COVID Forecaster: Interactive Visualizations of COVID Forecasts with Deep Learning

Andrew Wang
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA
andreww7@illinois.edu

Junyi Gao
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA
junyig5@illinois.edu

Jimeng Sun
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA
jimeng@illinois.edu

ABSTRACT

Prediction of the rate at which COVID-19 spreads has a critical impact on public health policy. Subsequently, many scientists have developed deep learning and epidemiology models to forecast future numbers of COVID-19 cases. However, it can be difficult to extrapolate trends and compare predictions made by different models. Furthermore, some models that were created previously may rely on the number of active COVID-19 cases which is no longer reported. Therefore, we propose a framework we call the COVID Forecaster framework for predicting and visualizing future numbers of COVID-19 cases which includes a back end deep learning prediction model (the COVID Forecaster Model), and a front end visualization (the COVID Forecaster UI (User Interface)). The COVID Forecaster Model was applied to two tasks. In both tasks, the same general architecture with GConvGRU (Graph Convolutional Gated Recurrent Unit) layers, skip connections, ELU (Exponential Linear Unit) activations, and linear layers was used. In the first task, we compared our model to existing work and show our model has up to a 31.44 percent decrease in mean squared error, and a 97.48 percent decrease in runtime. In the second task, we show our model can help public health officials continue making COVID forecasts on recent data where the number of active cases is no longer reported, and show our model has up to a 99.77 percent decrease in mean squared error in comparison to a standard GCN (Graph Convolutional Network). The COVID Forecaster UI allows users to extrapolate trends from visualizing and comparing COVID forecasts between different models. This UI was made with Dash, Plotly, Heroku, Crontab, and Bash.

CCS CONCEPTS

• **Human-centered computing** → *Visualization toolkits*; • **Applied computing** → *Health informatics.*

KEYWORDS

COVID Forecasting, Deep Learning, Interactive Visualizations

KDD-UC, Washington, D.C.,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As COVID-19 has spread since December 2019, it has become increasingly important to be able to accurately forecast future COVID-19 case numbers to be able to make judicious decisions when it comes to determining public health policies.

Models that employ traditional epidemiology, deep learning, and a mixture of both have been developed to forecast future numbers of COVID-19 cases. One such model is STAN, which stands for spatio-temporal attention network [7].

One problem that has arisen with the creation of these pandemic forecasting models (such as STAN) is that they may rely on the number of active COVID-19 cases from the JHU COVID-19 dataset if they use the SIR epidemiology model. This data was reported in the JHU dataset early on during the pandemic; however, recently JHU has stopped reporting this data [3, 4]. This is likely because the JHU COVID-19 dataset relies on the COVID Tracking Project (CTP) to report the number of recovered cases to determine the number of active cases, and the CTP has stopped reporting the number of recovered cases because: 1) many states have stopped reporting the number of recovered cases resulting in an undercount for the total number of national recovered cases, 2) there is no standardized way to measure the number of recovered cases at a state level, and 3) different states use different definitions of when someone is recovered [12].

Consequently, it may be difficult for models that rely on the use of the number of infected (active) and recovered cases from the JHU COVID-19 dataset to directly be used with recent date ranges in which SIR data is missing, or generalize and perform well in the absence of this data (due to their dependence on SIR data). Therefore, it would be beneficial if a model that doesn't rely on the use of data that is no longer reported was created so scientists can continue to make forecasts of COVID-19 numbers. Ideally, this model would be 1) able to learn how geography impacts the spread of disease by using a graph neural network, 2) able to understand how time impacts the spread of disease by using a time-series model such as an RNN, LSTM, or a GRU, 3) able to understand patterns in the spread of disease by leveraging convolutional filters, 4) easily adaptable for use with recent data (for predicting cumulative future cases) and older data (for comparison to models dependent on active number of COVID-19 cases), and 5) comparable in performance to previous models.

Another problem that has arisen with the creation of pandemic forecasting models is that the output of some models may be in the form of a CSV file or table with dates, location names, and predicted future numbers of COVID-19 cases for each date-location combination. It may be difficult to interpret or extrapolate trends from

a table or CSV file. An interactive visualization of the output may be easier and more intuitive to understand than looking through numbers in a table or file.

