Intellisys - Project Report

26th July 2021

Progresses

- We explored a number of combinations of parameters and models architectures (more than 370).
- We tried to change the input data:
  - Including/excluding the lane information. Each lane and also the routes that could be driven within the centre of the intersection has got its own ID. In the hope of putting more emphasis on the position, and make it easier for the model to learn the target. However, it turned out that this information made no significant difference, as the losses only differ slightly and not in favour of 'with lanes' or 'without lanes'.
  - Including/excluding the edge attributes (weights). Impression is that excluding the edge attributes improves the performances.
  - Using radians instead of degrees.

  We have extracted the TL information (which includes both the TL signal colour and the distance of the TL from the vehicle), but we haven't got to train with this yet.
- We trained longer with the activation functions ReLU and Tanh, where it seems that Tanh have a slightly better performance. Also we investigated the activation functions LeakyReLU, Sigmoid, SELU, CELU, SiLU, Softmax, ELU and RReLU with a smaller dataset, with LeakyReLu and SELU looking the most promising.
- We implemented lr schedulers see Fig 1:

```
lr_schedulers = [
                "ReduceLROnPlateau",
                "CosineAnnealingWarmRestarts",
                "None",
                "MultiStepLR",
                ]
```

- We experimented with normalization see Fig 2: GraphNorm seems to return the best values in terms of training loss, but still the validation loss seem to be the lowest without any normalization applied see Tab 1.
- We trained with different model architectures, up to 3 GC layers see Fig 3.

Issues/Questions

- The training and validations losses have improved, but the plots of the predictions are still not satisfying, see Fig 5 and 4 (also sometimes it seems that all the vehicles are dragged in the same direction)

Next Steps

- Get more data and train with that.
- Train longer with different activation functions apart from ReLU and Tanh.
- Try with different GCN layers.

Ref material

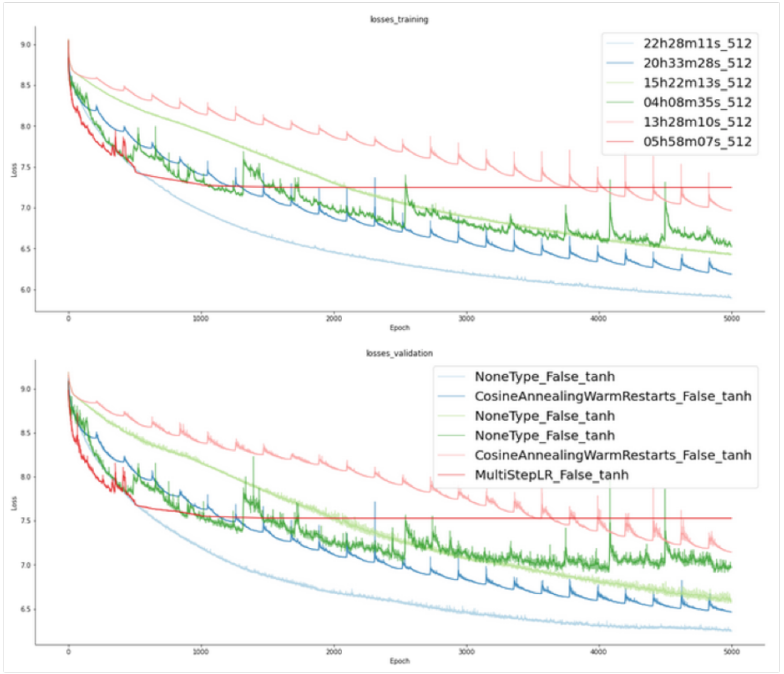
- Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- LEARNING GRAPH NORMALIZATION FOR GRAPH NEURAL NETWORKS
- Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting

