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.

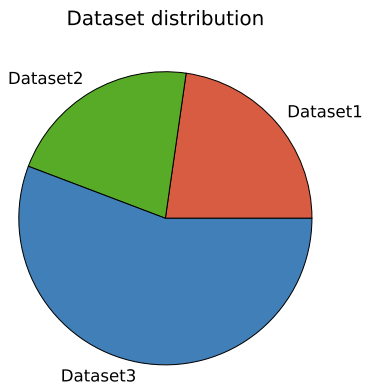

Figure 1.1: Datasets distribution

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

1. Dataset 1 - Contains 100,000 raw data as seen in figure 1.1. 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 $20°$. The control commands include acceleration, throttle, braking, and steering angle values.

2. Dataset 2 - Also contains 100,000 raw data. It is, however, collected 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 $20°$ 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 to centre RGB camera facing forward, and a radar sensor just in front of the car near the hood also facing forward.

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

Table 1.1: Time of the day

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.

Also till a single dataset is chosen, all datasets are of equal data entries.

1.1 Determine which datasets and 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(see table 1.1).

The steering angle is a continuous value ranging between -1 and 1(negative values to turn left and positive values to turn right), a continuous loss function has to be used. Because of that *mean square error*(MSE) as loss function is chosen. 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 number of collisions are noted down.

It is seen from figure 1.2(a) 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 with traffic toggled ON, as shown in figure 1.2(c), 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 1.2(b), we can observe that all three datasets do well even in traffic. However, it is surprising to see dataset 1 which had no traffic while the dataset was collected, performs remarkably well when driven in traffic.

1.1.2 Observations

1. All 3 datasets do well at afternoon time of the day.
2. Predicting only steering angle with MSE as loss function works as seen from 1.2.
3. Dataset 2 shows higher number of collisions. So it is better to avoid for further analysis.

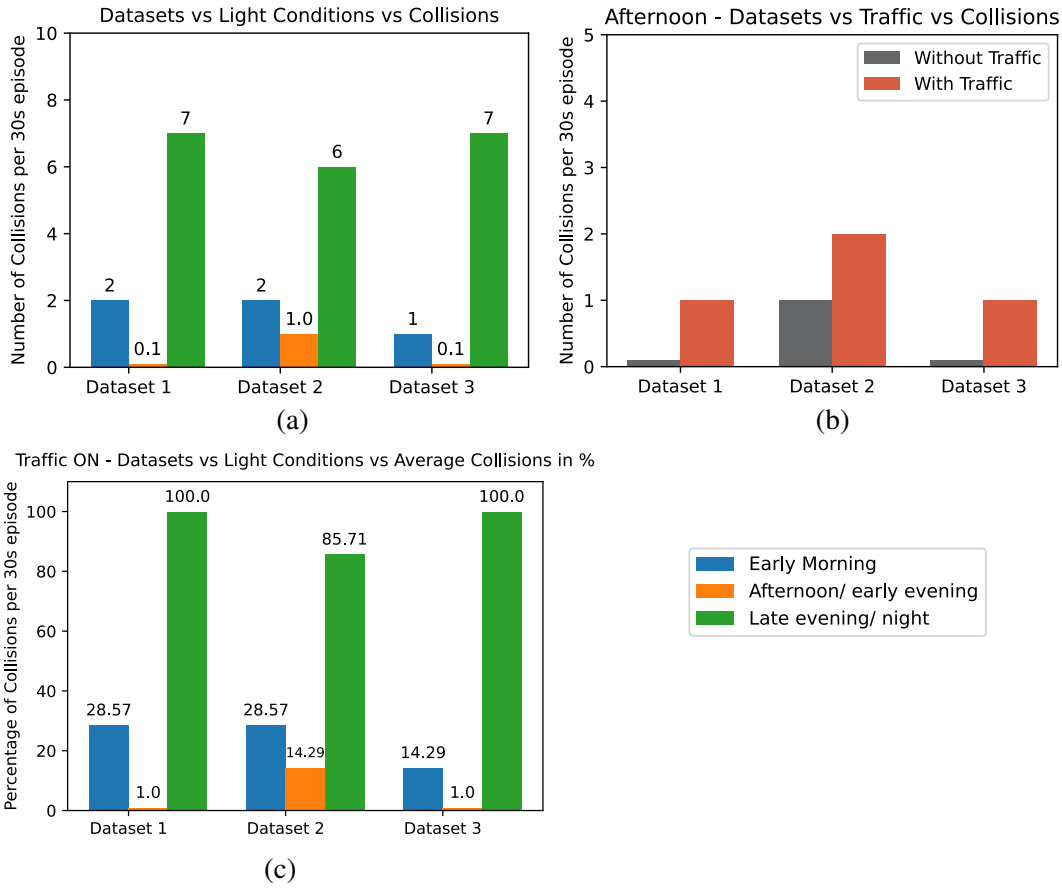

Figure 1.2: a) Datasets vs Light Conditions vs Collisions. b) Afternoon - Datasets vs Traffic vs Number of Collisions. c) Average number of collisions in percentage

1.2 Acceleration - Determine which activation and loss functions to use

1.2.1 Tanh as activation and MSE as loss functions

Since acceleration and steering values in LGSVL range from *-1* to *1*, *tanh* activation function is selected as the output dense layer activation function.

A set of criteria are listed and the trained model is evaluated based on these conditions. From table 1.2 it can be observed that dataset 1 outperforms dataset 3 in most of the conditions. Dataset 1 retains good steering control at high speeds and turning, but acceleration skews steering angle when traffic is switched ON. Dataset 3 when evaluated stops completely after moving a few metres. This causes difficulty in evaluating according to the criteria. Hence it is assumed that this dataset fails to meet the criteria.

One of the reasons as to why dataset 3 fails could be because the losses are much higher than dataset 1 and the validation loss starts to overfit too quickly in the training(see figure 1.3).

1.2.2 Sigmoid as activation and MSE as loss functions

The acceleration values are split into positive and negative values. Instead of negative values an another variable we will call as *braking* is introduced. Negative acceleration values mean braking is active. Using this knowledge, sigmoid as activation function and mean square error as loss function, a training is conducted for both datasets 1 and 3.

