COVID-19 Forecasting using Spatio-Temporal Graph Attention Networks E6691.2024Spring.GCNN.report.nhw2114

Han Yong Wunrow (nhw2114)*
*Department of Applied Physics and Applied Mathematics, Columbia University, New York, New York

***Abstract*—In this project, we aimed to implement the Graph Convolutional Network (GCN) architecture detailed in "Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks" by Kapoor et al. (2020), training the model solely on case, mortality, and mobility data from the 5 boroughs in New York City. Employing the `PyTorch Geometric` and `PyTorch Geometric Temporal` packages, we successfully reproduced the proposed architecture and compared it against a modified attention temporal graph convolutional network (A3T-GCN). The A3T-GCN model achieved the lowest RMSLE of 0.0550 followed by our implementation of the original paper's architecture with a RMSLE of 0.0605. The model performances differed from the results in Kapoor et al. since we trained our model on data only in New York City whereas Kapoor et al. trained their models on county-level data in the entire country.**

1. Introduction

During the SARS-CoV-2 pandemic, New York City (NYC) was the epicenter of the United States during the spring of 2020 and has reported over 3.5 million cases and 46,000 deaths as of April 20th, 2024 [3]. Although mathematical models have been increasingly used to support public health decision-making, many predictive models for infectious disease transmission have not explicitly and adequately accounted for the dynamic and heterogeneous nature of population connectivity, leading to potentially inaccurate or biased predictions of disease spread and intervention effectiveness.

Based on our literature review, there are two primary approaches in epidemiological modelling to address the well-mixed population assumption. The first class of models are referred to as metapopulation models [4], [5] where a population is segmented into discrete subpopulations, each characterized by its unique dynamics and connectivity patterns. Although they are an improvement over traditional epidemiological models, metapopulation models still rely on several assumptions about population structure, movement patterns, and disease dynamics. Sensitivity to these assumptions means that small changes in model parameters or assumptions can lead to significant differences in model outcomes, potentially affecting the reliability of predictions.

The other class of models are statistical models that take a time series learning approach – for example, applying curve-fitting [6], Autoregression (AR) [7], or deep learning [8] applied to epidemiological time series data. Compared to existing dynamical time-series models, graph neural networks (GNNs) are better suited for epidemiological modeling since they may be able to better capture the complex spatial and temporal dynamics of disease transmission with minimal assumptions on the actual disease dynamics. Several groups have examined the SARS-CoV-2 pandemics using GNNs [9], [10]. In this paper, we review "Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks" by Kapoor et al. (2020) [10], modify the authors' architecture, provide details of our implementation, and present our results.

2. Summary of the Original Paper

2.1. Methodology of the Original Paper

The authors employ GNNs to predict the change in the number of cases on day $t + 1$, given information on days $1, \ldots, t$, for each U.S. county during the first epidemic wave (02/22/2020 - 05/31/2020). For example, if there were 100 cases on day 10 in King County and 150 cases on day 11, the true value to be predicted is 50. The authors use three datasets: the New York Times (NYT) COVID-19 dataset [11], the Google COVID-19 Aggregated Mobility Research Dataset, and the Google Community Mobility Reports [12]. While both the New York Times (NYT) COVID-19 dataset and the Google Community Mobility Reports are publicly available, the Google COVID-19 Aggregated Mobility Research Dataset is not. We utilize a separate mobility dataset that we describe in section 4.1.

To incorporate both spatial and temporal trends into the GNN model, the authors design a static spatial-temporal network shown in figure 1. In each slice, the nodes represent a county on a particular day with edges representing the mobility between each county. In between slices, they also include temporal edges that connect each county to its respective county node in the previous 7 days. The node features include the node's state, county, day, case and death counts on the day and the previous 7 days from the New York Times (NYT) COVID-19 dataset, and the mobility trends from Google Community Mobility Reports. The edge weights are the inter-county and intra-county flows of users between and within counties (i.e., self-loops are included).

The edge weights for temporal edges were not described in the original text.

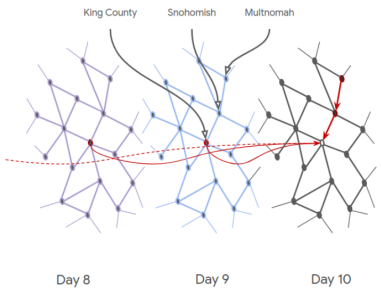

Figure 1. Snapshot of the author's static spatio-temporal network (directly from [10]). Each slice represents the spatial network between U.S. counties on a particular day.

