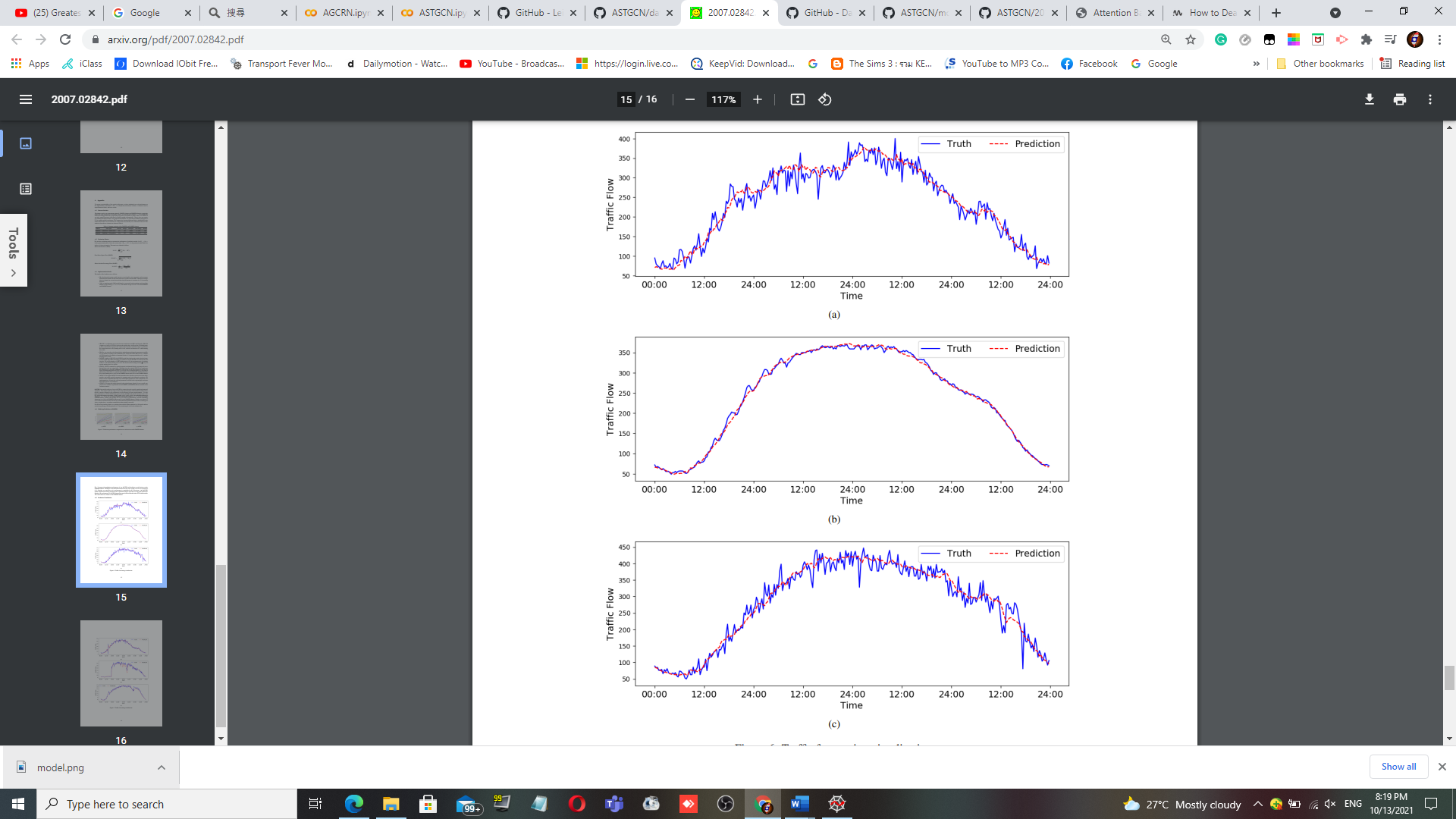
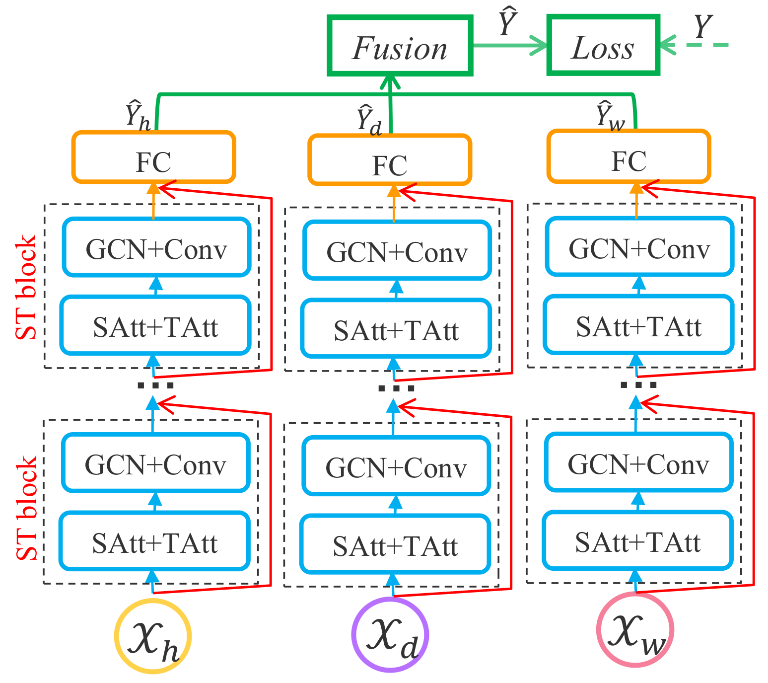
**Updates on October 29, 2021**

