A GNN based prediction of COVID-19 case flow within Hong Kong

NATARAJAN, Siddarth Jayakumar
HKUST
sjnatarajan@connect.ust.hk

THIEN, Zhong Vei
HKUST
zvthien@connect.ust.hk

LIM, Her Wei
HKUST
hwlim@connect.ust.hk

Abstract

The COVID-19 pandemic has turned into a stark reminder of our lack of preparedness against novel corona viruses. Hence, to further protect Hong Kong from the insurgence of future corona viruses we utilized Graph Neural Networks (GNN) to predict the flows of the COVID-19 virus in hopes to create better policies, and increase our safeguards against these viruses. From our investigation, with the use of 2-months of training data from the peak of the fifth wave, we were able to predict COVID-19 within a 3% error-rate, on average.

1. Introduction

On January 23, 2020, the COVID-19 virus was confirmed to have spread to Hong Kong for the first time. As of 24 November 2022, a total of 2,068,310 cases and 10,647 deaths due to COVID-19 were confirmed in Hong Kong. In the fifth wave alone, the highly infectious Omicron variant caused nearly 3000 deaths over the span of less than 3 months. Accurately predicting the spread of the coronavirus is critical for governments and policymakers in order to impose anti-virus measures or decide how to allocate healthcare resources in a pandemic.

The intensity of the pandemic has led researchers to implement machine learning forecasting methods for the disease spread, however the problem is complex with a multitude of factors. To accurately predict the spread of COVID-19, on paper at least, it is necessary to obtain the mobility data and social networks of the entire population. Without the availability of such data, we approach our problem differently: predict the rate of positive case spread of the disease based on the volume of recent cases in the area. The number of new positives in the area can be linked to the spread of the disease, i.e. the more people with COVID-19 visit an area, the higher the chance of citizens in that area becoming infected. The availableness of such information from HKSAR's open source datasets allows us to generate a graph representation of the magnitude of recent COVID-19 cases and the correlation of districts. Thus, we can deduce a natural graph representation of our problem, and enabling us to implement Graph Neural Networks (GNN) on our dataset.

We focalize on the prediction of future cases on each node. We propose a model that captures representations of neighbor nodes, thus combining neighboring district cases with the history of cases within itself.

2. Brief Literature Review

There are plenty of papers making predictions on COVID-19 with the use of machine learning. For example, Panagopoulos et al. (2021) proposed a GNN model to capitalize on the underlying relationship between movement of people and the spread of COVID-19, with the addition of a Model-Agnostic Meta-Learning (MAML)-based transfer learning method to leverage on knowledge from other countries' models [2]. Karwowski [et al.] (2021) studies the effectiveness of using a Graph theory based Graph Neural Network (GTNN) and a Neighbourhood-based Graph Neural Network (NGNN) effect on the prediction of Rt values (COVID-19 spread rate) within US states. This is all done whilst using a LSTM as a base model of comparison [1].

3. Dataset

As mentioned earlier, The Government of the Hong Kong Special Administrative Region (HKSAR) has released several datasets for the general public. The raw data contains the full list of buildings visited by cases tested positive for SARS-CoV-2 virus in the past 7 days.

We extracted the data from the peak of the fifth wave and compiled daily cases into the 18 Hong Kong districts. Figure 1 provides an overview of the preprocessed data. Due to erroneous district names from typographical mistakes (e.g. Yau Tsim Wong instead of Yau Tsim Mong), we used a k-nearest neighbors implementation in order to match the closest name to the list of districts. From initial inspection, it is clear that there are districts with higher correlations between each other.

This peak data (between Feb 1st to March 31st 2022) was used as our training data set, where we applied a four day

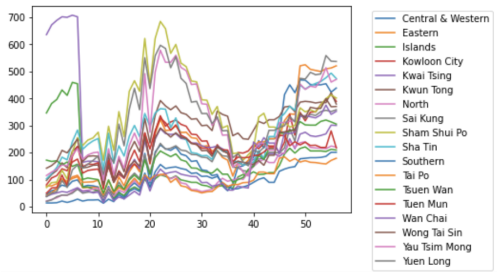

Figure 1. The aggregated cases of the 18 districts of Hong Kong during the absolute peak of the fifth wave, between February 1 and March 31.

sliding window to predict one day ahead. Moreover, we developed two testing data sets. One continuous, and another discontinuous. For our continuous data set, we used the cases data between April 1st to May 31st (i.e. the data right after our training data) and for our discontinuous data set we used the cases data between September 1st to October 31st (i.e. 7 months after our training data). We used these two data sets to evaluate the extent our models were able to evaluate the flow of COVID-19 within Hong Kong.