Tables and Figures

Table 1: Losses generated with different norms but same setting

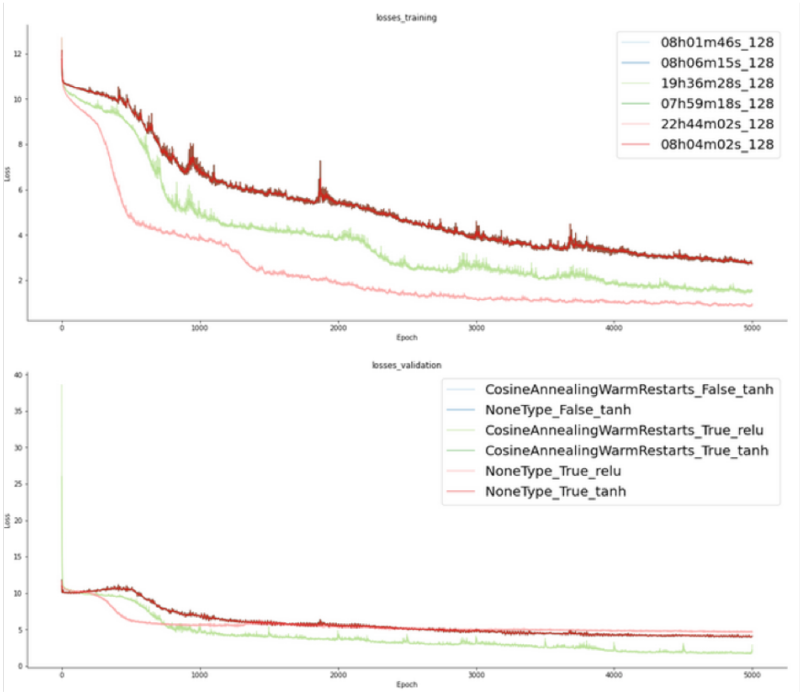
Norm	*Losses*
without norm	final_train_loss: 10.181829861232213 final_val_loss: 12.67313289642334
LayerNorm	final_train_loss: 10.618435314723424 final_val_loss: 14.753872871398926
BatchNorm	final_train_loss: 8.399091652461461 final_val_loss: 20.36924171447754
GraphNorm	final_train_loss: 8.452588285718646 final_val_loss: 14.96276569366455
InstanceNorm	final_train_loss: 9.636203629629952 final_val_loss: 16.914470672607422
PairNorm	final_train_loss: 10.122044290815081 final_val_loss: 15.48131275177002

Figure 1: effect of various lr schedules




```
--------------------------------------------------------------------------------

>20210721-22h28m11s
N epochs: 5000
num_rows_training: [10588]
final training: [5.89249493]
final validation: [6.24341822]

['mean', 512, 'Adam', 'L1', [128, 256, 64], 0.001, 0.6, 0, 'None', False, 'tanh']
--------------------------------------------------------------------------------

>20210721-20h33m28s
N epochs: 5000
num_rows_training: [10588]
final training: [6.18591495]
final validation: [6.46198972]

['mean', 512, 'Adam', 'L1', [128, 256, 64], 0.001, 0.6, 0, 'CosineAnnealingWarmRestarts', False, 'tanh']
--------------------------------------------------------------------------------

>20210721-15h22m13s
N epochs: 5000
num_rows_training: [10588]
final training: [6.43373208]
final validation: [6.61348883]

['mean', 512, 'SGD', 'L1', [128, 256, 64], 0.01, 0.6, 0, 'None', False, 'tanh']
--------------------------------------------------------------------------------

>20210721-04h08m35s
N epochs: 5000
num_rows_training: [10588]
final training: [6.51720304]
final validation: [6.97125117]

['mean', 512, 'Adam', 'L1', [128, 256, 64], 0.01, 0.6, 0, 'None', False, 'tanh']
--------------------------------------------------------------------------------

>20210721-13h28m10s
N epochs: 5000
num_rows_training: [10588]
final training: [6.96684815]
final validation: [7.14309216]

['mean', 512, 'SGD', 'L1', [128, 256, 64], 0.01, 0.6, 0, 'CosineAnnealingWarmRestarts', False, 'tanh']
--------------------------------------------------------------------------------

>20210721-05h58m07s
N epochs: 5000
num_rows_training: [10588]
final training: [7.24930836]
final validation: [7.52744738]

['mean', 512, 'Adam', 'L1', [128, 256, 64], 0.01, 0.6, 0, 'MultiStepLR', False, 'tanh']
```

