Exploring GNN's as a method of modeling pandemics

Team ID: 07-3J2S

Gary Braznichenko
University of Illinois
Champaign, Illinois
garyab2@illinois.edu

Philip Dohm
University of Illinois
Champaign, Illinois
pdohm2@illinois.edu

Samuel Guo
University of Illinois
Champaign, Illinois
sguo@illinois.edu

Jonathan Lu
University of Illinois
Champaign, Illinois
jjlu2@illinois.edu

Kara Wong
University of Illinois
Champaign, Illinois
karaww2@illinois.edu

***Abstract*–Like many others around the world, the current situation in regards to COVID-19 has led to massive disruptions in all of our lives. As such, two of the most important questions at this time are: When will all of this end? and How bad will it be? To answer these questions, we propose to use Graph Neural Networks as an alternative to current pandemic models as GNNs allow for a cleaner capture of temporal spatial data due to the natural connectivity of our world. By focusing on Canada and the US, we were able to construct graphs to represent each of these countries, train a GNN using COVID-19 data acquired through Kaggle, and make accurate predictions up to one week in the future. While our model was able to supply relatively good forecasted values, we hope to further build upon our initial work in this paper through idea outlined in the future work.**

I. Introduction

Given the meteoric (and catastrophic) rise of COVID-19 throughout the world, we would like to see if neural networks can help answer the pressing issues of how fast will the disease spread [2]. While the COVID-19 virus is relatively new and there has not been much time to collect data on its spread, we were able to find a Kaggle dataset that has the number of infected, deceased, and recovered individuals related to the virus in all reporting countries at each date and their respective provinces/states. In addition to these provided numbers, we intend to research how connected each of these regions and countries are to one another in an attempt to attach a numerical value to the "spread-ability" between regions. Since our focus is on how the virus spreads and develops in a region, we propose that the best way to model a geographical region is with a graph. In this graph we construct, each node will either represent a state/province and the weights on the edges between each node will be some numerical representation of the "connectedness" between the regions.

In order to fully analyze these graphs, we will utilize graph neural networks as the baseline from which we can explore multiple different architectures. The input for each of these architectures will remain relatively consistent. Our data set will be a collection of graphs each representing a country as outlined above. Where these may differ is in how each node is encoded. We will either chose to encode all of the time related data in each node or create individual graphs for each slice of time. While some of the information at each node will be provided, the output will be a graph with each node labeled with a value representing the number of infected people at that time or N steps in the future.

II. Related Work

Recently, a team of engineers at MIT has developed a neural-network model that quantifies the impact of quarantine control on the spread of COVID-19 [6]. In this model, they compared the reproduction numbers of the virus in Wuhan, Italy, South Korea, and the United States of America with regards to each of these countries' specific quarantine and isolation measures. This group leverages the SEIR (Susceptible, Exposed, Infected, and Recovered) epidemic model to capture growth rates. The SEIR model breaks the population into these four categories and models them, as well as respective rates, with differential equations. While the original SEIR model does not capture effects of quarantine strength, the MIT group's augmented model uses a neural network to capture effects of quarantine using SEIR model parameters as inputs.

The MIT group's results show that their model that incorporates isolation measures is able to capture a plateau in infected case count whereas the original SEIR model cannot. The developed model's predictions were also able to reflect actual observed data. Their results show that countries that implement strict public health policies in regards to quarantine and isolation will see improvement in their spread of infection. This study solidifies the fact that social distancing measures play a large role in the spread of the disease and thus must be represented in our model.

The following two papers, 'Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting' [7] and 'T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction' [8] discuss the architecture we'd like to mimic in our project, as they involve modeling spatial relationships over time. While these articles focus on predicting traffic flows, they can be generalized to

fit our objective. More specifically, these networks employ convolutions that have been applied to a graph structure (which is not necessarily a grid like in classical convolutions). One way these can be achieved is by creating levels of spectral clustering of the nodes based on matrix representations of the graph's nodes and edges. These convolutions then can be applied to the temporal dimension to solve issues of large training data and long training time, issues which plague recurrent neural networks and complex gate structures. The models in the paper also handle data that have numerical values such as the number of traffic observations or the speed of traffic as the dependent variable.

In short, the spatial-temporal graph neural network had good performance and is suitable to be used in modelling the spread over time of COVID-19 for our project. Of course, adaptations can and will be made to suit the details and time/complexity limitations of our project. We will employ a simpler method of performing graph convolutions and possibly use an LSTM structure, as outlined later in the report.

