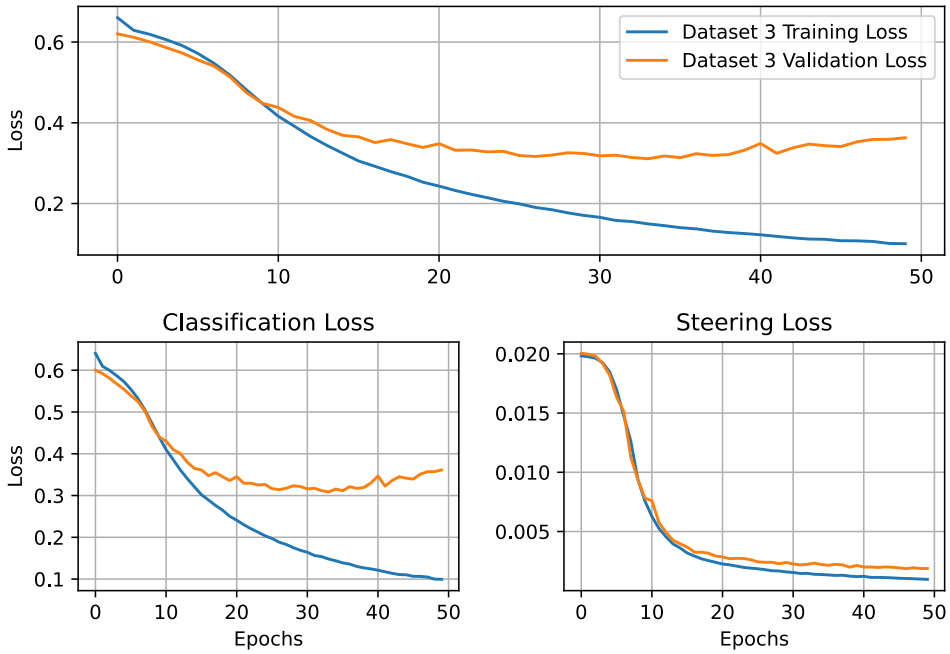



Figure 1.16: Dataset 3 - Categorical Crossentropy - Separate LSTM layers for classification and steering

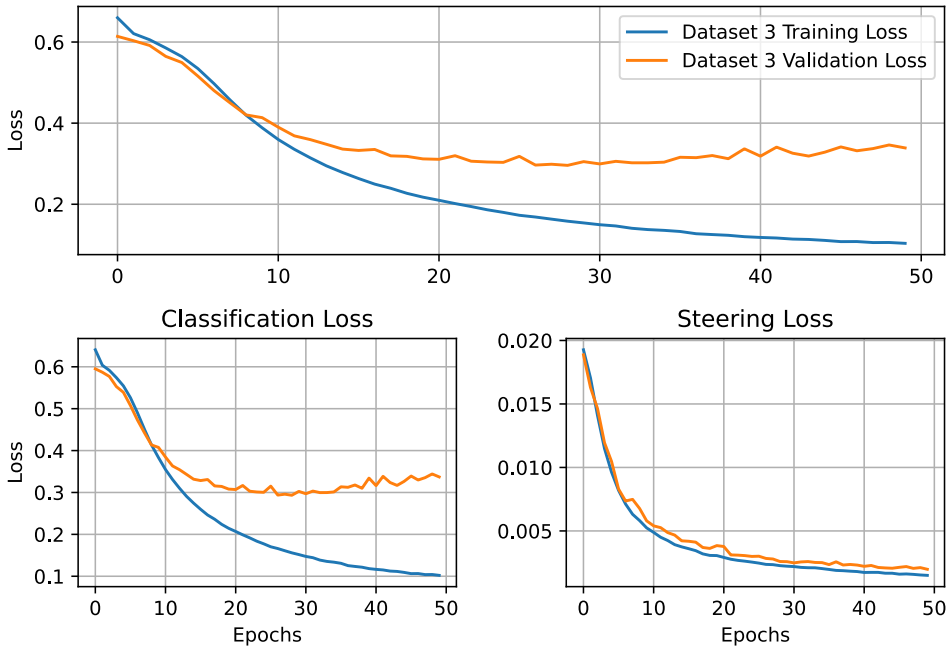



Figure 1.18: Dataset 3 - Categorical Crossentropy - Separate neural network for Classification and Steering