The GCN in the orignal paper has 4 layers: 1 embedding layer, 2 graph convolutional layers, and 1 final output layer. The model prediction P can be represented as:

$$H_0 = \text{mlp}\left(\mathbf{x_t} \,|\mathbf{x_{t-1}}| \ldots | \,\mathbf{x_{t-d}}\right) \tag{1}$$

$$H_1 = \sigma\left(\hat{A}H_0W_1\right) \,|\, H_0 \tag{2}$$

$$H_2 = \sigma\left(\hat{A}H_1W_2\right) \,|\, H_0 \tag{3}$$

$$P = \text{mlp}\left(H_s\right) \tag{4}$$

where x_t is the node feature vector on day t, $d = 7$, H_l represents the hidden state at layer l, $\hat{A}$ is the spectral normalized adjacency matrix, W_l is the learned weight matrix at layer l, $|$ is the concat operator, and σ is the ReLU activation function. The hidden layer architecture for W_0, W_1, W_2, and W_3 are [64, 32, 32, 32], respectively. As shown in figure 2, the first embedding hidden layer is concatenated to following hidden layer in order to avoid weakening the self-node feature state and address vanishing gradient issues.

For the graph convolutional layers, the authors use the standard graph convolutional operator from Kipf and Welling [13]. The node-level formulation is given by:

$$\mathbf{x}'_i = \mathbf{\Theta}^\top \sum_{j \in \mathcal{N}(i)\cup\{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{x}_j \tag{5}$$

where $\mathbf{x}_j$ is the feature vector for node j, $e_{j,i}$ edge weight from node j to node i, $\mathcal{N}(i)$ is the set of nodes adjacent to node i, $\hat{d}_i = 1 + \sum_{j \in \mathcal{N}(i)} e_{j,i}$, and $\mathbf{\Theta}$ are the learned weights. Neighbors were defined as the nodes with the top 32 highest edge weights.

For training, they utilize an ADAM optimizer with learning rate set to 1e-5. Each layer has a dropout rate of 0.5, and a $l2$ regularization term of 5e-4. GNN models were trained for 1M steps with a mean squared logarithmic error (MSLE) regression loss given by

$$\text{MSLE} = \frac{1}{n}\sum_{i=1}^{n}\left(\log\left(\hat{y}_i + 1\right) - \log\left(y_i + 1\right)\right)^2. \tag{6}$$

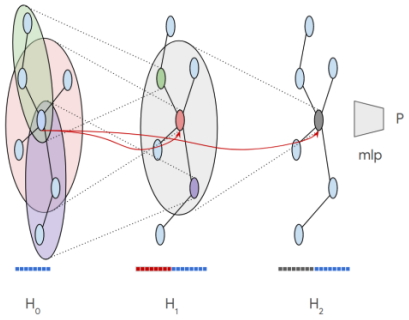

Figure 2. Original paper's architecture (directly from [10]). The elipses represent the graph convolutions. The hidden layer concatenation at the bottom represents the skip connections.

Models were trained from days 59-120 and days 120-150 from the year 2020 were used for testing.

2.2. Key Results of the Original Paper

The authors also compared against several other baseline models (ARIMA, LSTM, and Seq2Seq), but since we did not re-implement them for this project we do not show their results. Both the Previous Case and Previous Delta models were deterministic baseline models. The Previous Case model assumes that the change in cases was 0 and predicts that $\hat{y}_{t+1} = y_t$. The Previous Delta model assumes that the change in cases on day t will be the same as the change in cases on day $t-1$ and predicts that $\hat{y}_{t+1} = y_t + (y_t - y_{t-1})$. These two baseline models were chosen since the next day case prediction is highly correlated with the current day case values.

Table 1 shows the model performances against these two baseline models. Performance metrics were computed for the top 20 counties by population and averaged over the 30 day testing period. RMSLE is the root mean squared logarithmic error (i.e., square root of equation 6), and Corr is the Pearson Correlation between the predicted cases and actual cases. Δ RMSLE and Δ Corr are computed between the predicted change in cases and actual change in cases. Their Graph Convolutional Network (GCN) model performs either the best or second best across all metrics.

Model	RMSLE	Corr	Δ RMSLE	Δ Corr
Previous Cases	0.0226	**0.9981**	4.7879	NaN
Previous Delta	0.0127	0.9965	0.9697	0.1854
GCN	**0.0109**	<u>0.9980</u>	**0.7983**	<u>0.2230</u>

TABLE 1. ORIGINAL PAPER MODEL PERFORMANCES. **BOLDED** VALUES INDICATE THE BEST METRIC. <u>UNDERLINED</u> VALUES INDICATE THE SECOND BEST.