4. Methodology

4.1. Graph Construction

We compute the pearson correlation from the time series of COVID-19 cases in order to assess the relationship between individual districts. Obtained from hyperparameter tuning, we discovered and utilized a threshold of 0.81, such that entries in the pearson correlation matrix that is greater than or equal to the threshold will be represented as 1 (connected) where as the others will be represented as 0 (disconnected). This led to a well connected graph which exhibits accurate depictions in the . One example is the disconnection of the Kwai Tsing and Islands nodes from other districts nodes, suggesting a lower flow of people, and thus transmission of COVID-19, to and from the less populated neighborhoods. This methodology was inspired by Mohammad Reza et. al's graph construction method for covid 19 case prediciton within the United States [1]. Within this graph we provided each node a feature being a 4 day sliding window of the number of covid cases within the district. (Figure 1) On top of this graph, we designed two two temporal graphs. In the first temporal graph (Figure 2) we had $18 \times windowsize$ number of nodes. We had each node i_t (where i represent district i, of time t) have an edge to node i_{t-1} and an edge to all other nodes j_t, where j is connected to i if the Pearson correlation of i and j is greater than 0.81. In the second version ((Figure 3), we kept the same structure as the previous temporal graph, but we added an edge between each node i_t and all node i_n (where $1 \leq n < t$).

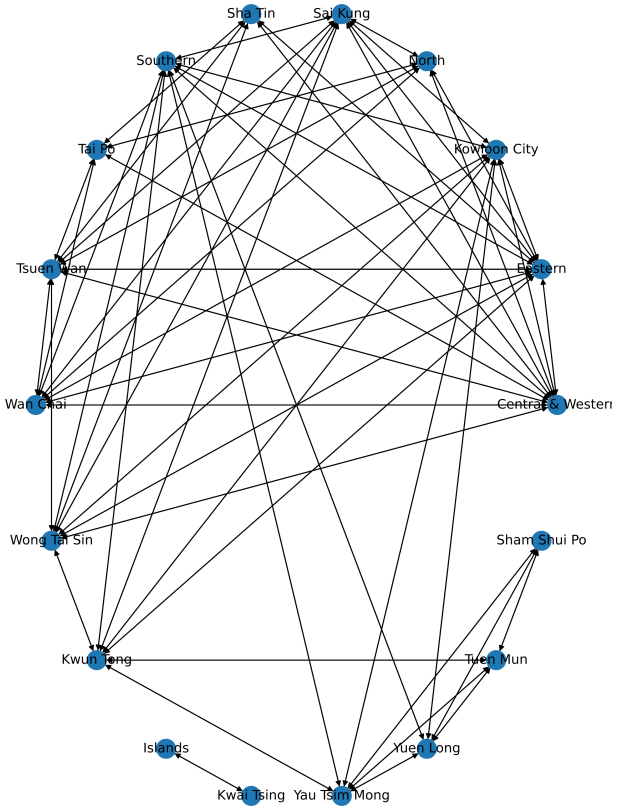

Figure 2. Time-Invariant Graph

4.2. COVIDGraphSAGE

Our COVIDGraphSAGE model utilizes three layers on GraphSAGE convolution followed by two layers of max pooling. The network has a hidden dimension of 10 with a ReLu activation function, followed by a 1-D max pooling of stride 6 and then a 1-D max pooling of stride 4. From preliminary testing, we found double max pooling will result in a faster convergence compare to a fully connected layer while decreasing the number of trainable parameters. Hence, this is why we implemented the double max pooling in place of a fully connected layer.