Criteria(Tanh/MSE)	Dataset 1	Dataset 3
Lane keeping/Drive straight	Yes	No
Gradual acceleration increase	Yes	No
Smooth braking behaviour observed	Yes	No
Smooth steering control at high speed(10m/s)	Yes	No
Smooth steering control at turnings	Yes	No
Detects traffic as dynamic objects	Yes	No
Navigates traffic smoothly	No	No
Doesn't stop at random places	No	No

Table 1.2: Tanh/MSE - How the model evaluates to different criteria

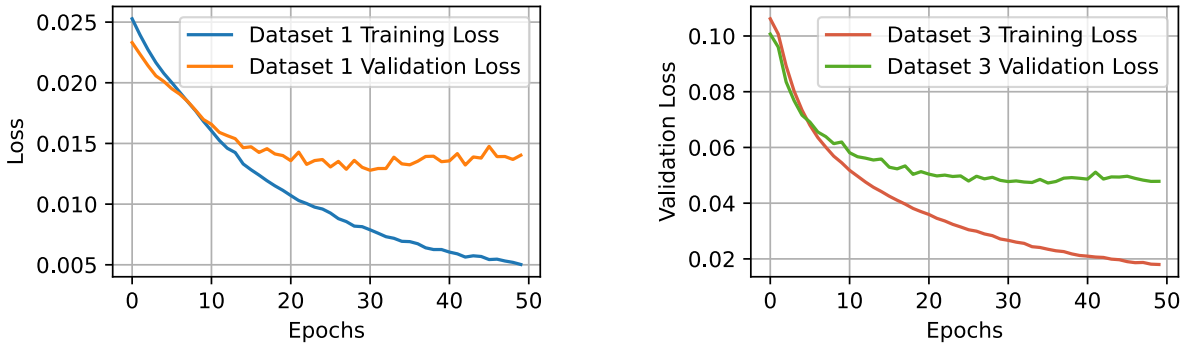

Figure 1.3: Datasets 1 vs 3 - Acceleration and Steering using Tanh activation and MSE loss functions.

Criteria similar to table 1.2 are put to test with this activation and loss function. As seen in table 1.3, both datasets fail to meet the conditions. During evaluation, both datasets trained models, accelerate to a huge velocity such as 40m/s and the steering cannot keep up, resulting in collisions.

Looking into their losses (see figure 1.4), don't really explain much. Though this needs more investigation, for now we assume this activation or loss function is not suitable. 1.3.

Criteria(Sigmoid/MSE)	Dataset 1	Dataset 3
Lane keeping/Drive straight	No	No
Gradual acceleration increase	No	No
Smooth braking behaviour observed	No	No
Smooth steering control at high speed(10m/s)	No	No
Smooth steering control at turnings	No	No
Detects traffic as dynamic objects	No	No
Navigates traffic smoothly	No	No
Doesn't stop at random places	No	No

Table 1.3: Sigmoid/MSE - How the model evaluates to different criteria

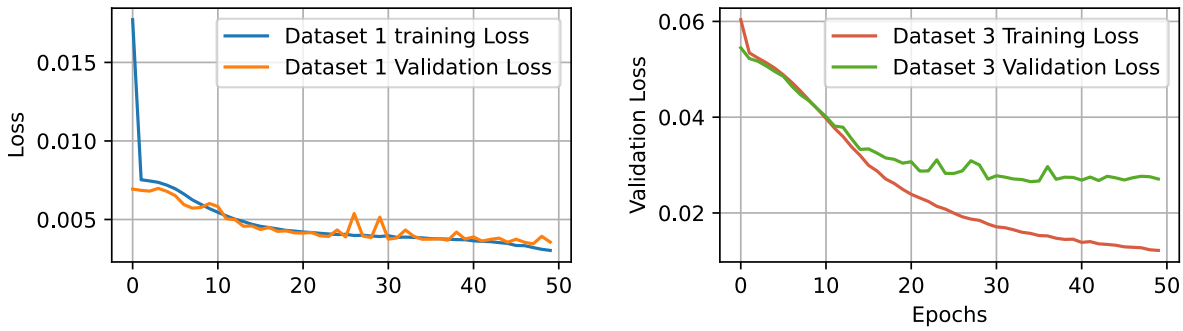

Figure 1.4: Datasets 1 vs 3 - Acceleration and Steering using Sigmoid activation and MSE loss functions.

1.2.3 Softmax as activation and Binary crossentropy as loss functions

Both tanh and sigmoid activation functions couldn't give stable results to continue pursuing with those parameters. Hence it is necessary to consider other functions which may suit our needs.

Since acceleration are basically two discrete values, it would be worthy to try the training as classification task. The goal of a classification task model is to classify to which category the prediction belongs to. In our case, acceleration or braking. Hence *softmax* activation function is needed. As loss function *binary crossentropy* is used to classify as binary classes. Of course for steering angle, being continuous, MSE is preferred.

From table 1.4, both datasets don't do well. Dataset 1 doesn't start at all as braking class dominates the evaluation whereas dataset 3 starts to move the car but resorts to brake indefinitely after a few metres of driving. This behaviour necessitates further analysis into the datasets.

Criteria(Softmax/Binary crossentropy)	Dataset 1	Dataset 3
Lane keeping/Drive straight	No	Yes
Gradual acceleration increase	No	No
Smooth braking behaviour observed	No	No
Smooth steering control at high speed(10m/s)	No	No
Smooth steering control at turnings	No	No
Detects traffic as dynamic objects	No	No
Navigates traffic smoothly	No	No
Doesn't stop at random places	No	No

Table 1.4: Softmax/Binary crossentropy - How the model evaluates to different criteria

Control commands distribution

When the datasets are analysed for patterns of different states – acceleration and braking, distribution chart(figure 1.5) reveals that in addition to acceleration and braking states, there is a third state called *no action* where the vehicle does absolutely nothing. In fact this state dominates in both datasets. Because of this, the binary crossentropy obviously fails to meet the conditions.