3. Methodology (of the Students' Project)

For this project, we re-implement the GCN architecture and replicate the results from Kapoor et al. [10]. Due to computational limitations, we train our model only using

data from the counties (boroughs) from New York City. Also, since the Google COVID-19 Aggregated Mobility Research Dataset is not publicly available, we utilize the SafeGraph [14] foot traffic data available from our research group. Due to the data use agreements we do not provide the raw data for this project.

We also modified the architecture by using an attention temporal graph convolutional network (A3T-GCN) model from Bai et al. [15]. We compare the results from these two models along with the baseline Previous Cases and Previous Delta models.

3.1. Objectives and Technical Challenges

We have three main objectives: (1) implement the spatio-temporal GCN architecture outlined in Kapoor et al. [10], (2) modify the architecture using temporal graph convolutional network from Bai et al., and (3) compare their performances against baseline Previous Cases and Previous Delta models using only data from New York City.

There were many details left out of the original paper that made the first task of implementing the authors' model challenging. The authors did not provide any of their source code so all of the architecture and training details had to come from the main text. Another technical challenge was that they did not describe how they differentiated the temporal and spatial edges in their convolution operator since the temporal edges do not have well-defined edge weights. We defaulted to edge weights of 1 for temporal edges.

Another challenge was processing the SafeGraph mobility data. Just in New York City there were over 2 million unique locations and 100,000 devices that had to be aggregated to the borough level. The original JSON files amounted to 75 GB for this time period. The majority of time for this project was spent on processing this data.

3.2. Problem Formulation and Design Description

The objective of both models is to find the optimal weights Θ that minimize the RMSLE as shown in equation 7 where $\hat{y}_i$ are the model predictions for node i and y_i are the true values for node i.

$$\underset{\Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(\log\left(\hat{y}_i + 1\right) - \log\left(y_i + 1\right)\right)^2. \qquad (7)$$

4. Implementation

To implement the GCN and A3T-GCN models we utilize the `PyTorch Geometric` and `PyTorch Geometric Temporal` libraries. Both libraries are built upon `PyTorch`, offer GPU support, and are the standard libraries used for geometric deep learning (i.e., deep learning on graphs). As shown in figure 7, the 3 datasets used are converted into `PyTorch Geometric` and `PyTorch Geometric Temporal` data objects. Since the network was small enough, using data only from NYC, all learning could be performed in a single batch.

Regarding hardware, we used Google Cloud Platform's Compute Engine with 1 NVIDIA T4 GPU and 1 n1-standard-2 (2 vCPU 7.5 GB RAM). All of the data processing was performed locally using a Apple M2 Pro. The framework that we used is `PyTorch` 2.2.2.

4.1. Data

We utilized three datasets for this project.

1) NYC COVID-19 dataset [16]: Daily data on coronavirus cases and deaths in NYC aggregate to the borough level during the first wave 02/29/2020 - 05/30/2020.
2) Safegraph Mobility Data [14]: Anonymized mobility location data from mobile devices. The dataset in NYC included over 2 million locations and 100,000 devices.
3) Google Community Mobility Reports [12]: Reports charted movement trends over time by geography, across different categories of places such as retail and recreation, groceries and pharmacies, parks, transit stations, workplaces, and residential.

We visualize these data in figures 3, 4, and 5. As shown in figure 4, the mobility data is highly biased towards Manhattan (i.e., the flows into and within Manhattan are much higher). The mobility trends in figures 4 and 5 track well with the epidemic curves in figure 3 where as cases and deaths increase, mobility tends downwards with the exception of residential (people are staying home) and parks (people are going outside).

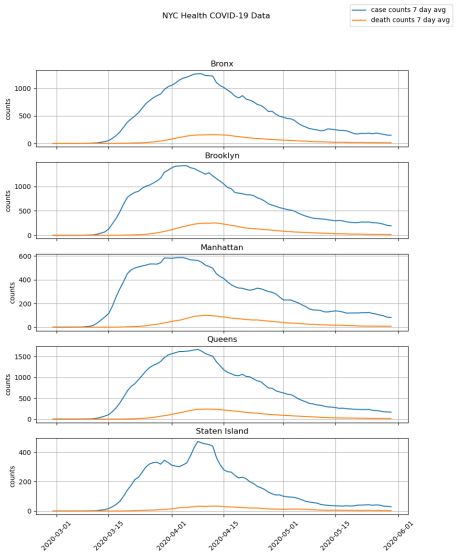

Figure 3. NYC COVID-19 dataset.

