# Intellisys - Project Report

19th July 2021

## Progresses

- The x and y coordinates for the input and output data are zero centered.
- One more boolean column called 'still_vehicle', which replaces the 'traffic_light' column for now. Reasons why this column is currently preferred over the 'traffic_light' column:
  1. We can use our already generated data. The value of this column is 'True', when the car was at the same position in the target timestamp, and else 'False'.
  2. It is probably a bit easier to learn for the model with 'still_vehicle' than with 'traffic_light', since a car can drive or stand still, when the traffic light is red.
- GPU use is now possible. Cuda was not available, because of wrong (for that purpose) pytorch version in environment.
- (For the follwing data generated by a simple crossing was used. As input the position, 'speed', 'yaw', 'intention' and 'still_vehicle' and as output the position at t+2 was used.)
- Overfit model for a single training sample.
  - Overfitting didn't work with every setting. It works best with:
    * Higher amount of epochs i.e. about 70 000.
    * Higher amount of hidden units i.e. 64 and 128.
    * Two hidden layers works better than one. For used model architecture see Figure 1 and 2.
    * Learning rate of 0.01 performed better than learning rate of 0.1 or 0.001.
  - Achieved loss after 70 000 epochs: 0.1617. Loss curve see Figure 3.
  - Sanity check: Comparison target and prediction looks acceptable (see Figure 4).
  - Questions:
    * Is this an acceptable loss?
    * Does it make sense that so many epochs are needed to overfit one single sample?
- Overfit model for two training samples.
  - All settings have remained unchanged compared to the one single training example. However, a slightly higher loss was obtained: 0.734.
  - Sanity check: Performed a bit worse than the one for the single training sample (see Figure 5).
  - Question: Would it make sense to include another column to indicate whether the cars are in the right or left lane or give the lanes a number, especially with regard to Figure 5?
- Hyperparameter tuning for 30 000 samples.
  - Different hyperparameter settings were used. For an overview see Figure 6.
  - For all tried settings the loss didn't go under 10 and stagnated there. All loss curves looks quite similar. For one representative example loss curve see Figure 7.
  - Questions:
    * So far less epochs than it was needed to overfit a single sample was used to train the model, since it takes a lot of time. (One epoch takes about two second. Meaning for e.g. 5 000 epochs training takes already 3 hours.) Could it be that one just have to train longer?
    * What else could be the reason for that result and what else can we try?

## Next Steps

- Focus is on achieving a satisfying result for the simple crossing.
- If the training for the simpler crossing works, proceed with the data of a more complicated crossing and train a model with that data.

## Referred figures

Figure 1: Used architecture with two hidden layers.

```python
def forward(self, x, edge_index, edge_attr):
    x = self.conv1(x, edge_index, edge_attr)
    x = x.relu()
    x = self.conv2(x, edge_index, edge_attr)
    x = x.relu()
    x = self.conv3(x, edge_index, edge_attr)
    return x
```

Figure 2: Used architecture with one hidden layer.

```python
def forward(self, x, edge_index, edge_attr):
    x = self.conv1(x, edge_index, edge_attr)
    x = x.relu()
    x = self.conv2(x, edge_index, edge_attr)
    return x
```

Figure 3: Train and validation loss curve for one single sample.

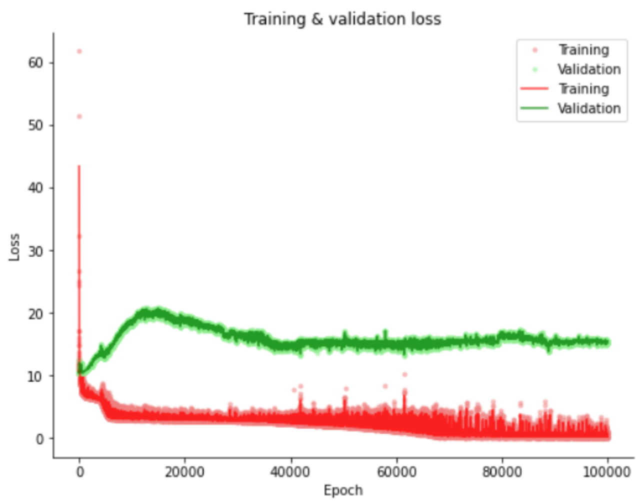


Figure 4: Sanity check for single training sample.