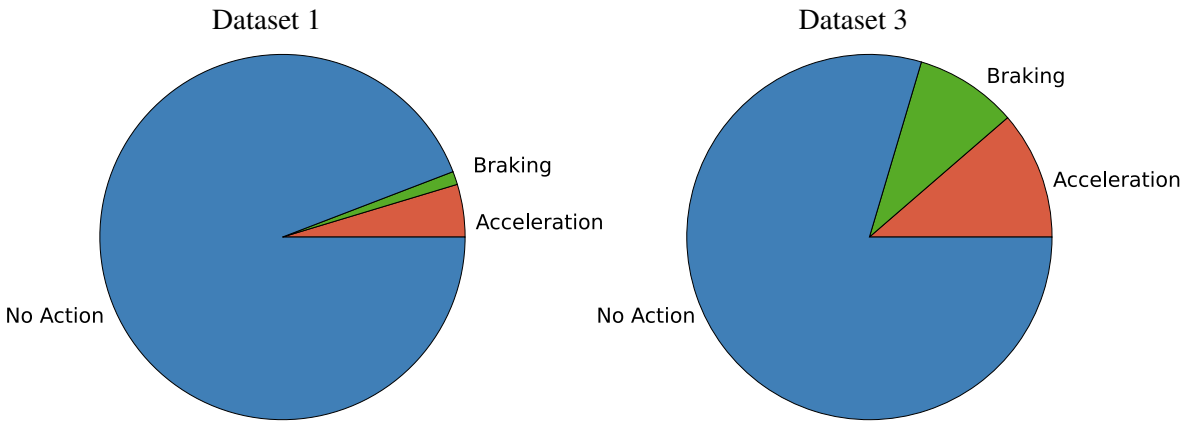

Figure 1.5: Datasets 1 vs 3 control commands distribution

1.2.4 Softmax as activation and Categorical crossentropy loss functions

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.

Criteria(Softmax/Categorical crossentropy)	Dataset 1	Dataset 3
Lane keeping/Drive straight	No	Yes
Gradual acceleration increase	Yes	Yes
Smooth braking behaviour observed	No	Yes
Smooth steering control at high speed(10m/s)	No	No
Smooth steering control at turnings	No	No
Avoids colliding into static objects	No	No
Detects traffic as dynamic objects	Yes	Yes
Navigates traffic smoothly	No	No
Doesn't stop at random places	No	No

Table 1.5: Softmax/Categorical crossentropy - How the model evaluates to different criteria

From table 1.5, dataset 1 fails in most conditions and dataset 3 though performs well in 4 out of 9 conditions, shows bad steering behaviour at traffic or high speeds. If the acceleration is controlled manually, the model reacts better and adapts itself.

Since no action state dominates, it is necessary to continue as a classification task. Also dataset 1 has only a small portion for acceleration and even smaller for braking. Accordingly, dataset 3 is collected with an attempt to increase acceleration and braking values share. However, since the vehicle most times has to drive straight, *no action* state even dominates in dataset 3.

The important condition in a classification task is to have balanced classes. Unfortunately in our case, this balance is not achieved. With this limitation, the evaluation is carried forward.

1.2.5 Observations

1. Steering angle uses tanh activation and MSE loss functions

2. Classify acceleration prediction as classification task which means softmax as activation.
3. Since acceleration carries three states, categorical crossentropy as loss function.
4. Dataset 3 has more of acceleration and braking states than dataset 1. So dataset 3 is preferred.

1.3 Predicting acceleration - categorical crossentropy

Now that the basic criteria for training is fixed such as which dataset, activation, and loss functions, we can move ahead and optimise the predicted classes by tuning the neutral network.

LSTM vs Non-LSTM

Before going into tuning the neural network, it is necessary to tell that acceleration prediction needs temporal information; meaning decision to drive slower or faster depends on the previous, historical frames. LSTM is used for this purpose. When non-LSTM model is used to predict acceleration, it only predicts for the current frame(doesn't provide past frames information) which often results in vehicle being stationary. For our setup, we choose a $timestep = 15$. That means acceleration of current time

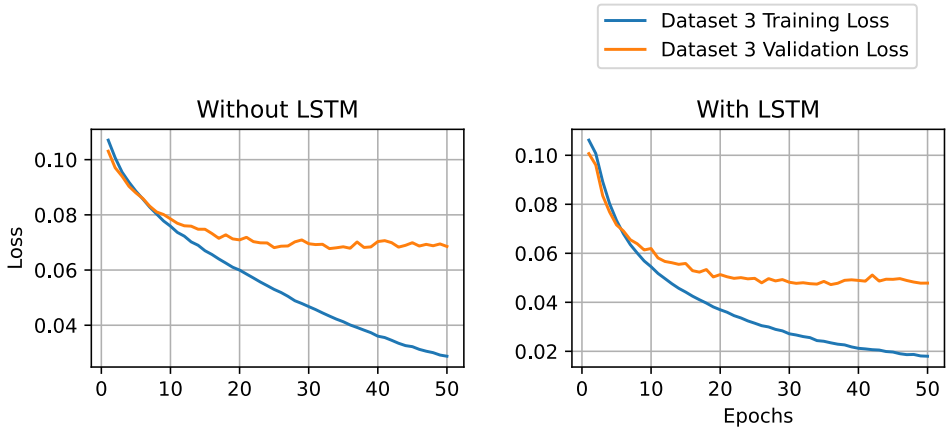

Figure 1.6: Datasets 3 - No LSTM vs LSTM comparison

frame is predicted using previous 14 time frames.

Determining the optimal LSTM output units

In Keras, the LSTM units refer to the dimension of hidden state vector *h* that is the state output from RNN cell. The table 1.6, compares different unit values and the number of trainable parameters(weight that can be trained during backpropagation) at this LSTM layer. In our case, it means that for a time series of 15, there will be 15 *cell states*, 15 *hidden states*, and 15 *outputs* each of vector size defined by the units in the table such as 20, 60 or 100.

Upon evaluation with these different units, 100 output units though has the highest trainable parameters for this layer alone, retains more information needed for training the model. Hence, a LSTM unit of 100 is chosen.