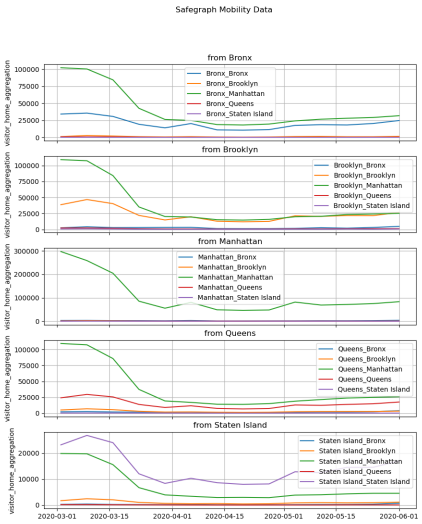

Figure 4. Safegraph Mobility Data.

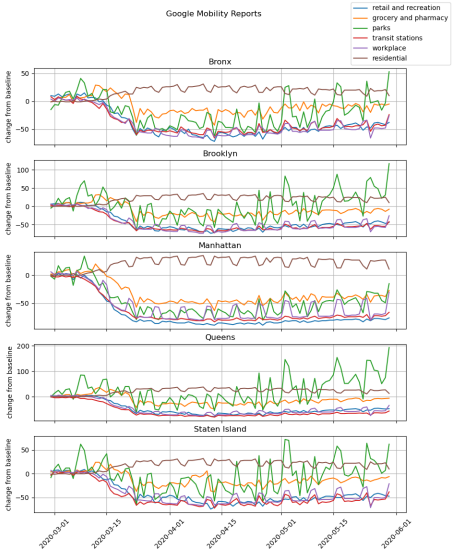

Figure 5. Google Community Mobility Reports.

4.2. Deep Learning Network

The GCN model architecture is outlined in section 2.1 and figure 2.

The A3T-GCN model architecture is shown in figure 6. Instead of a static spatio-temporal network, the A3T-GCN model uses a dynamic spatio-temporal network where each set of data on day t has a snapshot network with node features X_t. Each snapshot is then fed into a T-GCN block which has two components: (1) GCN for spatial trends and (2) a gated recurrent unit (GRU) for temporal trends. These sets of T-GCN blocks were perfomed on windows of snapshots of length n. For each set of snapshots, attention scores were calculated simply using the softmax of the outputs of a MLP and a weighted sum of the annotation vectors.

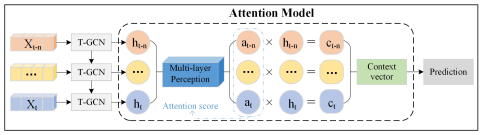

Figure 6. A3T-GCN architecture (directly from [15]).

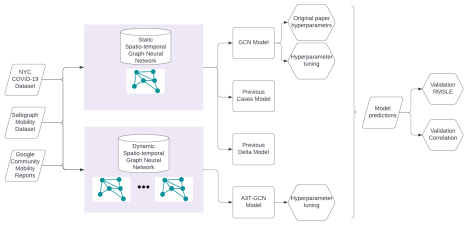

Figure 7. Data processing and modelling pipelines.

For training the GCN model, we used two sets of hyperparameters: (1) the original paper's hyperparameters outlined in section 2.1 and (2) hyperparameters tuned for this project. For (2), we utilized an ADAM optimizer with learning rate set to 1e-2. Each layer had a dropout of 0.1 and a $l2$ regularization term of 5e-4. The model was trained for 100,000 steps. The set of hyperparameters searched were: dropout: [0.0001, 0.001, 0.01, 0.1, 0.5], base_lr: [0.00001, 0.0001, 0.001, 0.01], max_epoch: [1000, 10000, 100,000], and weight_decay: [0.0005, 0.001, 0.005, 0.01, 0.05].

For training the A3T-GCN model, we utilized an ADAM optimizer with learning rate set to 1e-3. Each layer had a dropout of 0.001 and a $l2$ regularization term of 0.001. A period length of 7 was used, and the output channels from the A3T-GCN was 16. The model was trained for 5000 steps. The set of hyperparameters searched were: dropout: [0.001, 0.01], out_channels : [16, 32], periods : [3, 7], and base_lr: [0.00001, 0.0001, 0.001].

4.3. Software Design

Here we provide a brief overview of the software design of this project, providing descriptions of the main scripts and directories shown in figure 8.

4.4. `process_data.py`

This script processes the three datasets within a specified date range between `START_DATE` and `END_DATE`. The `create_torch_geometric_data` function returns a `torch_geometric.data.Data` object to be used for the GCN model. The `create_torch_geometric_temporal_data` function returns a `torch_geometric_temporal DynamicGraphTemporalSignal` object representing