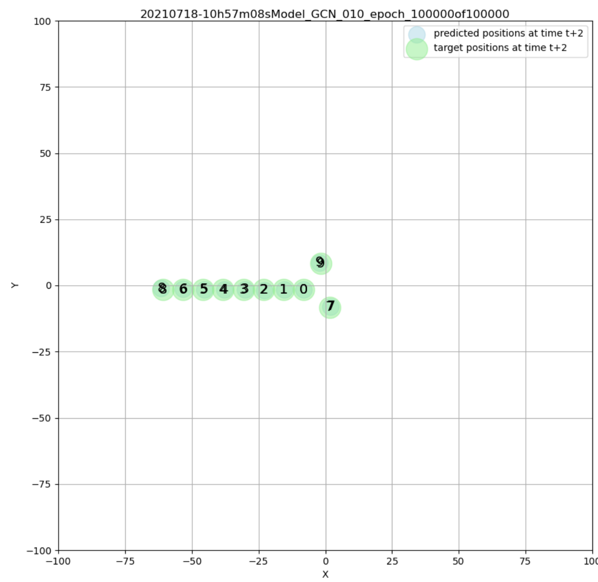


Figure 5: Sanity check for two training samples.

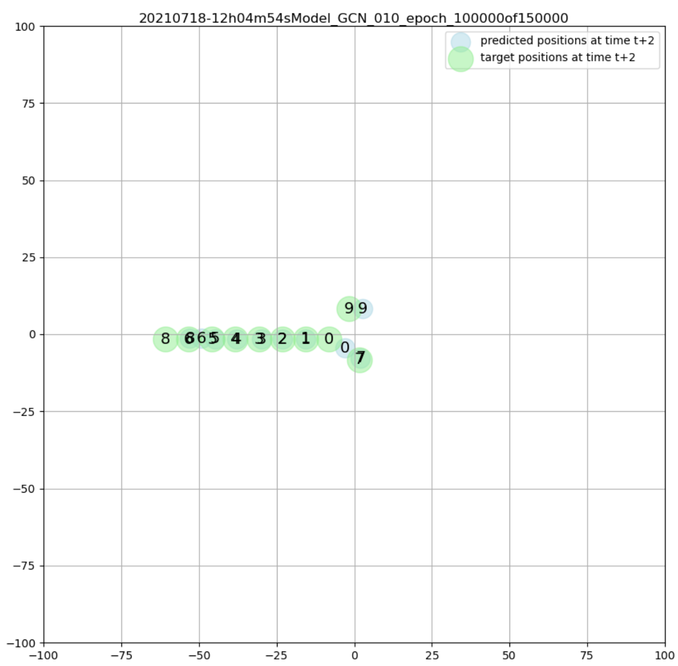


Figure 6: Overview hyperparameter settings.

| batch_size | optimizers | criterion | hl_sizes | lr_sizes | momentum_sizes | weight_decays | epochs | architecture |
|---|---|---|---|---|---|---|---|---|
| 8 | Adam | L1 | 16 | 0.01 | 0.9 | 0 | 500 | 1 hl |
| 16 | Adam | L1 | 16 | 0.001 | 0.9 | 0 | 500 | 1 hl |
| 32 | Adam | L1 | 16 | 0.001 | 0.9 | 0 | 500 | 1 hl |
| 32 | Adam | L1 | 16 | 0.0001 | 0.9 | 0 | 1000 | 1 hl |
| 32 | SGD | L1 | 64 | 0.001 | 0.9 | 0 | 400 | 1 hl |
| 128 | SGD | L1 | 128 | 0.01 | 0.9 | 0 | 400 | 1 hl |
| 64 | Adam | L1 | (128, 128) | 0.01 | 0.9 | 0 | 3000 | 2 hl |
| 64 | Adam | L1 | (64, 128) | 0.01 | 0.9 | 0 | 3000 | 2 hl |
| 64 | Adam | L1 | (64, 128) | 0.001 | 0.9 | 0 | 5000 | 2 hl |
| 64 | Adam | L1 | 128 | 0.001 | 0.9 | 0 | 5000 | 1 hl |

Figure 7: Representative loss curve.