LSTM Output Units	Trainable Parameters(ca.)	Processing time needed
20	20000	1hr 44m
60	61000	1hr 42m
100	434000	1hr 40m

Table 1.6: LSTM Output Units vs Trainable Parameters vs Training time

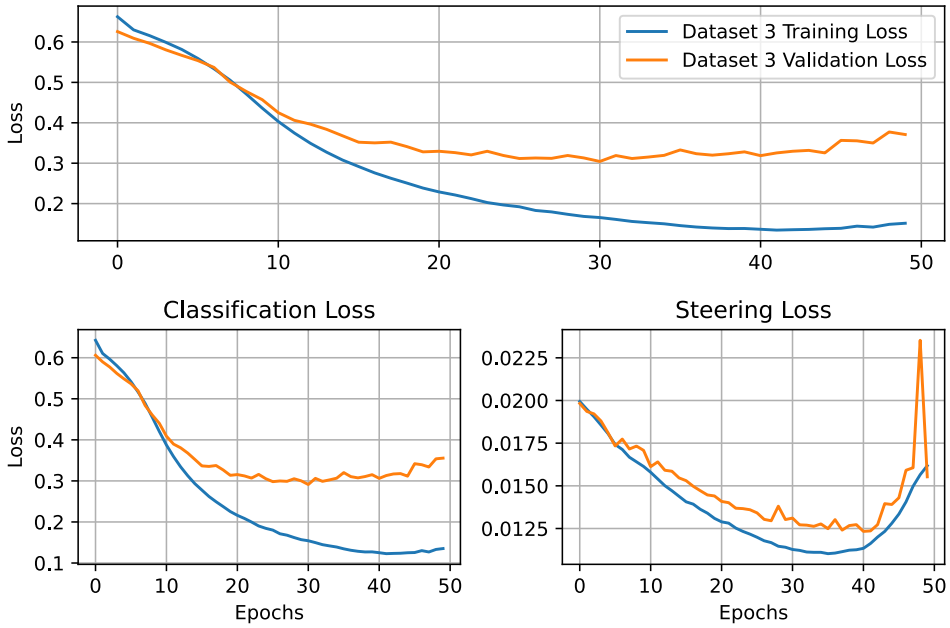

Figure 1.8: Basic model

1.3.1 Basic Model

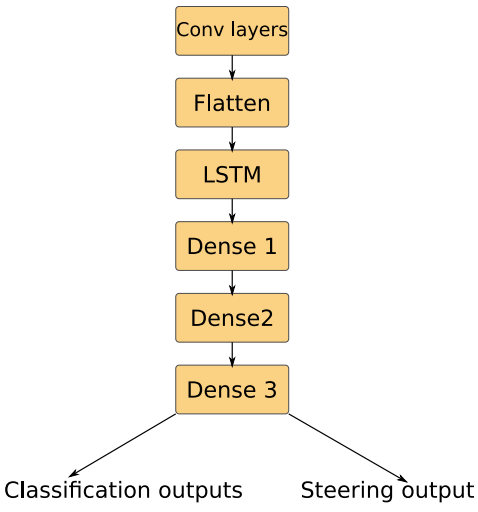

Figure 1.7: Basic model

For training, a model as shown in figure 1.7, is designed and its result is seen in figure 1.8. Interestingly, the training loss curve *follows* the classification loss as it dominates the model. Steering loss however, after epoch 32 starts to increase. Sure enough, upon evaluation, the steering was all over the place and acceleration was not stable at all, resulting in many collisions as shown in table 1.5 (dataset 3 column).

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 3, sufficient learning is not observed as steering is perhaps overwhelmed while training and its weight are not properly optimised. This paves way nicely to the next design iteration.

1.3.2 Splitting at the dense layers

To alleviate some of the burden the second dense layer(dense 2) is split into two separate dense layers; one for classification outputs and other for steering as show in figure 1.9. The result 1.10, stops the strange steering loss increase. Upon evaluation, this model shows better steering control but still the acceleration is not stable or consistent as shown in table 1.7.