When looking for articles specifically related to the use of neural networks in conjunction with infection data, there were surprisingly few results. However, of the results that appeared, most chose to instead try and classify whether or not an area is of high risk. In "A dynamic neural network model for predicting risk of Zika in real time" an article authored by Akhtar [9], Kraemer, and Gardner in September 2019, they explored the applicability of neural networks to the spread of Zika. The reason why we wanted to highlight this article is that the input data that they used for these classifications parallels our ambitions. They utilized flight data between the Carribeans and mainland US to capture the "connectedness" of these two areas. They also utilized the socioeconomic statuses of each of those countries in this model. Their usage of this data represents the node embedding that we hope to achieve in our GNNs. While their data is extremely similar to what we hope to find for our graph, they're missing the connected nature that our graph manages to achieve solely because of the structure of input data. In their model, they managed to achieve 85% accuracy forecasting risk up to 12 weeks into the future. This is extremely impressive as it shows that a simpler feed-forward neural network is able to capture some of the spatial relationship in data. Since GNNs further expand on the spatial relationship between data, we are very optimistic for our results even though prediction is much harder.

While most of the academic articles that we found focused on node/graph classification, we managed to find one instance of node prediction. In a paper authored by Di Zhu and Yu Liu titled "Modelling Irregular Spatial Patterns using Graph Convolutional Neural Networks" [2]. In this article, the pair explore predicting the number of intraday check ins to each "POI" or point of interest. The reason why they chose to use a GNN is due to the spatial relationship between these POIs. It can be assumed that most people who chose to go to such locations are oftentimes tourists which means it is highly likely that they would go to another POI that is close by. While they captured the spatial relativity of the data by simple using the distance in miles between the POIs, they also attempted to include a temporal aspect by including the number of visitors to a set POI and its "neighbouring" POIs lagged by a certain time frame. By using these values, they created a spatial-temporal GNN, one which is extremely similar in nature to the graph that we will construct. The main difference will be how we assign values to the edges in our graph as we are aiming to try and quantify the movement between both our nodes (states and/or provinces). However, at this time, it must be noted that most of the data that we would hope to acquire on this is often locked behind a paywall of sorts. While their model seemed to consistently overpredict the number of visits to each individual POI, the heatmaps they constructed in regards to their predictions seemed to match the actual values relatively well as it managed to highlight the most visited areas with pretty impressive precision. Their results strengthen the ideas on which we base our current project on as it shows that GNNs are able to successfully incorporate spatial and temporal data, an assumption that our project hinges on. Funnily enough, the tendency to overpredict is also a good thing as in the case of viral transmission, overestimating is always better than underestimating infection.

III. Data

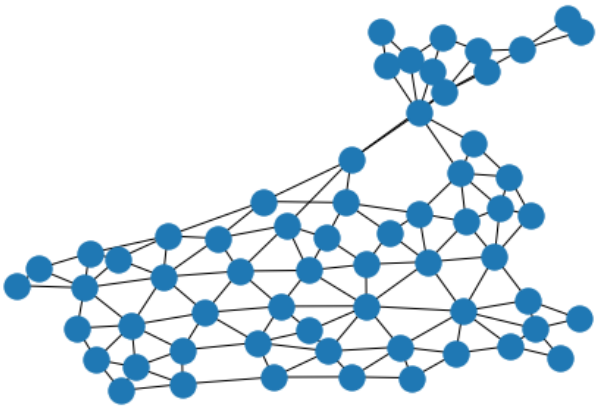
The data we are using comes from Kaggle's Novel Corona Virus 2019 Dataset. It is a CSV file in the format of the number of confirmed cases, deaths, and recoveries from COVID-19 by each reporting country (and province/state/county when applicable) by observation dates starting in late January.

The size of the data is 991.6 KB and it contains 16409 entries across 86 different observation dates since the start of the recording. It is important to note that the dataset is being continuously updated whenever new figures are being released, therefore we will have more and more data to work with and test on as time goes on. We decided to cut the training data at the end of April to start testing the model's accuracy.

We utilized a training testing split that had 100 days of training data starting from January 22, 2020 up until April 30, 2020 and then used a week of data from May 1st to May 8th to test on. This allowed us to evaluate the models ability to predict future data and gather enough training data for all regions to be reporting at a standard accuracy.

In order to properly utilize the data in our network architecture we pre-processed the Kaggle dataset to remove variables that we felt were irrelevant to prediction, such as "last update" and then further restructure the data to use the observed date as the key and record the number of cases, deaths, and recoveries for each reporting region available for each date. When fully constructed, our network had data for 50 US regions and all Canadian provinces for COVID-19 cases and deaths reported. However, due to Hawaii's lack of land-neighbors, we removed Hawaii from our graph which left us with 49 US states. In order to format the data towards the graph based architecture we utilized networkx to create a graph where each node was represented by a reporting region and its associated numbers and the edges symbolizing land

borders between all these regions. We would like to explore incorporating the presence of international airports in states and regional population density to see if it would significantly affect our predictions.