Initially we use the time invariant graph (static) and pass the whole graph into the model normally with the learning rate and weight decay shown in the figure below. We then utilized three different dynamic graph sub-sampling methods (Node, Edge, Walk) and kept the learning rate and weight decay constant. The parameters for the node sub-sampling was just a upper bound of the number of nodes within the graph; Edge follow similarly with an upper bound of the number of edges within the graph. However

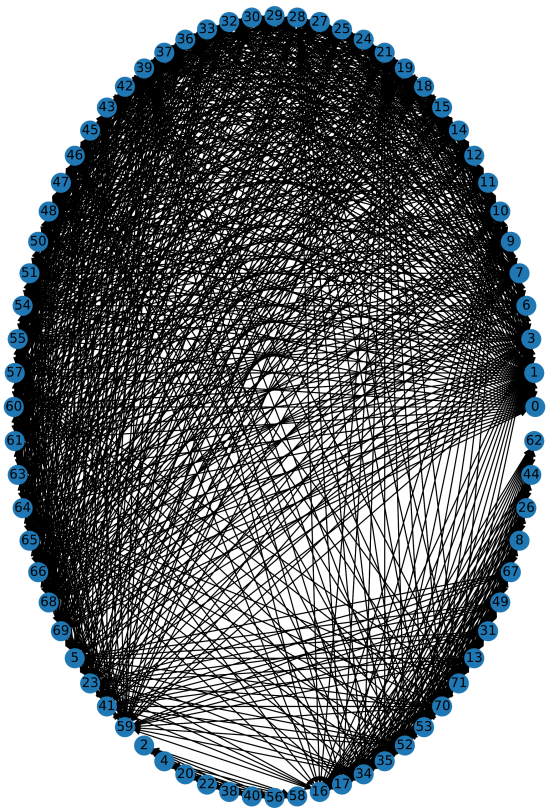

Figure 3. Temporal Graph

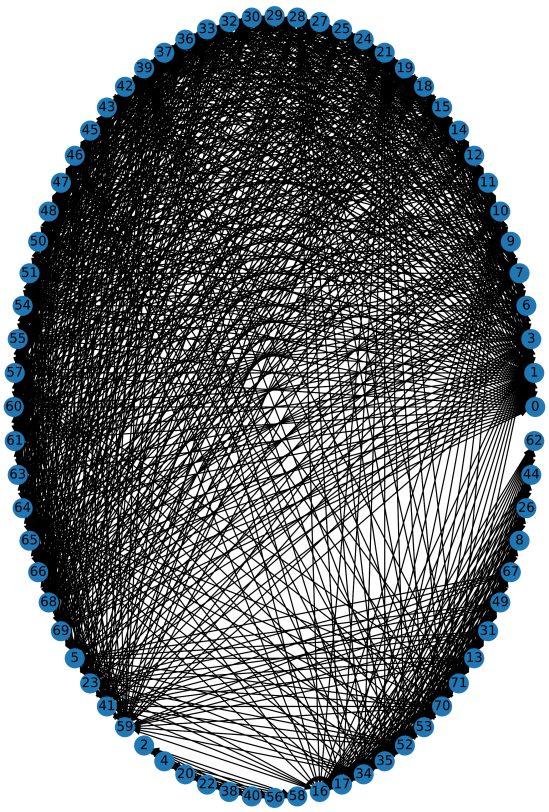

Figure 4. Tight Temporal Graph

for the walk, we gave an upper bound for the source node being the number of nodes within the graph and restrict the walk length to be the window size which is the size of the sliding window.

For our temporal graphs, we only implemented them dynamically as we saw a significant performance improvements and speed in training between the two time invariant graph versions. Moreover, our dropout rate for overall was 0.5 while using the Adam Optimizer with the MSE loss function, and 150 epcohs of training. The learning rate and weight decay for our different models is listed in the figure below.