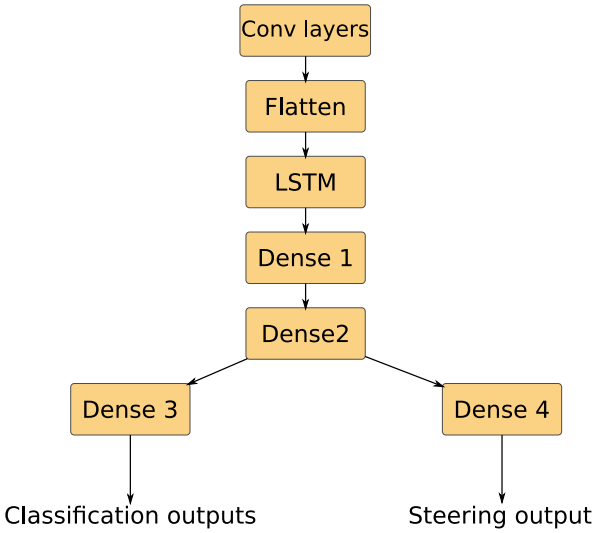

Figure 1.9: Split at the second dense layer

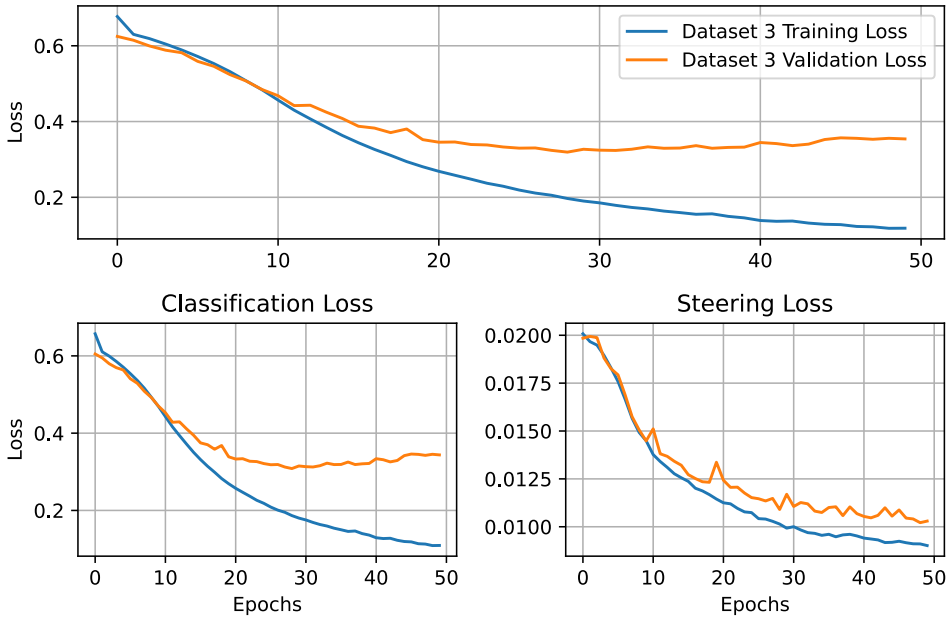

Figure 1.10: Separate dense layers for classification and steering

1.3.3 Splitting at the LSTM layers

Continuing the theme of tuning the network, the model is split further at LSTM layer as shown in figure 1.11. The main aim here is to see if steering control improves and is stable for considerable acceleration prediction. From figure 1.12, the steering loss gets a marginal gain. Still at evaluation, the trained models stops at random places, and steering control is 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 as seen from table 1.8.

1.3.4 Using two different NN for acceleration and Steering

It can be deduced that optimised weights play 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

Criteria(Softmax/Categorical crossentropy)	Dataset 3
Lane keeping/Drive straight	Yes
Gradual acceleration increase	Yes
Smooth braking behaviour observed	Yes
Smooth steering control at high speed(10m/s)	No
Smooth steering control at turnings	No
Avoids colliding into static objects	No
Detects vehicles as dynamic objects	Yes
Navigates traffic smoothly	No
Doesn't stop at random places	No

Table 1.7: Separate dense layers - How the model evaluates to different criteria

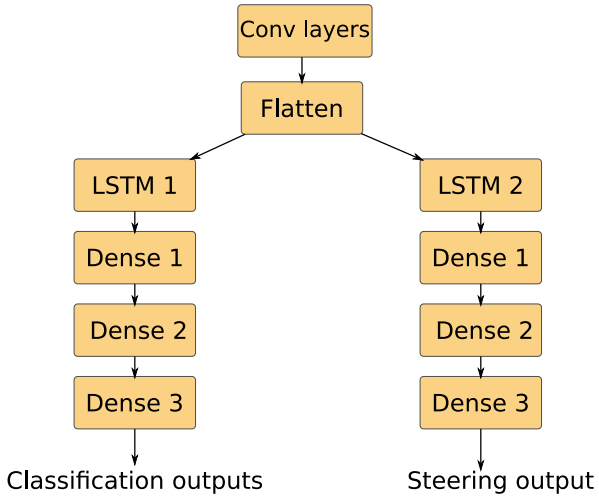

Figure 1.11: Split at the LSTM layer

Criteria(Softmax/Categorical crossentropy)	Dataset 3
Lane keeping/Drive straight	Yes
Gradual acceleration increase	Yes
Smooth braking behaviour observed	Yes
Smooth steering control at high speed(10m/s)	Yes
Smooth steering control at turnings	Yes
Avoids colliding with static objects	No
Detects vehicles as dynamic objects	Yes
Navigates traffic smoothly	Yes
Doesn't stop at random places	Yes
Smooth evaluation experience	No

Table 1.8: Split at the LSTM layer - How the model evaluates to different criteria

1.13. Though the result 1.14 looks similar to 1.12, the model predicts stable, consistent acceleration values.

From table 1.9, it even exhibits occasional turning behaviours at junctions. When it is exposed to

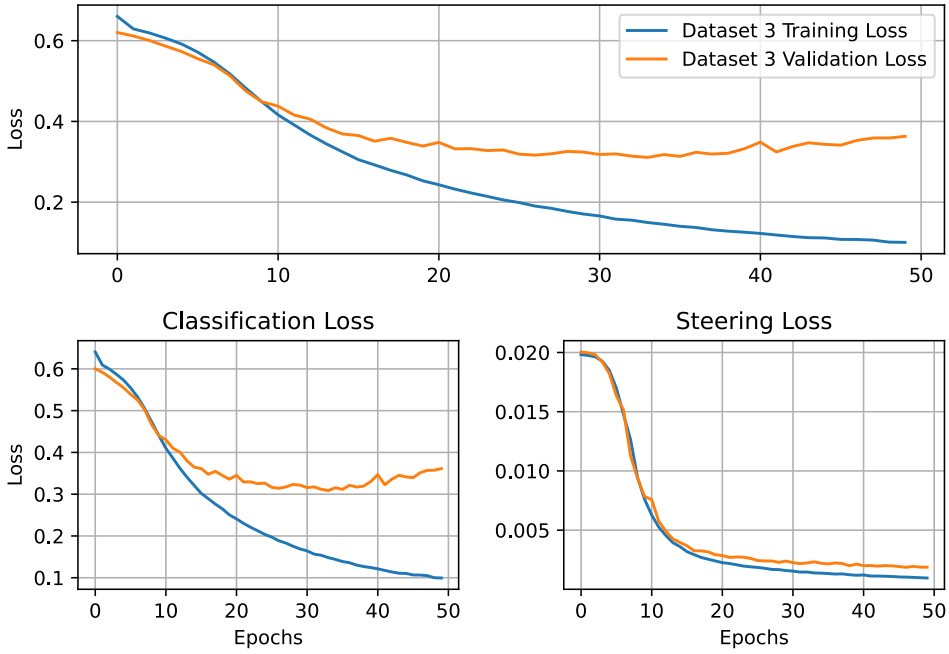

Figure 1.12: Separate LSTM layers for classification and steering

traffic, the model does well to navigate, brake, and accelerate.