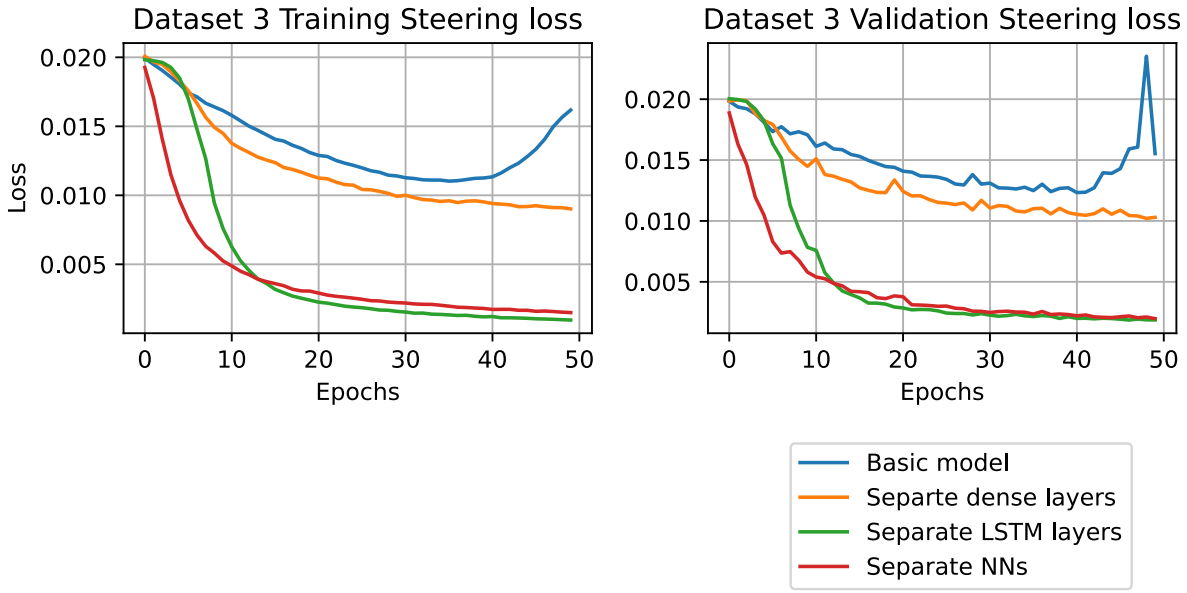



Figure 1.19: Dataset 3 - Categorical Crossentropy - Steering command loss comparison

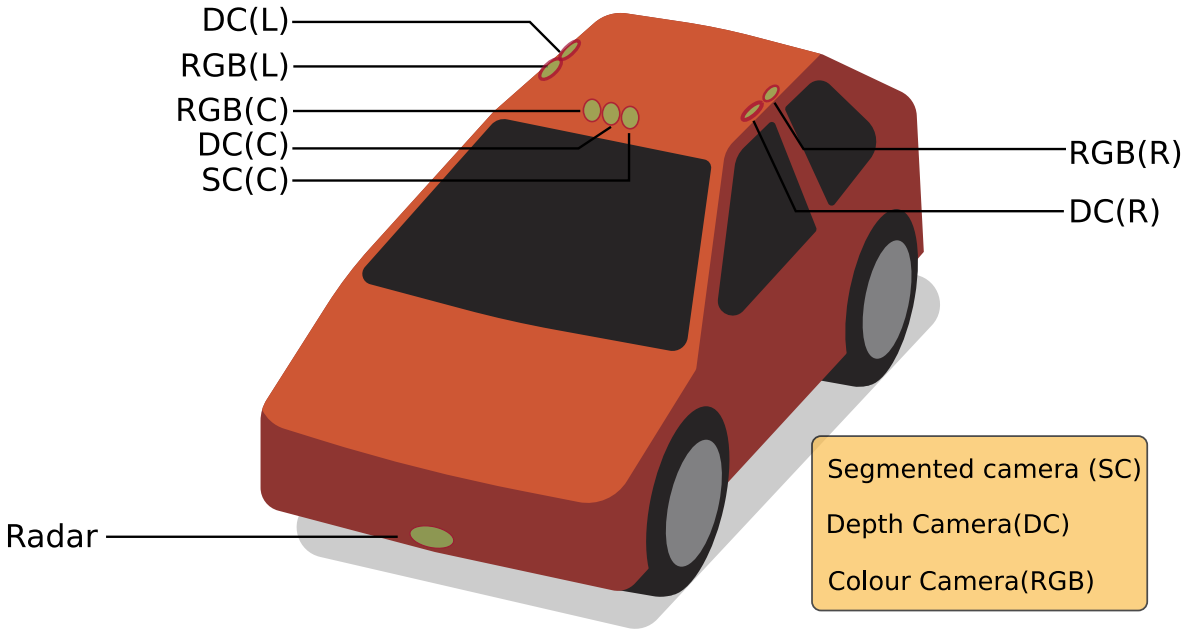



Figure 1.20: Sensor Constellation

# List of Figures

# References