Graph	Mean	GCN	Pool	LSTM
Time invariant Graph (static)	2×10^{-4}	1.6×10^{-4}	2×10^{-4}	2×10^{-4}
Time invariant Graph (dynamic)	2×10^{-4}	1.6×10^{-4}	2×10^{-4}	1.7×10^{-4}
Temporal Graph	2×10^{-4}	1.9×10^{-4}	2×10^{-4}	2×10^{-4}
Tight Temporal Graph	2.2×10^{-5}	1.6×10^{-4}	2×10^{-4}	2×10^{-4}

Table 1. Learning Rate

Graph	Mean	GCN	Pool	LSTM
Time invariant Graph (static)	1.5×10^{-5}	2×10^{-8}	1×10^{-5}	1×10^{-5}
Time invariant Graph (dynamic)	1×10^{-5}	2×10^{-8}	1×10^{-5}	1×10^{-5}
Temporal Graph	1×10^{-5}	2×10^{-6}	1×10^{-5}	1×10^{-5}
Tight Temporal Graph	1×10^{-7}	2×10^{-8}	1×10^{-5}	1×10^{-5}

Table 2. Weight Decays

4.3. Baseline Method

To fully evaluate the extend of graph structure assists within the flow prediction of covid-19 predictions in Hong Kong, we decided to compare it with the traditional method of LSTM to predict this time series, as talked about in Karwowski's paper [1].In the LSTM model proposed by Karwowski's a 1-to-1 prediction was used in the sense that one district time series was use to predict that district own future cases. However, we thought of supplementing our baseline with correlational information and hence developed a second LSTM model which uses a N-to-1 prediction model. In this model, the districts with the highest similarity to a district x (Pearson Correlation ¿ 0.81) was use as the additional

input for the model to predict the future case for district x.

The LSTM model has two layers of LSTM cells followed by a fully connected layer. We used 150 epochs with a 0.01 learning rate for both baseline models. The optimizer used was LBFGS with MSE as the loss function. The learning rate and Weight decays we used for our LSTM models can be seen in the table below.

LSTM Version	Learning Rate
1 to 1	0.01
1 to N	0.01

Table 3. LSTM Parameters

5. Evaluation

5.1. Evaluation Metric

To evaluate the performance of the model, we used the SMAPE (Symmetirc Mean aboslute Perentage Error) scaled between 1 to 0 to determine the error rates of our models. We then calculate the mean, and variance of the error rates to determine the accuracy and precision of the predictions of the various models. The formula for the SMAPE metric is shown below. (A_t is the actual value and F_t is the predicted value)

$$SMAPE = \frac{1}{n} \times \sum_{t=1}^{n} \frac{|F_t - A_t|}{(A_t + F_t)/2}$$

We decided to use this metric as it was the same evaluation metric used by Karwowski [1], and since our work is additive to his it made sense to keep the same metric.

5.2. Results

5.2.1 Continuous Dataset

LSTM Version	SMAPE	Standard Deviation
1-to-1	0.5423	0.2377
1-to-N	0.3450	0.1634

Table 4. Baseline Model

We ran all of our models, initially on the continous testing data set. Based on the results table above it can be seen that by using the original LSTM implementation proposed by Karwowski [1] we get a high error rate of about 54%. However, when utilizing the method of LSTM we proposed, which considers correlational information of the districts, we are able to gain a lower error rate with a lower standard deviation of errors. Thus, providing an overall improved baseline model.

Aggregation Method	SMAPE	Standard Deviation
Mean	0.0863	0.0149
GCN	0.1433	0.0187
Pool	0.0388	0.0148
LSTM	0.0411	0.0255

Table 5. Time Invariant (static) Model

When using the initial time invariant graph we had developed, as inspired by Karwowski [1], we were able to obtain the above results via the different Aggregation methods of GraphSAGE. The lowest of which occurred, for pooling, and the highest of which occurred within the GCN aggregation method. However, two interesting things can be observed. Firstly, the standard deviations amongst the different aggregation methods seems quite similar, with LSTM being slightly more inaccurate. Secondly, all of the different aggregated versions of the static training of our time invariant graph provided better SMAPE results, and lower standard deviation within the SMAPE, compared to the baseline by a significant margin. Thus, providing evidence of the significance of a graph structure in predicting the flow of COVID-19 cases within Hong Kong.