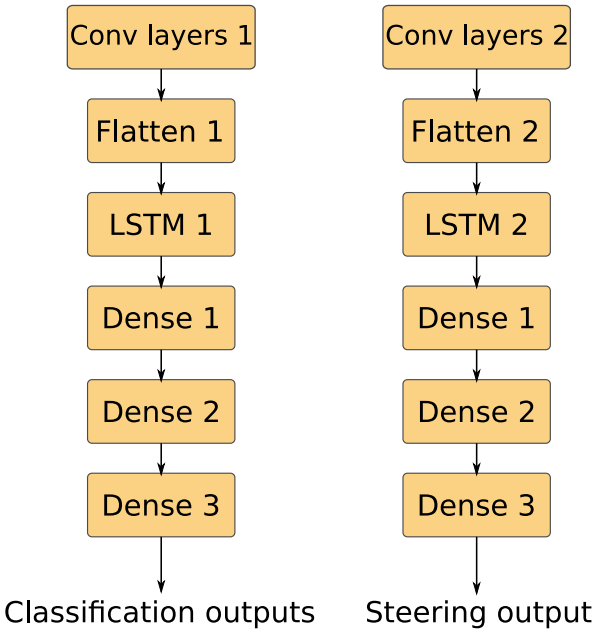

Figure 1.13: Separate NN training model

A quick overview of steering losses across different NN changes is shown in figure 1.15.

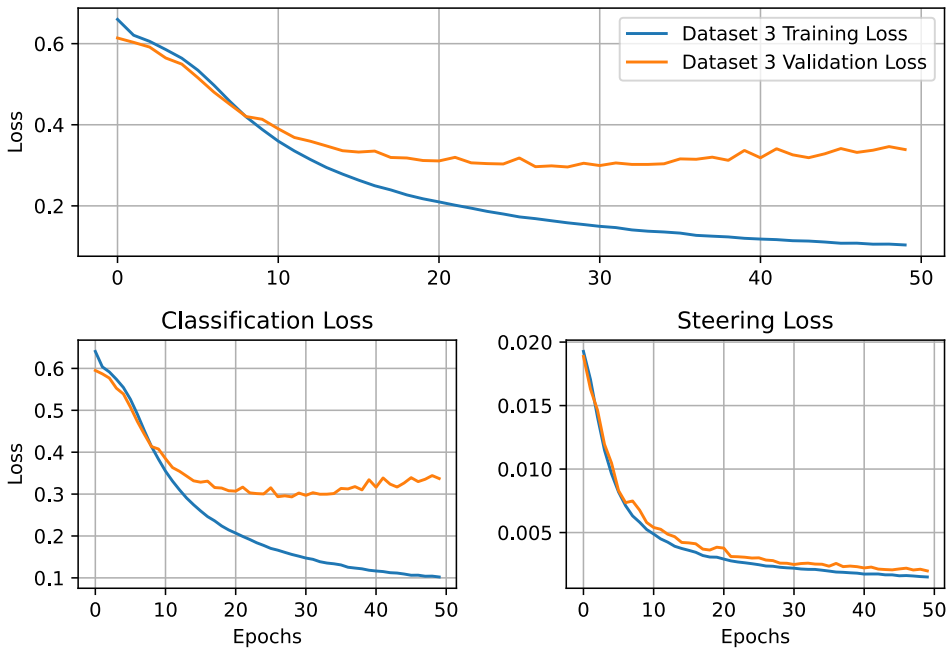

Figure 1.14: Separate neural network for Classification and Steering

Criteria(Softmax/Categorical crossentropy)	Dataset 3
Lane keeping/Drive straight	Yes
Gradual acceleration increase	Yes
Smooth braking behaviour observed	Yes
Smooth steering control at high speed(10m/s)	Yes
Smooth steering control at turnings	Yes
Avoids colliding with static objects	Yes
Detects vehicles as dynamic objects	Yes
Navigates traffic smoothly	Yes
Doesn't stop at random places	Yes
Smooth evaluation experience	Yes

Table 1.9: Separate neural network for classification and steering outputs - How the model evaluates to different criteria

1.3.5 Observations

1. Tuning the neural network albeit only the fully connected layers, results in optimised prediction of steering control corresponding to the classification outputs.

2. Probably because of imbalance in the dataset, the car while stops at random positions, and struggles to recover from it.

3. Separating into two NNs allows better steering control at a higher speed.

4. Sunlight and shadows still play a major role. They do some random, strange things to models that eventually lead to crashes out-of-nowhere. It could be deduced that some buildings' shadows could be considered as static or dynamic objects.