This is a visualization of the network created using Canadian provinces, US states and their connections

IV. Methods

A. Graph Convolutional Network

We used a Graph Neural Network (GNN) architecture called a Graph Convolutional Network (GCN). The easiest and most intuitive way to describe a GCN is a Convolutional Neural Network (CNN) that acts on graph inputs. However, CNN's assume a natural structure to the data (e.g. an image has a natural structure), but graph data is inherently unstructured.

Suppose we have a (weighted and directed) graph G. Now, each node v in G is labeled with some input and output parameters of interest. We must modify the adjacency and degree matrices A and D to create $\tilde{A}$ and $\tilde{D}$, respectively. The modification is adding a "self-connection" to each node. We have the input labels/features X which is $n_v \times n_f$ where n_v is the number of nodes and n_f is the number of input features. We then perform the "convolution" by

$$\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}X.$$

This operation transforms the features in X based on a weighted average of the features of its first order neighbors. We haven't applied weight matrix, and are just aggregating data based on the graph itself. Introducing the weight and bias parameters, we have a more familiar feed-forward structure:

$$A^{(1)} = \sigma(Z^{(1)}) = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW^{(1)} + b^{(1)}).$$

Here, σ is just some nonlinear activation. We can continually stack similar layers, until a final output layer. We omit these details since the interest lies in graph convolutions, and we all know how an MLP works. Generally, more convolutions allow each node to aggregate information from higher-order neighbors. Backpropogation is performed similarly to a standard MLP.

B. Long Short Term Memory

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network (RNN). Just like CNN's and GNN's are designed for image and graph data, respectively, RNN's are designed for sequential data. In particular, they are often applied to time-series and text data.

Since we have time-series data on each node, we can use LSTM modules instead of fully-connected layers after performing graph convolutions. Each LSTM cell takes a vector of features for a single time step and outputs a vector of hidden states for that time step, which we can then pass into more LSTM cells or flatten and use fully-connected layers. Within each cell, the data are fed through "input", "output", and "forget" gates, and the sequential relationship is preserved across cells. One feature of LSTM's is Backpropogation Through Time (BPTT) which helps to update weights for the different gates of each cell.

C. Our Model

Our model has three main layers: a modified GCN layer, a standard GCN layer, and a final standard GCN layer. Note that each node is labeled with and input of 10 days of lag with the number of confirmed cases and deaths. The output labels are the confirmed cases and deaths for the 11th day.

The modified GCN layer performs the graph convolution on the input data, then passes the aggregated data into a stack of 3 LSTM's with 10 hidden states each. We use dropout with $p = 0.3$ for regularization in between each LSTM layer. The final LSTM layer outputs a sequence of 10 days of data with 10 hidden states per day. We flatten this into 100 features which becomes the new node label.

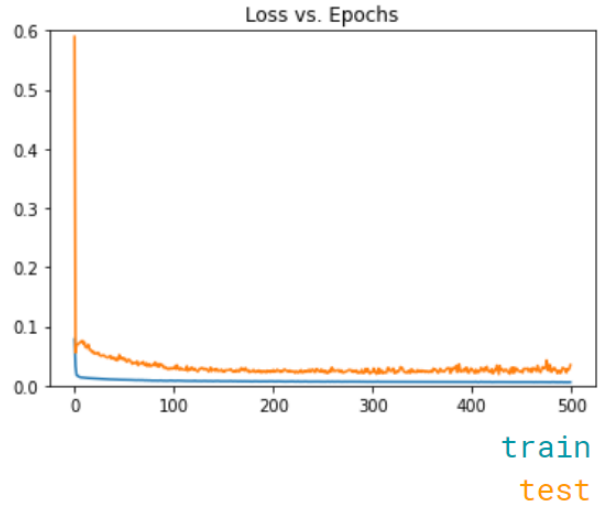
We passed the flattened data into a standard GCN layer (i.e. graph convolution followed by a linear layer) with 50 output features and an activation function of ReLU. These become the new node labels, which we pass into another standard GCN layer with 2 output features: the confirmed cases and deaths for the 11th day. The activation function into the output layer is sigmoid. We then evaluate loss using MSE and backpropgate error throughout the network, and we use the Adam optimizer with a learning rate of $\alpha = 0.001$.

For backpropogation, we update weights after each graph propagates through the network (this sounds like SGD). On the other hand, each graph has n_v nodes that we average error over before updating (this sounds like BGD). So in some sense, we have used both SGD and BGD.

One hyperparameter is the number of lags which determines the number of observations and number of features. We settled at 10 lag variables. We iterated a number of models, tuning number of stacked LSTM's, number of hidden states in LSTM layers, and number of output features in the standard GCN layers. We found the best performance with the model detailed above.