Figure 2: effect of data normalization



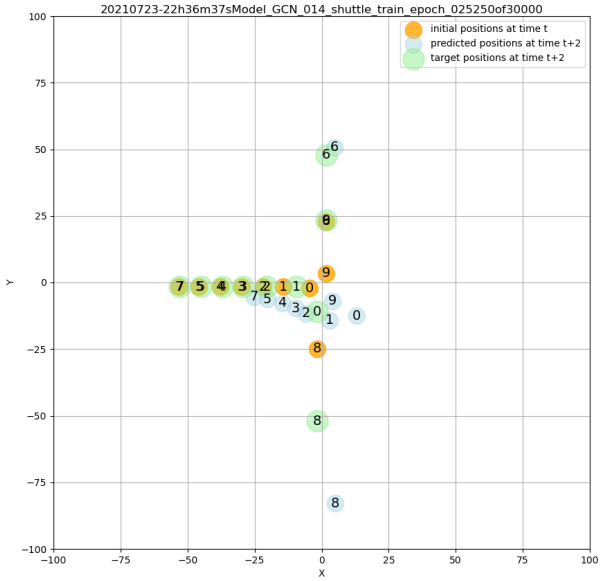

```
>20210724-08h01m46s
N epochs: 5000
num_rows_training: [127]
final training: [0.81653428]
final validation: [2.51633573]

['mean', 128, 'Adam', 'L1', [128, 256, 64], 0.001, 0.6, 0, 'CosineAnnealingWarmRestarts', False, 'tanh']
--------------------------------------------------------------------------------

>20210724-08h06m15s
N epochs: 5000
num_rows_training: [127]
final training: [0.29485065]
final validation: [2.72600365]

['mean', 128, 'Adam', 'L1', [128, 256, 64], 0.001, 0.6, 0, 'None', False, 'tanh']
--------------------------------------------------------------------------------

>20210723-19h36m28s
N epochs: 5000
num_rows_training: [127]
final training: [1.49044275]
final validation: [2.87384343]

['mean', 128, 'Adam', 'L1', [64, 128, 256], 0.001, 0.6, 0, 'CosineAnnealingWarmRestarts', True, 'relu']
--------------------------------------------------------------------------------

>20210724-07h59m18s
N epochs: 5000
num_rows_training: [127]
final training: [2.71957254]
final validation: [4.07233477]

['mean', 128, 'Adam', 'L1', [128, 256, 64], 0.001, 0.6, 0, 'CosineAnnealingWarmRestarts', True, 'tanh']
--------------------------------------------------------------------------------

>20210723-22h44m02s
N epochs: 5000
num_rows_training: [127]
final training: [0.9234668]
final validation: [4.65001392]

['mean', 128, 'Adam', 'L1', [64, 128, 256], 0.001, 0.6, 0, 'None', True, 'relu']
--------------------------------------------------------------------------------

>20210724-08h04m02s
N epochs: 5000
num_rows_training: [127]
final training: [1.49594164]
final validation: [5.18078184]

['mean', 128, 'Adam', 'L1', [128, 256, 64], 0.001, 0.6, 0, 'None', True, 'tanh']
```

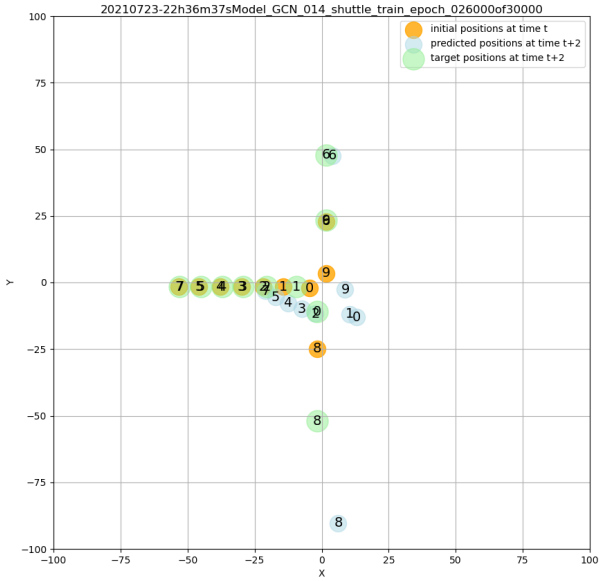
Figure 3: 3 layers GCN

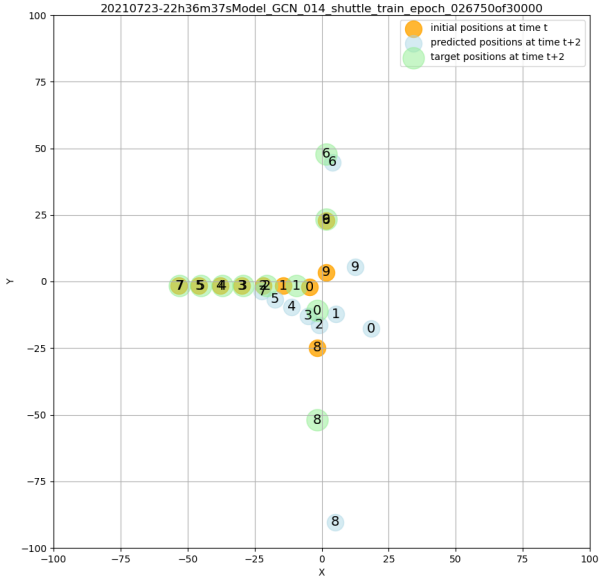
	370
run_time	08h01m46s
model	GCN_014
train_size	0.9
batch_size	128
Nepochs	5000
size_input	6
size_output	3
model_architecture	GCN_HL03_bn_tanh(\n (conv1): GraphConv(6, 128...
criterion	L1Loss()
optimizer	Adam (\nParameter Group 0\n amsgrad: False\...
reduction	mean
scheduler	CosineAnnealingWarmRestarts
final_train_loss	0.816534
final_val_loss	2.516336