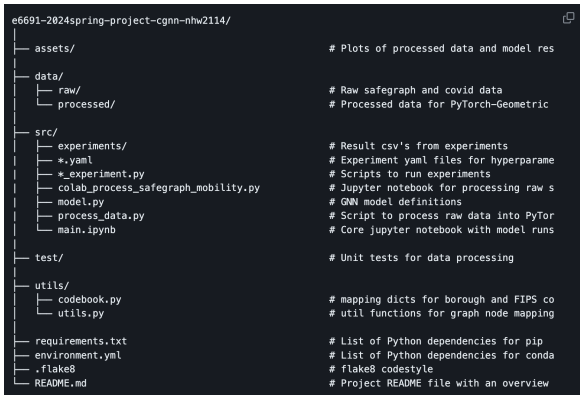

Figure 8. Repository directory tree.

temporal signals (temporal node features) defined on a dynamic graph (changing edge weights). The sizes of each of these networks are outlined in table 2. A key processing step that we perform that was not explicitly stated in the original paper was that we perform a 7-day moving average smoothing of the case and death data. This is a standard process performed on epidemiological time-series data due to the weekend reporting effect (case and death counts from the weekend are often reported on the following Monday). It is not of interest to learn this weekend effect when training our models.

Network	Nodes	Snapshots	Edges
Static Network	460	NA	5275 (2300 spatial, 2975 temporal)
Dynamic Network	5	92	25

TABLE 2. DESCRIPTION OF THE STATIC `torch_geometric.data.Data` USED FOR THE GCN MODELS AND DYNAMIC `torch_geometric_temporal` `DynamicGraphTemporalSignal` USED FOR THE A3T-GCN MODEL.

4.5. `colab_process_safegraph_mobility`

This notebook processes the raw safegraph data. The raw data is not shared due to data use agreement with our research group. The main pre-processing steps were:

- Map postal code of each place of interest (POI) to borough
- Drop rows with no postal code/ borough (around 2% of rows)
- Since data was weekly, we map each day to the most previous Sunday
- To define spatial edge weights we aggregate the `visitor_home_aggregation` values from destination to origin

4.6. `model.py`

This script defines the PyTorch modules for each model used in this project.

- `RMSLELoss`: A custom loss function that calculates the Root Mean Squared Logarithmic Error (RMSLE) between predicted and actual values.
- `AttentionGCN`: A neural network module that combines an Attention Temporal Graph Convolutional Network (A3TGCN) with a Multi-Layer Perceptron (MLP) for prediction. It utilizes attention mechanisms to capture temporal dependencies in the graph data. The detailed model architecture is shown in figure 9.
- `GCN`: Another neural network module that implements a Graph Convolutional Network (GCN) architecture with skip connections and a embedding and prediction MLP.
- `prevCase`: A simple model that predicts the number of cases to be the same as the previous day, effectively predicting a delta of 0. The detailed model architecture is shown in figure 10.
- `prevDelta`: A model that predicts the delta in the number of cases to be the same as the delta from the previous day, based on historical data.

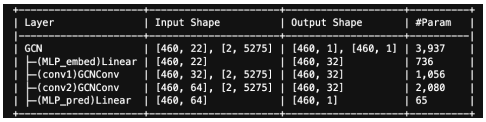

Figure 9. GCN architecture.

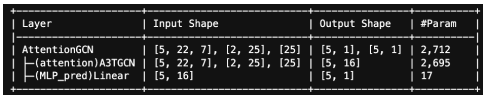

Figure 10. A3T-GCN architecture.

4.7. `main.ipynb`

This is the core Jupyter notebook with that runs data pre-processing, model runs, and outputs.

4.8. `experiments.py`

These scripts ran the experiments for hyperparameter tuning. Each script takes in a YAML file that defines which hyperparameter values to use through a nested for loop. Each hyperparameter set is ran three times to account for stochasticity of the optimization algorithms. The average training and validation loss is stored and output into CSV in the `experiment` directory.

4.9. `utils/`

We wrote several utility functions for data preprocessing and plotting including mapping dicts for borough and FIPS code and functions for graph node mapping.

4.10. `tests/`

We wrote several unit tests for data processing to be run every time a new commit is made. Since the `torch_geometric.data.Data` and `torch_geometric_temporal DynamicGraphTemporalSignal` object formats were quite complicated, we performed simple checks to make sure the node and edge features were stored as expected.

5. Results

5.1. Project Results

Here we show the model performances of the baseline Previous Cases and Previous Delta models compared against the original paper architecture GCN† model, the GCN* with tuned hyperparameters, and A3T-GCN model. The A3T-GCN performs either best or second best across all metrics with the exception of Δ Corr.

As shown in figures 12 and 13, the training and validation loss decreases as the number of training steps increase. The loss is not monotone due to the stochastic nature of the optimizer and model.

We also perform hyperparameter tuning for the GCN* and A3T-GCN models. The validation loss for different hyperparameters are shown in figures 14-20. For each set of hyperparameters, three model runs were ran due to the stochastisticity in the optimization.