To summarize, problems that have come up with the creation of pandemic forecasting models include:

(1) **Reliance on data that has recently stopped being reported.**
(2) **Interpreting and extracting trends from model predictions.**

To address these challenges, we propose the COVID Forecaster framework which has the following contributions:

(1) **COVID Forecaster Model.** This work attempts to provide a solution to the reliance of some models on unreported data by proposing a model we call the COVID Forecaster model. The COVID Forecaster model uses graph convolutional GRU layers to capture temporal and geographic features of pandemic data, along with skip connections to ensure outputs are scaled accurately, and linear layers to ensure tensor shapes match when added during forward propagation [13, 14]. The COVID Forecaster model was applied to two different tasks. The first task was to compare our model to existing work. In this case, our model was trained on older data when the active number of active COVID-19 cases was still reported, our model predicts the number of future active cases, and our model shows up to a 31.44 percent decrease in mean squared error and a 97.48 percent decrease in runtime. The second task was to allow public health officials to continue making COVID forecasts on recent data where the number of active cases is not reported. In this case, our model was trained on more recent data when the number of active cases is unreported, our model predicts the number of cumulative future cases, and our model shows up to a 99.77 percent decrease in mean squared error.
(2) **COVID Forecaster UI.** This work also attempts to solve the problem of improving interpretability of pandemic forecasting models by creating an interactive interface we call the COVID Forecaster UI that visualizes both stored and uploaded predictions for COVID-19 case numbers across the United States. Our UI can be found here: https://covid-forecaster-ui.herokuapp.com/

2 PREVIOUS WORK

2.1 COVID Deep Learning Models

STAN is a hybrid deep learning model that the COVID Forecaster model architecture was inspired by. STAN combines a graph attention network (GAT) with a gated recurrent unit (GRU) and the SIR (Suspected-Infected-Recovered) epidemiology model [1, 15, 17]. The GAT captures the geographic impact of disease transmission, the GRU captures temporal disease patterns, and the SIR model captures disease transmission dynamics. The JHU COVID-19 dataset as well as geographic data such as the latitude, longitude, and population of each location are used as the model's input [5]. The predicted number of future active and recovered COVID-19 cases are STAN's output. While STAN is a state of the art model that performs well when trained on older data where the number of active cases is reported, STAN has difficulty performing well on newer data where the number of active cases has stopped being reported.

2.2 COVID Visualization

Well known COVID visualizations include the JHU COVID dashboard and the NY Times COVID dashboard [10, 16]. While these interactive dashboards visualize real time data, they suffer from the following problems: 1) they only visualize historical data (as opposed to future predictions), 2) they visualize data from one source (either the NY Times or the JHU COVID dataset), 3) they aren't extensible as they don't allow the user to visualize their own custom dataset, and 4) they don't allow for comparisons between different models and data sources. Other work on COVID visualization includes that of Wissel et al., as well as that of Comba et al.; however, they also only visualize historical data from a set data source and aren't directly extensible to visualizing data that is uploaded by the user [2, 18]. Our work aims to create an extensible dashboard that solves these problems by visualizing predictions (as opposed to historical data) from files uploaded by the user. This allows the user to compare and contrast predictions from different models (as opposed to a single model) and data sources, thereby creating a more well-rounded view on future case number predictions.

3 PROPOSED WORK

Table 1: Notation used to describe model

Variable	Meaning
G	Graph given as input to model
V	Vertices/Nodes (US states) in model
E	Edges between nodes in model
X	Input data (for all US states) to model
x_i	Input data/Feature vector (for a single US state) to model
H	Number of days of historical data used for each prediction
F	Number of different types of features in input data
D	Number of days into future for which predictions are made
Y	Model output
$h_{v_i}^{(l+1)}$	Feature vector of node v_i in layer l+1
σ	Activation function used
GRU	A pass through a GRU cell
j	Indices of neighboring nodes of v_i
c_{ij}	Normalization constant for edge (v_i, v_j)
$h_{v_j}^{(l)}$	Feature vector of neighboring node j in layer l
W(l)	Weight matrix of layer l for GConvGRU
z	Input to ELU activation function
a	Constant defining smoothness of function when inputs are negative

3.1 COVID Forecaster Model

To aid in the prediction of future COVID-19 case numbers and to remove the reliance on unreported SIR data, we propose a deep learning model we call the COVID Forecaster Model. The COVID Forecaster Model uses graph convolutional GRU (GConvGRU) layers to simultaneously learn the impact of time on the spread of disease from using convolutions and GRU cells, while also learning the impact of location on the spread of disease from using a graph neural network. Furthermore, linear layers and skip connections are used to scale the output correctly.