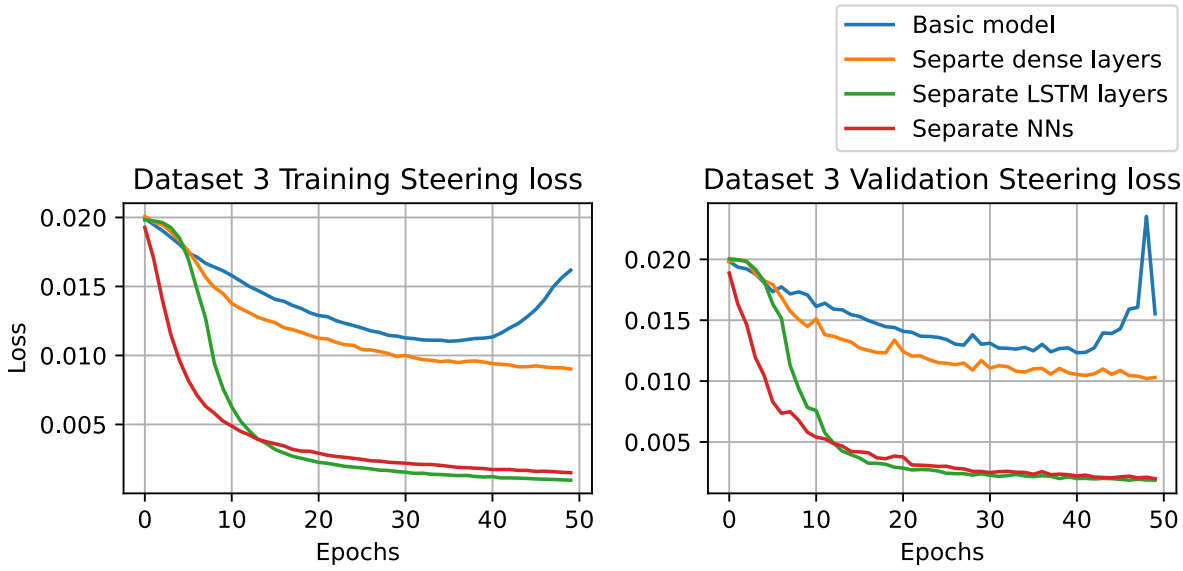

Figure 1.15: Steering command loss comparison

1.4 Velocity

Velocity is a scalar value, labelled output. Predicting velocity using convolutional layers that extract features would be a tedious task. However, we test the neural network with various configuration, like we accomplished in section 1.3, to force the convolutional layers to extract features, LSTM to carry the temporal information, and fully connected/dense layers to predict the velocity in addition to classification and steering outputs.