Model	RMSLE	Corr	ΔRMSLE	ΔCorr
Previous Cases	0.0626	**0.9977**	2.3047	NaN
Previous Delta	0.0606	0.9968	**0.9174**	0.6210
GCN†	0.0685	0.9958	1.2066	0.5270
GCN*	0.0605	0.9970	1.0098	**0.6498**
A3T-GCN	**0.0550**	0.9972	1.0066	0.3609

TABLE 3. SUMMARY OF MODEL PERFORMANCES. †USING THE SAME HYPERPARAMETERS AS THE ORIGINAL PAPER. *USING HYPERPARAMETERS TUNED FOR THIS PROJECT. **BOLDED** VALUES INDICATE THE BEST METRIC. UNDERLINED VALUES INDICATE THE SECOND BEST.

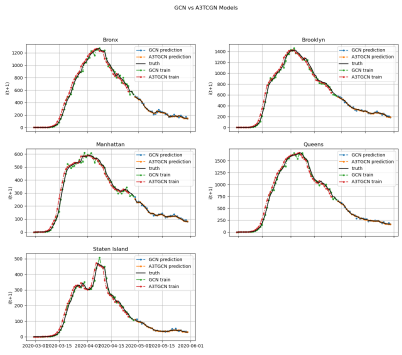

Figure 11. GCN* and A3T-GCN model predictions.

Model	Num of Params	Num of steps ran	Total train time
GCN†	3,937	1,000,000	1:33:58
GCN*	3,937	100,000	0:05:58
A3T-GCN	2,712	5,000	0:17:22

TABLE 4. MODEL SIZE AND TRAINING TIME

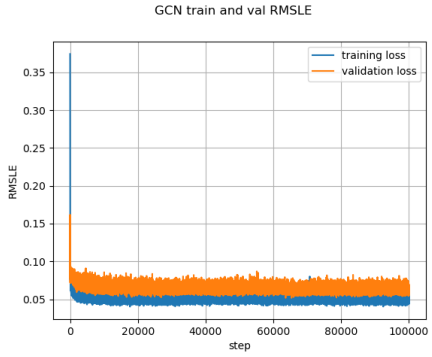

Figure 12. GCN* training and validation loss.

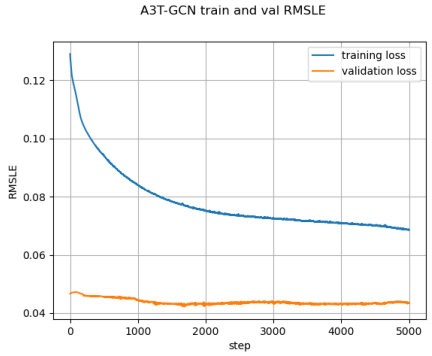

Figure 13. A3T-GCN training and validation loss.

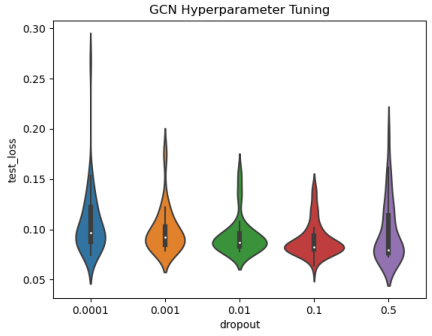

Figure 14. GCN* dropout hyperparameter tuning.

5.2. Comparison of the Results Between the Original Paper and Students' Project

As seen in tables 1 and 3, the scales of the validation RMSLE and Δ RMSLE are quite different. However, the rankings of the baseline and GNN models were similar.

The A3T-GCN had the lowest RMSLE followed by

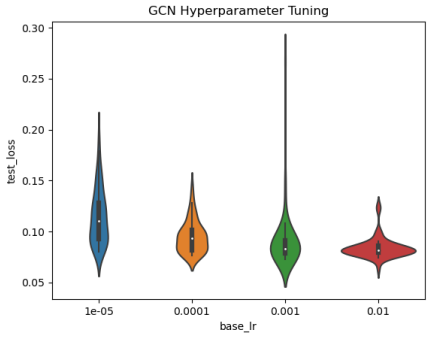

Figure 15. GCN* base_lr hyperparameter tuning.

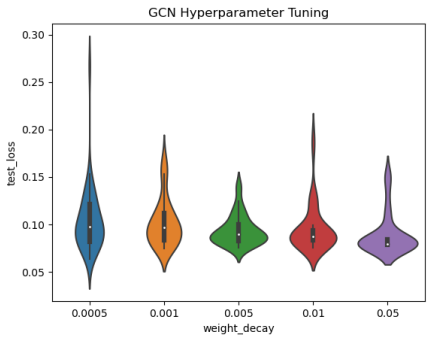

Figure 16. GCN* weight_decay hyperparameter tuning.