3.1.1 Model inputs and output. The input to the COVID Forecaster model should comprise of a graph G, and a matrix X (recent historical COVID-19 data). The graph G(V, E) is used to construct the graph neural network and is comprised of nodes/vertices (V) which represent US states/regions, as well as edges (E) which convey which states are adjacent and allow for message passing/aggregation in graph neural networks. Furthermore, the historical input data (X = $[x_1, x_2, ..., x_{|V|}]$) should be a matrix of dimension |V| by H*F where H is the number of days of past data that are used, and F is the number of different features that are used. Additionally, $x_i = [f_{1,1}, f_{1,2}, ..., f_{1,H}, f_{2,1}, ..., f_{F,H}]$ is the feature vector for state i where $f_{j,k}$ represents the value for feature j on day k of the historical data for state i.

The output of the COVID Forecaster model should be a matrix Y of dimension |V| by D (# days of into the future for which predictions are made). Therefore, Y[i, j] would be the predicted value for state i, at time j+1 days into the future.

3.1.2 Graph Construction. In the COVID Forecaster Model, we use a graph neural network to capture disease transmission dynamics from location in the United States. This requires the use of a graph G(V, E) which is defined by its nodes (V) and edges (E).

Each node (V) in our graph represents a US state. In the COVID Forecaster Model, there are 52 nodes in the graph we use.

Each node (representing a state) is connected by edges to all of its adjacent states in the geographic United States. If a node does not have any adjacent states in the United States, then it is not connected to any other nodes in the graph. Each edge (E) connecting two nodes in our graph has weight 1 and is neither weighted differently due to proximity between nodes, nor weighted differently due to differences or similarities in population size.

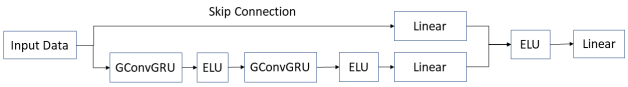

Figure 1: COVID Forecaster Model Architecture

3.1.3 Architecture. In our COVID Forecaster model, the input data is passed through multiple graph convolutional GRU (GConvGRU) layers (each of which can be thought of as a GRU unit stacked upon a graph convolution for each node in the graph), exponential linear unit (ELU) activations, and linear layers.

Of importance, we use skip connections in our architecture and show that this results in better performance in our experiments. Each time our model makes an inference, the previous H days of data for various features are used as input data, and predictions of the output feature (# active cases, or # cumulative cases) for the next D days are made. If the output feature is the number of active cases, then the input data will include the past H days of the number of active cases. If the output feature is the number of cumulative cases, then the input data will include the past H days of cumulative cases. Since the next D days of active cases (or cumulative cases) will likely differ by a small perturbation from the previous D days of active cases (or cumulative cases), we hypothesize that by allowing this data to bypass several graph convolutions and ELU activations via a skip connection, the network has a smaller number (perturbation) to learn which allows for faster training and more accurate inference. However, since H may not always equal D, we use linear layers to change the shape of the tensors to the desired shape before the previous H days of the output feature is added to the predicted perturbation from the output feature (where the perturbation is the output of the 2 GConvGRU and ELU passes).

Below, we formalize a pass through a graph convolutional GRU layer and activation function, where:

$$h_{v_i}^{(l+1)} = \sigma(GRU(\sum_j \frac{1}{c_{ij}} h_{v_j}^{(l)} W^{(l)}))$$

Below, we formalize a pass through the exponential linear unit (ELU) activation function, where:

$$\sigma = ELU(z, a) = \begin{cases} z & z > 0 \\ a * (e^z - 1) & z \leq 0 \end{cases}$$

Below, we will use X to denote the input data to the COVID Forecaster Model. X is a matrix with dimension |V| by H*F, where |V| is the number of locations used, H is the number of days of historical data used to make each prediction, and F is the number of different features used. In our case, |V| is 52 since we model 52 US states/regions, H is 6 because we use the 6 previous days of data to make each inference, and F is either 4 or 6 depending on what is being predicted. F is 4 when the future number of confirmed cases is being predicted (and therefore the past number of confirmed cases, past number of deaths, past daily change in confirmed cases, and past daily change in deaths are used). F is 6 when the future number of active cases is being predicted (and therefore the past number of active cases and past daily change in number of active cases are used in addition to the previously mentioned 4 features, yielding a total of 6 different features).