Chapter 2:

Bai et al, (2020) [71], designed a traffic forecasting model named “Adaptive Graph Convolutional Recurrent Network” (AGCRN), whose data is gathered from Los Angeles Traffic Data (PEMS04 & 08). The model, with GCN sub-architecture implemented, is capable of handling spatial and temporal correlations of the traffic flow, speed, and volume. AGCRN could capture fine-grained spatial and temporal correlations in traffic series automatically based on the two modules and recurrent networks. And the model outperformed several state-of-the-art models by a significant margin without pre-defined graphs about spatial connections.



Guo et al, (2019) [72], Designed a traffic forecasting GCN based architecture named “Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting” (ASTGCN). With non-GCN models incapable of modelling dynamic ST-correlations of the traffic data, the prediction is somewhat inaccurate without graph analysis. ASTGCN comprises of 3-Independent components; recent, daily, and weekly dependencies; and 2-submodels: 1. ST-Attention Mechanism to capture ST-correlations in the data 2. ST-Convolution simultaneously employs graph convolutions to capture spatial patterns with the common convolutions describing temporal features. Then, the weights generated by these dependencies are fused to generate prediction results – outperforming most existing state-of-the-art models. The dataset was “Caltrans Performance Measurement System (PeMS)”.



Ghaderi et al (2017) [73], designed a serial 6x Multi-LSTM framework named “Deepforecast – DL-Spatiotemporal Forecasting (DL-STF)”. In that paper, they modelled the spatiotemporal information by a graph whose nodes are data generating entities and its edges basically model how these nodes are interacting with each other. The model can forecast all nodes in the graph simultaneously based on one framework. The dataset is MS\_winds.dat, the collection of windmills in northeast USA. Compared to other benchmark models, the DL-STF was excellent at short term forecasts.

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

Update to Chapter 4:

**4.2. Preliminary Results**

The yellow cells indicate the models requiring significant architectural changes to calculate the MSE due to unequal input-output tensor dimensions. The blue characters indicate possible Metric values as if whose models were run (can be turned black later if a suitable environment is found). And the pink cells indicate models with memory problem – insufficient RAM – requiring a cloud environment to run. Despite stringent environmental restrictions preventing such models from running on COLAB, it is obvious that the model performed well, at least for its architecture. To see the detailed issues and possible solutions, see the section 4.3.