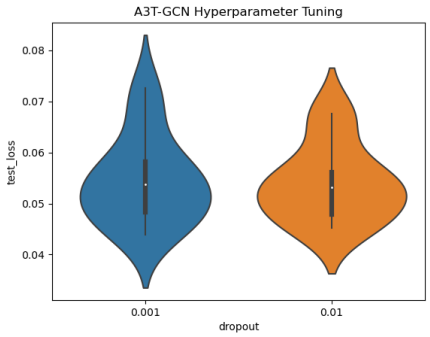

Figure 17. A3T-GCN dropout hyperparameter tuning.

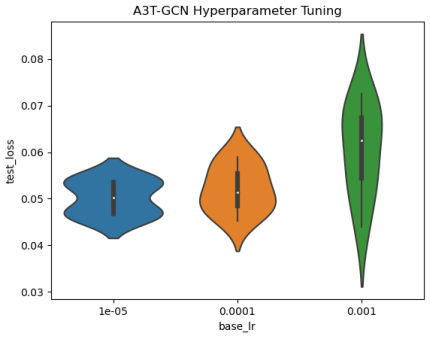

Figure 18. A3T-GCN base_lr hyperparameter tuning.

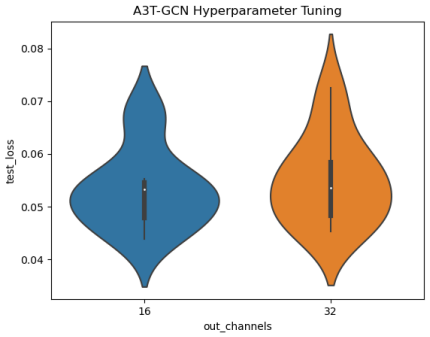

Figure 19. A3T-GCN out_channels hyperparameter tuning.

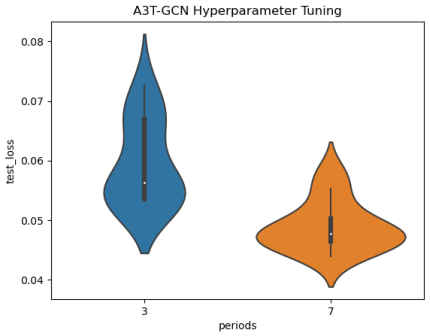

Figure 20. A3T-GCN num periods hyperparameter tuning.

GCN* and Previous Delta. This is similar to the original paper where GCN had the lowest RMSLE followed by Previous Delta and Previous Cases.

Similar to the original paper, the Previous Cases model had the highest Pearson correlation score followed by the GCN model. The A3T-GCN had the second highest correlation score followed by the GCN* model.

The Δ RMSLE and Δ Corr scores had different from the original paper, but in our results, the either the GCN* or the A3T-GCN had the best or second best model performances.

The original paper did not report training time, but as shown in table 4, the GCN models train much faster per step compared to the A3T-GCN model. This is because of the recurrent nature of the GRU modules within the A3T-GCN model.

5.3. Discussion / Insights Gained

The differences in the scales of RMSLE between our results and the original paper is largely due to the fact that we used a smaller dataset only using data from NYC. During the beginning of the pandemic, NYC had a much larger caseload than the rest of the country which helps explain why our RMSLE values are higher. We also used different

mobility data that was highly biased towards mobility into and within Manhattan. We trained two GCN models one with the same hyperparameters as the original paper and another set that was tuned to our data. It did not make sense to train for 1M steps as the authors did for this much smaller dataset and use a high dropout of 0.5 for a smaller GCN model.

One insight as two why the A3T-GCN model performed better than the GCN models was because of the attention mechanism to better learn temporal trends. Whereas the static spatio-temporal network used for the GCN models only had temporal edges connecting each county node to its respective county nodes in previous days, the dynamic spatio-temporal network used for the A3T-GCN model is better able to use temporal information from previous information from other county nodes and weight that information useing the attention mechanism. The downside is that the training time is much longer.

6. Future Work

In future work, we aim to enhance the comprehensiveness of our epidemiological modelling framework by addressing several issues with the data used. First, we intend to integrate data from more if not all of the counties within the United States. As shown in the results, the boroughs of New York City have very similar epidemic curves (i.e., their time scales are similar). The mobility data would be more informative to the models if the epidemic curves varied more across the county nodes (i.e., the epidemic seeded in NYC and then due to population movement sparked outbreaks in other major cities). This expansion will enable a more comprehensive understanding of population movements and their implications for epidemic spread. Additionally, addressing the challenge of underreporting of cases remains paramount. We plan to implement methods to account for this underestimation, ensuring more accurate estimations of disease burden and transmission rates. Furthermore, our future investigations will involve testing the model against different epidemic waves and significant dates, such as school closures, vaccination campaigns, and other non-pharmaceutical interventions.