To summarize, in the COVID Forecaster Model, the input X is first passed through a linear layer. Below, X[: , 0 : H] represents the past H days of active (or confirmed, depending on what is being predicted) case numbers for every US state.

$$h1 = Linear(X[:, 0 : H])$$

The input X is concurrently passed through two graph convolutional GRU layers followed by ELU activations, and a linear layer.

$$h2 = ELU(GConvGRU(X, G))$$

$$h3 = ELU(GConvGRU(h2, G))$$

$$h4 = Linear(h3)$$

Then, the output of the skip connection and the output from the graph convolutional GRU layers are added together and passed through a final ELU activation and linear layer to give the final output Y of dimension |V| by D (# days of future predictions).

$$h5 = ELU(h1 + h4)$$

$$Y = Linear(h5)$$

3.1.4 Comparison between COVID Forecaster Model and Existing Work. STAN and the COVID Forecaster model are similar in that they both use a graph neural network, and they both use the JHU COVID-19 dataset. STAN and the COVID Forecaster model also both use exponential linear units (ELU's) as activation functions, and they both use the mean squared error (MSE) function for loss calculations.

However, STAN and the COVID Forecaster model are different in that the COVID Forecaster model does not use the SIR model, the COVID Forecaster model uses skip connections, and the COVID Forecaster model uses Seo et. al's Chebyshev Graph Convolutional Gated Recurrent Unit Cells (whereas STAN uses graph attention layers that are combined with a GRU cell) [8]. Furthermore, the COVID Forecaster model constructs the graph that is passed to the graph neural network by connecting every state (represented as a node in the graph) to each of its geographically neighboring states, whereas STAN does not necessarily connect each state to all of its geographically adjacent neighboring states. Instead, STAN constructs the graph passed to the GNN by calculating the similarity between every pair of US states based on location and population size, and then connecting each state to a select number of states that are most similar. In addition, COVID Forecaster trains one model that can make predictions for all states in the United States, whereas STAN trains a separate model for each US state. COVID Forecaster also uses smoothing, whereas STAN uses normalization. Lastly, COVID Forecaster was created with PyTorch Geometric due to its implementation of graph convolutional gated recurrent unit layers, whereas STAN was created with Deep Graph Library.

3.2 COVID Forecaster Interactive UI

To aid in efforts towards increasing interpretability of future COVID case number forecasts and making trend extrapolation easier, we created an interactive UI.

Our UI has a line graph showing the number of predicted COVID-19 cases for the near future for each state and allows the user to:

- See how COVID-19 cases are slated to change in the near future for each state
- Interact with the UI by zooming in/out and hovering over the graph and seeing relevant information pop up

Since disease trends tend to be similar in locations that are close to each other, our UI also has a choropleth map (a map where different geographical locations are divided and the color, shading, or pattern of each particular region corresponds to the intensity or predicted number of COVID-19 cases). Our choropleth map allows the user to:

- Interact with the map by using a slider (representing the number of days into the future) to see how the intensity of COVID-19 can change over time across states in the US
- Interact with the map by hovering their mouse over each state
- Automatically update the state line graph displayed to correspond to the state the user's mouse is hovering over in the US choropleth map so the user can compare and contrast trends between different US states

While a UI displaying COVID-19 prediction trends for our COVID-19 Forecaster model is useful, our UI also allows the user to upload, visualize, and compare COVID-19 predictions from other models so 1) the user has a more well-rounded view of predictions, and 2) the functionality of the UI can be extended to other models.

3.2.1 UI Layout. In the top half of the COVID Forecaster UI, the user can view our model's predictions for the future number of COVID-19 cases. This is shown through a choropleth map (US map) on the top left half of the page and a line graph (state line graph) on the top right half of the page.

In the middle of the COVID Forecaster UI, the user can upload a CSV file with their model's predictions for comparison to our model's predictions.

Subsequently, in the bottom half of the UI, the user's uploaded predictions are visualized similarly to how predictions are visualized in the top half of the UI - with a choropleth map on the left and a line graph on the right.

3.2.2 Technology. Our UI was created in Dash – a Python framework written on top of Flask, Plotly.js, and React.js for building interactive web applications [6]. We chose to use Dash because it abstracts many technologies and protocols that would traditionally be needed to build full stack web applications which allowed us to make progress more quickly and iterate over different versions of our UI. Another reason we chose to use Dash over other Python web frameworks such as Flask and Django is because it was easier to customize and create interactive visualizations (as opposed to static visualizations) because Dash provides callback functions which allow the web app to dynamically respond to the user's actions while using the web app. To add on, Dash also provides detailed documentation with examples of how to use it. Furthermore, another reason we chose to use Dash in particular was because it was a Python based framework and we wanted it to be able to more easily interface with deep learning models/frameworks since many machine learning/deep learning frameworks (Ex: Deep Graph Library (DGL), PyTorch/PyTorch Geometric, and TensorFlow) are Python based.