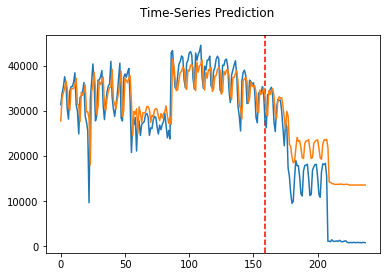
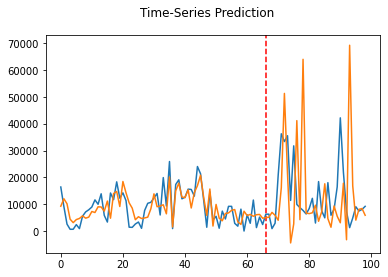
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Architecture** | **Architecture Desc** | **Dataset Name/Desc** | **Loss Function** | **learning rate** | **Epochs** | **Major DL Module** |
| Taxi-Simple-LSTM-pytorch | Simple-LSTM | Time series of Taxi-Uber DS. (2014-15) | MSE | 0.01 | 200 | Pytorch |
| Uber-Simple-LSTM-pytorch | Simple-LSTM | Time series of Taxi-Uber DS. (2014-15) | MSE | 0.01 | 200 | Pytorch |
| Taxi-Simple-LSTM-Keras | Simple-LSTM | Time series of Taxi-Uber DS. (2014-16) | MSE | 0.01 | 200 | Tensorflow Keras |
| Uber-Simple-LSTM-Keras | Simple-LSTM | Time series of Taxi-Uber DS. (2014-17) | MSE | 0.01 | 200 | Tensorflow Keras |
| CRANN-Temporal | Bahdanau Att.Mech Autoencoder (LSTM based) | temporal time series of hourly/daily car traffic (in Madrid) | MSE | 0.01 | 200 | Pytorch |
| CRANN-Spatial | CNN+ST-Att.Mech | incidence captured by 30 sensors + Timestamps (A 17000x30 matrix) | MSE | 0.01 | 200 | Pytorch |
| CRANN-Dense | Fully Connected Feedforward NN (FCFFNN) | dense 3D+ tensor of the both preceeding modules | MSE | 0.01 | 200 | Pytorch |
| Seq2seq (flow) | Improved Seq2eq | Traffic flow & speed dataset | MSE | 0.01 | 200 | MXNET |
| GAT Seq2seq (flow) | Improved Seq2eq | Traffic flow & speed dataset | MSE | 0.01 | 200 | MXNET |
| ST-Metanet (flow) | Improved Seq2eq | Traffic flow & speed dataset | MSE | 0.01 | 200 | MXNET |
| Seq2seq (speed) | Improved Seq2eq | Traffic flow & speed dataset | MSE | 0.01 | 200 | MXNET |
| GAT Seq2seq (speed) | Improved Seq2eq | Traffic flow & speed dataset | MSE | 0.01 | 200 | MXNET |
| ST-Metanet (speed) | Improved Seq2eq | Traffic flow & speed dataset | MSE | 0.01 | 200 | MXNET |
| AGCRN - PeMSD4 | Attentive Graph CRN | Caltrans PEMS04&08 | MAE | 0.003 | 100 | Pytorch |
| AGCRN - PeMSD8 | Attentive Graph CRN | Caltrans PEMS04&08 | MAE | 0.003 | 100 | Pytorch |
| ASTGCN - PeMSD4 | Attention Based GCN | Caltrans PEMS04&08 | MAE | 0.001 | 100 | MXNET |
| ASTGCN - PeMSD8 | Attention Based GCN | Caltrans PEMS04&08 | MAE | 0.001 | 100 | MXNET |
| Deepforecast | Multi-LSTM | MS\_winds - Wind Speed & Flow Dataset | MAE | 0.001 | 80 | Tensorflow Keras |
|  |  |  |  |  |  |  |
| DCRNN - 15min | R-CNN | PEMS & METR-LA | MAE | 0.01 | N/A | Tensorflow Keras |
| DCRNN - 30min | R-CNN | PEMS & METR-LA | MAE | 0.01 | N/A | Tensorflow Keras |
| DCRNN - 1hr | R-CNN | PEMS & METR-LA | MAE | 0.01 | N/A | Tensorflow Keras |
| STGCN - 15min | Graph-CNN | PEMS & METR-LA |  | 0.001 | 50 | Pytorch |
| STGCN - 30 min | Graph-CNN | PEMS & METR-LA |  | 0.001 | 50 | Pytorch |
| STGCN - 1hr | Graph-CNN | PEMS & METR-LA |  | 0.001 | 50 | Pytorch |