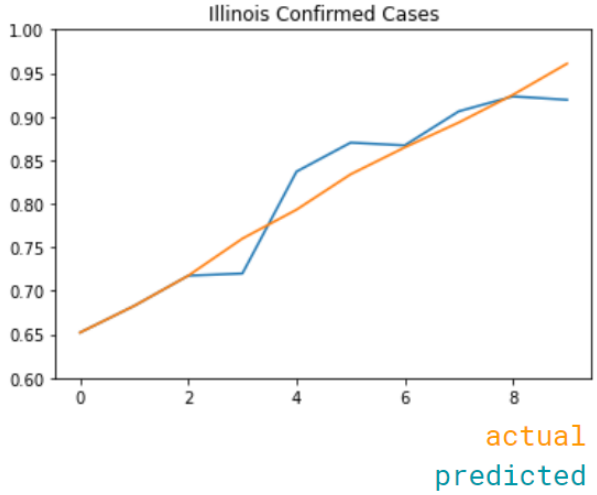
V. Results & Discussion

The training and testing MSE loss by epoch:



The January 22 to April 30 period data was used for training, and the May 1 to May 8 period data was used for testing. To forecast (predict) cases for a series of dates, the model's output is recursively produced then fed back into it.

We see that the model is capable of capturing general trends, but its results can be volatile compared to the ground truth, as seen in the plot below of actual and predicted cases in Illinois for the test period. In addition, the model sometimes struggles to extrapolate or generalize due to higher numbers of cases in testing data that the model was unfamiliar with.



VI. Conclusion

In conclusion, the model currently overemphasizes spatial relationship over temporal ones. To solve this, we could increase the number of lags and/or decrease the convolutions. Despite these weakness, our model is still able to make reasonable forecasts. Our final RMSE for the model with LSTMs is 0.1881198 which can be interpreted as .19 standard devaiations from the expected value which is not so bad given the lack of data and hyper-parameter tuning.

Our model would likely perform better with larger datasets that include recovery period, that are more granular by region, and are more accurate on a global scale. Further improvements include exploring additional variables such as air travel, population density, and level of quarantine control.

VII. Future Work

One of the largest constraints when it came to this project was a lack of time. Because of this, the future work can be broken up into three main sections:

A. Hyperparameter tuning

In total, we explored two GCNN "architectures". One which use a MLP to aggregate the values from the neighbours and another which utilizes an LSTM to further build upon the temporal aspect of our data. While non of these two architectures were tuned in regards to their hyperparameters (number of nodes per layer, number of layers, dropout), the LSTM architecture performed much better than the MLP. Because of this, we believe that if we were tune the hyperparameters for the LSTM architecture, we would be able to see improvements in model performance.

B. Further expansion of external data

While in our initial proposal, we aimed to include a method of representing the connectivity between states and provinces, this was something that we ran out of time to explore as scraping the data and building initial models took much longer than anticipated. In our files, we've included code and data tables to scrape external data such as the number of international airports and population density of each state/province. We hoped to use the international airports count as a proxy for each states total airline travel; We also planned on using the population density as an additional node label as we felt that would be an important statistic. Apart from this, we believe that there are other values which would be used as external data that should also be explored.

C. Generalization of Location

Lastly, we would want to expand the work that we've done to more than just the US and Canada. The reason why we initially limited our dataset to these two countries is because of their reasonable breakdown in regards to states and provinces. We were unable to come up with a methodology to generate the graph structure of each country and had to resort to hard-coding all of the edges and labels. If a better method is developed, we would be able to expand this to a world-wide scale which would in turn lead to the external data being more meaningful.

VIII. Contributions

Gary worked on the data cleaning, manipulation, and preprocessing steps. Philip worked on how to apply our neural network architecture to our project. Samuel, Jonathan, and Kara researched related works to pick out what was applicable to our project. Jonathan and Kara searched for external data sources and scraped Wikipedia for airport and population density data. Additionally, Samuel looked into node embedding.

Gary Braznichenko: 16.75, Philip Dohm: 33, Samuel Guo: 16.75, Jonathan Lu: 16.75, Kara Wong: 16.75.

REFERENCES

[1] D. Adam, "Modelers Struggle to Predict the Future of the COVID-19 Pandemic," The Scientist Magazine, March 2020.

[2] D. Zhu and Y. Liu, "Modelling irregular spatial patterns using Graph Convolutional Neural Networks," 2018.

[3] J. Zhou, "Graph Neural Networks: A Review of Methods and Applications," 2019.

[4] Z. Wu, "A Comprehensive Survey on Graph Neural Networks," 2019.

[5] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," 2016.

[6] R. Dandekar and G. Barbastathis, "Quantifying the effect of quarantine control in Covid-19 infectious spread using machine learning," 2020.

[7] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. Presented at the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018.

[8] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," IEEE Transactions on Intelligent Transportation Systems, 2019.

[9] M. Akhtar, M.U.G. Kraemer, and L.M. Gardner, "A dynamic neural network model for predicting risk of Zika in real time," BMC Med, 2019.