Aggregation + Sub-Graph Method	SMAPE	Standard Deviation
Mean + Node	0.0541	0.0172
Mean + Edge	0.0562	0.0211
Mean + Walk	0.0648	0.0117
GCN + Node	0.1422	0.0180
GCN + Edge	0.1515	0.0206
GCN + Walk	0.1501	0.0200
Pool + Node	0.0568	0.0220
Pool + Edge	0.0823	0.0196
Pool + Walk	0.0586	0.0193
LSTM + Node	0.0350	0.0228
LSTM + Edge	0.0366	0.0253
LSTM + Walk	0.0404	0.0266

Table 6. Time Invariant (dynamic) Model

From the advice of Professor Yangqiu Song, we had further implemented a dynamic training of the time-invariant graph we had developed. Within this methodology we tried out all cross combinations of the SAINTSampler Subgraph methodology, with all the GraphSAGE aggregation methods to generate the table above. From which we found that the SMAPE and standard deviations of the results were lower then that of the static version. Thus implying that this version of the model was more stable and accurate within our continious testing set. However, the margin of improvement between the dynamic and static is significantly smaller between the improvement of static time invariant to our baseline.

From pure curiosity, we had considered if the use of a temporal graph, instead of the time-invariant one proposed

Aggregation + Sub-Graph Method	SMAPE	Standard Deviation
Mean + Node	0.1574	0.0165
Mean + Edge	0.1577	0.0165
Mean + Walk	0.1609	0.0181
GCN + Node	0.1542	0.0161
GCN + Edge	0.1570	0.0162
GCN + Walk	0.1571	0.0162
Pool + Node	0.1652	0.0146
Pool + Edge	0.2247	0.0211
Pool + Walk	0.1696	0.0141
LSTM + Node	0.2149	0.0196
LSTM + Edge	0.2148	0.0188
LSTM + Walk	0.2111	0.0176

Table 7. Temporal Graph Model

by Karwowski , would lead to better results. Hence we tried the same Subgraph and aggregation cross combinations as before, on our temporal graph and obtained the results above. However, the results of this version (both SMAPE and standard deviation) were much worse than our time-invariant graphs. Although, one intersting observation is that the GCN aggregation method used to be the most inaccurate method for the time-invariant graphs, while now the LSTM seems to be the least inaccurate methodolgy. We believe this is because with the temporal graphs there are more nodes, each with less features. Thus, LSTM aggregations are more ineffective with the less feature dense, and high frequency nodes as there is less feature information to learn from each node and pass on. This lower information aggregation most probably causes this higher SMAPE and standard deviation of errors. We initially suspected, insufficient message passing as the leading cause for lower error rates within the temporal graphs. Thus, to introduce more message passing we developed the tighter temporal graph proposed earlier. However, from the results observed earlier, it can be seen that the tighter temporal graph performed worse than the temporal graph model. Furthermore, within the Mean aggregation, the model actually performed worse than the baseline. With further investigation, we realised that introducing this many edges actually caused excessive aggregation and lead to an over smoothing problem for the Mean aggregation. Thus, causing to results even worse than the baseline at times.

Aggregation + Sub-Graph Method	SMAPE	Standard Deviation
Mean + Node	0.5758	0.0307
Mean + Edge	1.0000	0.0000
Mean + Walk	0.1538	0.0176
GCN + Node	0.1552	0.0162
GCN + Edge	0.1570	0.0163
GCN + Walk	0.1571	0.0163
Pool + Node	0.1853	0.0129
Pool + Edge	0.1949	0.0202
Pool + Walk	0.1624	0.0171
LSTM + Node	0.2183	0.0191
LSTM + Edge	0.2103	0.0170
LSTM + Walk	0.2145	0.0184

Table 8. Tight Temporal Graph Model

5.2.2 Discontinuous Dataset

When we rerun the same pretrained models within our discontinuous testing data (which takes place 7 months later) set we find that the results (All results within the Appendix) for the tight temporal and temporal graph performed similar to within the continuous data set. Hence, indicating to us that these models, although not accurate, seem to possess a grasp of the structural flow of COVID-19 cases within Hong Kong.