Running Results

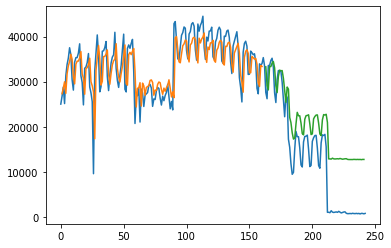
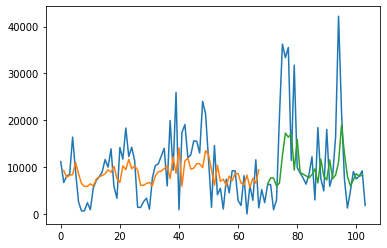
The AGCRN, and the DCRNN models have a serious dataset error, even if running on COLAB. Import fails are common and remain unsolved. Running it on a physical PC also returns a value error issue even before the training commences. Meanwhile, the code of the ASTGCN model is OK, but the model consumed too much RAM – more than what COLAB can support (Like the ST-METANET model, 64GB+ is a safe level)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Key Metrics** |  | **Other Metrics** |  |
| **Architecture** | **MSE** | **RMSE** | **MAE** | **Type** | **Value** |
| Taxi-Simple-LSTM-pytorch | 75534780 | 8691.075 | N/A |  |  |
| Uber-Simple-LSTM-pytorch | 0.032575 | 0.1804854 | N/A |  |  |
| Taxi-Simple-LSTM-Keras | 17995921.47 | 4242.16 | N/A |  |  |
| Uber-Simple-LSTM-Keras | 33913501.66 | 5823.53 | N/A |  |  |
| CRANN-Temporal | 6954.6177 | 83.39435 | N/A |  |  |
| CRANN-Spatial | 58546.699 | 241.96425 | N/A |  |  |
| CRANN-Dense | 67154.617 | 259.14208 | N/A |  |  |
| Seq2seq (flow) | 1626.986623 | 40.33592224 | 21.3 |  |  |
| GAT Seq2seq (flow) | 1098.041371 | 33.13670731 | 18.3 |  |  |
| ST-Metanet (flow) | 813.1885148 | 28.51646042 | 16.9 |  |  |
| Seq2seq (speed) | 44.4751941 | 6.668972492 | 3.55 |  |  |
| GAT Seq2seq (speed) | 36.92343427 | 6.076465607 | 3.28 |  |  |
| ST-Metanet (speed) | 33.63158961 | 5.799274921 | 3.05 |  |  |
| AGCRN - PeMSD4 | 1040.7076 | 32.26 | 19.83 |  |  |
| AGCRN - PeMSD8 | 636.0484 | 25.22 | 15.95 |  |  |
| ASTGCN - PeMSD4 | 1240.4484 | 35.22 | 22.93 |  |  |
| ASTGCN - PeMSD8 | 787.3636 | 28.06 | 18.25 |  |  |
| Deepforecast | 2.481881285 | 1.5753988 | 1.1590073 | NRMSE\_maxmin(%) | 15.575216 |
|  |  |  |  | NRMSE\_mean(%) | 43.097104 |
| DCRNN - 15min | 28.9444 | 5.38 | 2.77 |  |  |
| DCRNN - 30min | 41.6025 | 6.45 | 3.15 |  |  |
| DCRNN - 1hr | 57.6081 | 7.59 | 3.6 |  |  |
| STGCN - 15min | 59.07250543 | 7.68586 | 3.70366 | %Wmape | 7.29083 |
| STGCN - 30 min | 78.55590653 | 8.86318 | 4.51874 | %Wmape | 8.89655 |
| STGCN - 1hr | 132.2214585 | 11.4988 | 5.99748 | %Wmape | 11.8111 |

To explain the results on Table 4, the simple LSTM models had a tremendous RMSE error in thousands due to simplicity of the model. Also due to overfitting, the UBER-LSTM Pytorch model is somewhat accurate. Nevertheless, we should never trust the number unless we see the graphs of figure 29 and 30. The TPA and the Lotto Seq2seq models are incomparable with the others due to metric difference. This time, the TPA-LSTM and the Lotto Seq2seq models were removed due to incompatible metrics with other models



*Figure 29: Pytorch-LSTM prediction results for (left) UBER and (right) TAXI ridership data*



*Figure 30: Keras-LSTM prediction results for (left) UBER and (right) TAXI ridership data*

It is also apparent that complex state-of-the-art models (such as CRANN), when learning correct datasets, greatly outperformed simple models (like Keras-LSTM). As we can see on the table above, the “Deepforecast” Multi-LSTM model, to date, is the best model for spatiotemporal traffic prediction due to the depth of a classic but simple to calibrate LSTM – nothing could match it when trained by MS-Winds dataset. Nevertheless, it takes 2 hours to train in COLAB.

A rivaling model to ST-METANET is DCRNN trained by PEMS & METR-LA datasets, which has a comparable performance with ST-METANET of the speed prediction – 2.77 vs 3.05 MAE, and 5.38 vs 5.80 RMSE.

Updates to references:

[71] Bai, L., Yao, L., Li, C., Wang, X., & Wang, C. (2020). Adaptive graph convolutional recurrent network for traffic forecasting. *arXiv preprint arXiv:2007.02842*.

[72] Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019, July). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 922-929).

[73] Ghaderi, A., Sanandaji, B. M., & Ghaderi, F. (2017). Deep forecast: Deep learning-based spatio-temporal forecasting. *arXiv preprint arXiv:1707.08110*.