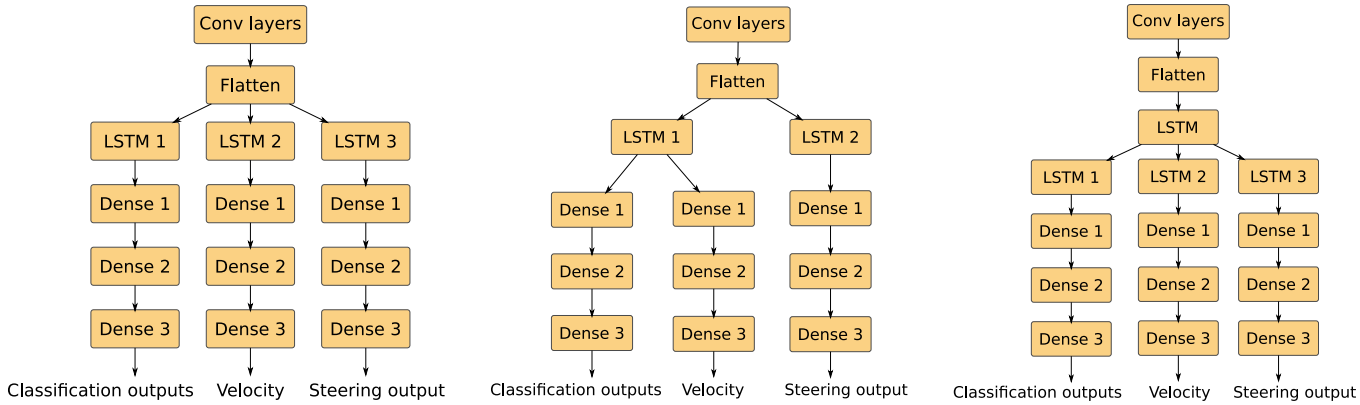

Figure 1.16: Different architectures used while predicting velocity

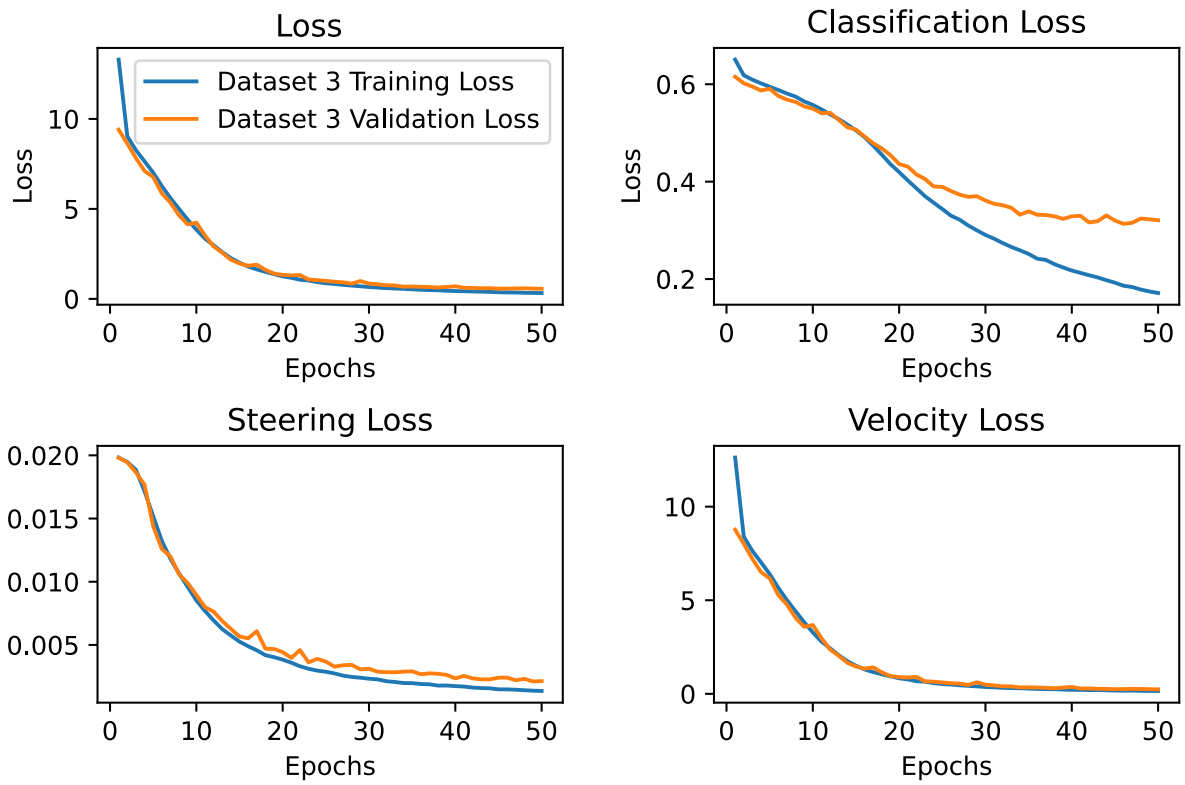

Figure 1.17: Losses for NN architecture a) in figure 1.16

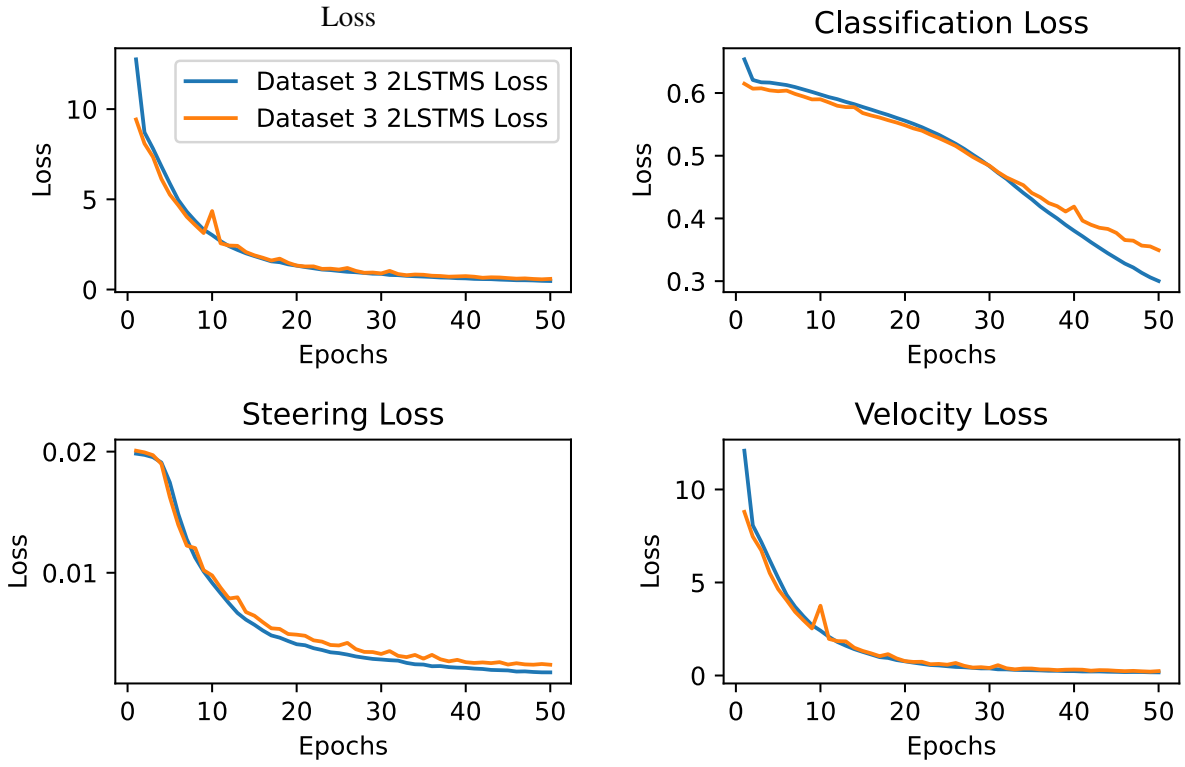

Figure 1.18: Losses for NN architecture b) in figure 1.16

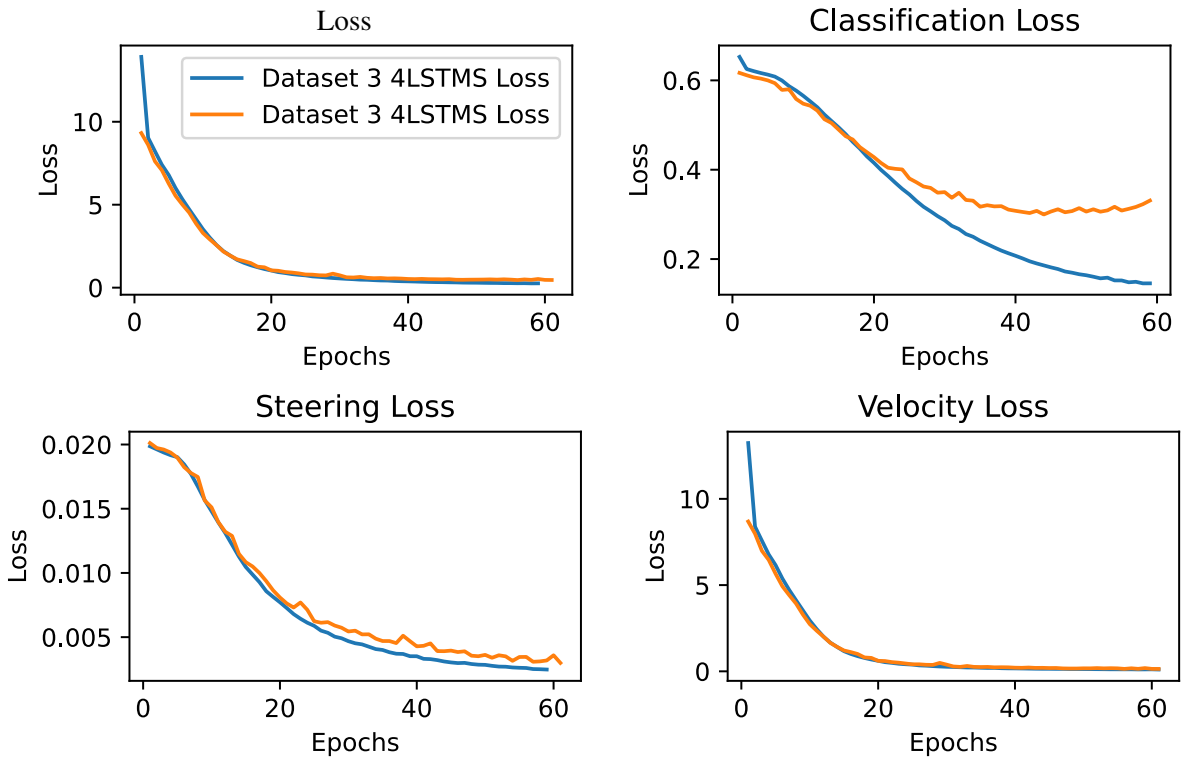

Figure 1.19: Losses for NN architecture c) in figure 1.16