However, two key observations were made between the two datasets. The first being that withint the Discontinous dataset the 1-to-N LSTM baseline model actually performed much worse than the 1-to-1 LSTM model. Although, both version of the Time-invariant graph models showed worse error rates within the Discontinious dataset, what was interesting was that within the static-time invariant model now had the best results for the LSTM aggregation method. Thus, one of the two key takeaways from this is , the correlations between districts is hightly dynamic and hence, a one time LSTM based correlational graph will only work for so long. The other key takeaway is, with the time-invariant graphs we give low node density and high feature density, and thus can utilize LSTM aggregation to obtain the best results.

6. Conclusion

In this paper we investigated the extent to which a Graph Neural Network can aid with the prediction of COVID-19 flow within Hong Kong. From our investigation we had found that the use of time invariant graphs, over temporal graphs, lead to the lowest error rates. However, the difference in performance between our continuous and discontinuous data sets makes it clear to us that both dynamic and static time invariant graph models should be utilized when trying to predict the flow of COVID-19 cases. By using, both models the results can be more rigorously used within the implementation of policies to reduce the flow of COVID-19 or any future novel corona viruses.

One area we think that this paper could still explore, is the possibility of trying to implementing some defense methods whilst training. This can be done to try to counteract the human errors (which can be viewed as adversarial attacks) within our data set.

All the code we developed, and the models we trained can be found within our github page, https://github.com/RajaSJN/COVIDGraphSAGE

References

[1] Karwowski W Aljuaid AM Taiar R Davahli MR, Fiok K. Predicting the dynamics of the covid-19 pandemic in the united states using graph theory-based neural networks. *Int J Environ Res Public Health*, 2021. 1, 2, 3, 4

[2] Michalis Vazirgiannis George Panagopoulos, Giannis Nikolentzos. Transfer graph neural networks for pandemic forecasting. *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, 2021. 1

Appendix A. Discontinuous Testing Result

Aggregation Method	SMAPE	Standard Deviation
1-to-1	0.6162	0.2973
1-to-N	0.6715	0.3873

Table 9. Baseline Model

Aggregation Method	SMAPE	Standard Deviation
Mean	0.0869	0.0405
GCN	0.1286	0.0296
Pool	0.0692	0.0583
LSTM	0.0555	0.0515

Table 10. Time Invariant (static) Model

Aggregation + Sub-Graph Method	SMAPE	Standard Deviation
Mean + Node	0.0732	0.0504
Mean + Edge	0.0663	0.0472
Mean + Walk	0.0838	0.0543
GCN + Node	0.1307	0.0310
GCN + Edge	0.1324	0.0343
GCN + Walk	0.1306	0.0322
Pool + Node	0.0765	0.0626
Pool + Edge	0.0835	0.0458
Pool + Walk	0.0673	0.0481
LSTM + Node	0.0546	0.0489
LSTM + Edge	0.0558	0.0515
LSTM + Walk	0.0566	0.0541

Table 11. Time Invariant (dynamic) Model

Aggregation + Sub-Graph Method	SMAPE	Standard Deviation
Mean + Node	0.1521	0.0209
Mean + Edge	0.1544	0.0239
Mean + Walk	0.1532	0.0240
GCN + Node	0.1509	0.0196
GCN + Edge	0.1537	0.0232
GCN + Walk	0.1537	0.0231
Pool + Node	0.1570	0.0187
Pool + Edge	0.2137	0.0248
Pool + Walk	0.1559	0.0194
LSTM + Node	0.2141	0.0273
LSTM + Edge	0.2170	0.0275
LSTM + Walk	0.2168	0.0275

Table 12. Temporal Graph Model

Aggregation + Sub-Graph Method	SMAPE	Standard Deviation
Mean + Node	0.6024	0.0539
Mean + Edge	1.0000	0.0000
Mean + Walk	0.1496	0.0170
GCN + Node	0.1518	0.0209
GCN + Edge	0.1534	0.0228
GCN + Walk	0.1536	0.0230
Pool + Node	0.1718	0.0207
Pool + Edge	0.1759	0.0245
Pool + Walk	0.1509	0.0181
LSTM + Node	0.2211	0.0282
LSTM + Edge	0.2186	0.0278
LSTM + Walk	0.2185	0.0277

Table 13. Tight Temporal Graph Model