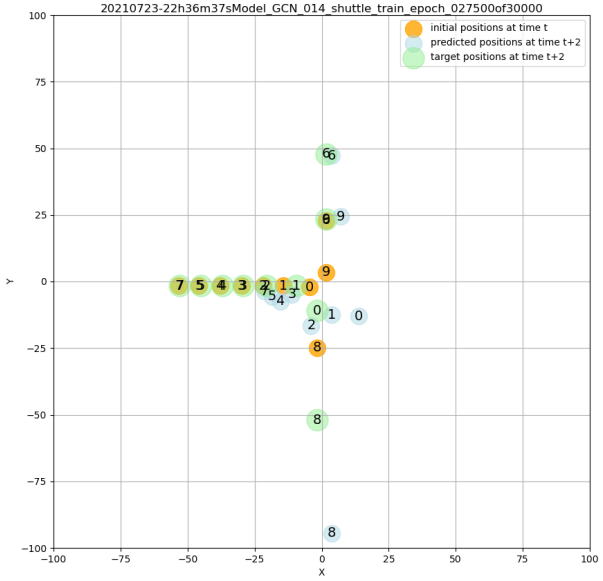
```
architecture=GCN_HL03_bn_tanh(
  (conv1): GraphConv(6, 128)
  (conv2): GraphConv(128, 256)
  (conv3): GraphConv(256, 64)
  (conv4): GraphConv(64, 3)
  (bn1): GraphNorm(128)
  (bn2): GraphNorm(256)
  (bn3): GraphNorm(64)
)
```

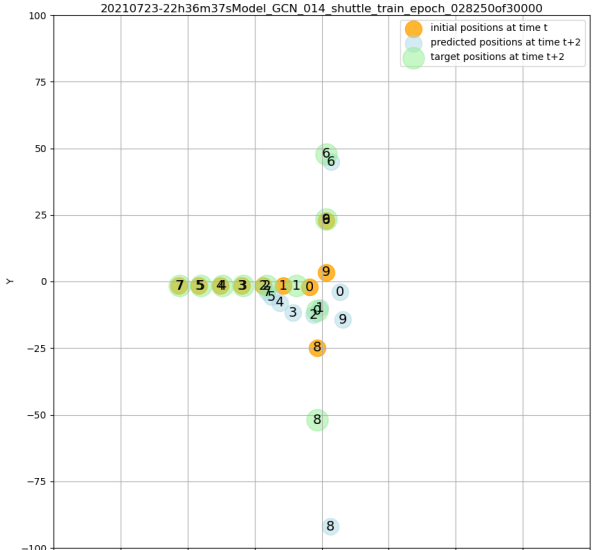
Figure 4: example of prediction, taking one training frame











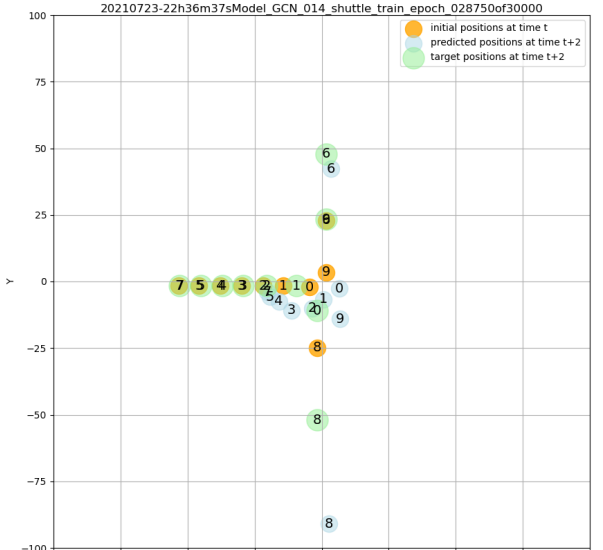


Figure 5: example of prediction, taking one validation frame