7. Conclusion

In this project, we successfully implemented and compared the Graph Convolutional Network (GCN) architecture outlined in Kapoor et al.'s study [10]. Our analysis included a comparison against a modified attention temporal graph convolutional network (A3T-GCN) [15]. Notably, the A3T-GCN model achieved the lowest Root Mean Squared Logarithmic Error (RMSLE) of 0.0550, outperforming our implementation of the original paper's architecture, which yielded an RMSLE of 0.0605. For both models, we performed hyperparameter tuning on the learning rate, dropout, weight decay, number of output channels, and number of periods in the attention window. Our project achieved its objectives of implementing and evaluating the GCN architecture. Through this process, we gained valuable insights into building epidemiological models using mobility data.

8. Acknowledgement

I would like to thank Professor Kostic and the teaching assistants of his E6691 class for their support and help. I would also like to thank my advisors Professor Shaman, Professor Spiegelman, and Professor Pei for their feedback.

References

[1] "Link to the github for this project." [Online]. Available: https://github.com/ecbme6040/e6691-2024spring-project-cgnn-nhw2114/tree/main

[2] "Link to the slides for the final presentation." [Online]. Available: https://github.com/ecbme6040/e6691-2024spring-project-cgnn-nhw2114/tree/main

[3] "Covid-19: Data trends and totals - nyc health." [Online]. Available: https://www.nyc.gov/site/doh/covid/covid-19-data-totals.page

[4] R. Li, S. Pei, B. Chen, Y. Song, T. Zhang, W. Yang, and J. Shaman, "Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (sars-cov-2)," *Science*, vol. 368, no. 6490, pp. 489–493, 2020.

[5] S. Chang, E. Pierson, P. W. Koh, J. Gerardin, B. Redbird, D. Grusky, and J. Leskovec, "Mobility network models of covid-19 explain inequities and inform reopening," *Nature*, vol. 589, no. 7840, pp. 82–87, 2021.

[6] I. C.-. H. S. U. F. Team and C. J. Murray, "Forecasting covid-19 impact on hospital bed-days, icu-days, ventilator-days and deaths by us state in the next 4 months," *MedRxiv*, pp. 2020–03, 2020.

[7] D. Benvenuto, M. Giovanetti, L. Vassallo, S. Angeletti, and M. Ciccozzi, "Application of the arima model on the covid-2019 epidemic dataset," *Data in brief*, vol. 29, p. 105340, 2020.

[8] Z. Yang, Z. Zeng, K. Wang, S.-S. Wong, W. Liang, M. Zanin, P. Liu, X. Cao, Z. Gao, Z. Mai *et al.*, "Modified seir and ai prediction of the epidemics trend of covid-19 in china under public health interventions," *Journal of thoracic disease*, vol. 12, no. 3, p. 165, 2020.

[9] C. Fritz, E. Dorigatti, and D. Rügamer, "Combining graph neural networks and spatio-temporal disease models to improve the prediction of weekly covid-19 cases in germany," *Scientific Reports*, vol. 12, no. 1, p. 3930, 2022.

[10] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O'Banion, "Examining covid-19 forecasting using spatio-temporal graph neural networks," *arXiv preprint arXiv:2007.03113*, 2020.

[11] "GitHub - nytimes/covid-19-data: A repository of data on coronavirus cases and deaths in the U.S. — github.com," https://github.com/nytimes/covid-19-data, [Accessed 10-05-2024].

[12] "COVID-19 Community Mobility Report — google.com," https://www.google.com/covid19/mobility/, [Accessed 10-05-2024].

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[14] "Places Data Curated for Accurate Geospatial Analytics — SafeGraph — safegraph.com," https://www.safegraph.com/, [Accessed 10-05-2024].

[15] J. Bai, J. Zhu, Y. Song, L. Zhao, Z. Hou, R. Du, and H. Li, "A3t-gcn: Attention temporal graph convolutional network for traffic forecasting," *ISPRS International Journal of Geo-Information*, vol. 10, no. 7, p. 485, 2021.

[16] "GitHub - nychealth/coronavirus-data: This repository contains data on Coronavirus Disease 2019 (COVID-19) in New York City (NYC), from the NYC Department of Health and Mental Hygiene. — github.com," https://github.com/nychealth/coronavirus-data, [Accessed 10-05-2024].

9. Appendix

9.1. Individual Student Contributions in Fractions

Since there was only one group member, Han Yong Wunrow (nhw2114) completed all of the work for this project.