Heroku is a cloud platform that manages hardware, servers, and the infrastructure needed for web apps [9]. We chose to use Heroku for our UI because it takes care of both hosting and deployment.

To update the UI with new data, we can use a Bash script. Bash is the shell for the GNU operating system, and a bash script is a text file containing a series of commands that we would normally type into the command line to execute. We can use a bash script to automate running these commands sequentially. In the bash script, we can activate a Python virtual environment, run the training pipeline (which should pull any new data, inference on the new data, and write the predictions to a CSV file), and re-deploy the Dash web app with the updated CSV file to Heroku, thereby updating the interface.

Since we would like to update the UI automatically and regularly, crontab in Linux can be used to run the bash script at pre-defined intervals [8]. Crontab stands for cron table and is a job scheduling system that is built into Linux. A crontab entry can be created to periodically run the bash script previously discussed to update the UI automatically and regularly, therefore removing the need to manually re-run the bash script to update the UI.

4 RESULTS AND DISCUSSION

We created a deep learning model to allow scientists to continue making COVID forecasts in the absence of data previously used by older models. We also created a UI to increase the interpretability of COVID forecasts by other models.

4.1 COVID Forecaster Model Experiments

In this work, we applied the same general COVID Forecaster model architecture with the same source of data (the JHU COVID-19 dataset) to two different experiments. In both experiments, our model uses a graph neural network to model the impact of geography on the spread of disease, a GRU to allow the model to learn how time impacts the spread of disease, skip connections to scale the output correctly, and linear layers to ensure tensors from previous layers can have the desired shape when they are added to future layers in skip connections. In our experiments, we chose to use mean squared error (MSE), mean absolute error (MAE), and training time to quantify performance.

In one experiment, we used older JHU data (May 1st, 2020 to December 1st, 2020) when the number of active cases was still reported, and the model output was the number of future active cases. In this case, the input features used were the number of confirmed cases in the past, the number of deaths in the past, the daily change in the number of confirmed cases in the past, the daily change in the number of deaths in the past, in addition to the number of active cases in the past and the daily change in the number of active cases in the past. In this experiment, we show how the performance of the COVID Forecaster model compares to previous work (STAN).

In another experiment, we used recent JHU data (May 1st, 2020 to March 7th, 2022), and the model output was the future number of cumulative infected (confirmed) cases. In this case, the input features were the number of confirmed cases in the past, the number of deaths in the past, the daily change in the number of confirmed cases in the past, and the daily change in the number of deaths in the past. In this experiment, we show that predictions with recent JHU COVID-19 data (where the number of active cases is no longer reported) can be made.

Table 2: Preprocessing of model inputs and ground truth labels

Data Category	Model Input	Ground truth
Training	Smoothed	Smoothed
Validation	Smoothed	Unsmoothed
Testing	Smoothed	Unsmoothed

4.1.1 Experimental Design: Data Preprocessing. The input to the COVID Forecaster Model is always smoothed because smoothing the model input doesn't require past information about the data, and because we would like to avoid irregularities in the data (such as how frequently the data is reported) from affecting how the model calculates its output.

However, the COVID Forecaster Model does not always use smoothed data as the ground truth. Smoothed data is only used as the ground truth for the training split of the data because we would like to avoid irregularities in the data from affecting how the weights for the model are updated in the loss calculation and subsequent backpropagation, and because we don't want the model to learn the irregularities present in real world data. In contrast, unsmoothed data is used as the ground truth for validation and testing because we would like to see how well the model performs in the real world where the data is not always smoothed, and because the validation and testing ground truth values aren't used in backpropagation calculations so keeping the validation and testing dataset's ground truth labels unsmoothed won't affect future predictions by the model.

To smooth numbers, the COVID Forecaster Model uses the convolve function provided by the NumPy (Numerical Python) library [11].

4.1.2 Predicting number of future active cases. In one experiment, we compare the COVID Forecaster Model to the STAN model (baseline). In this case, older data when the number of active COVID cases was reported was used, and the number of future active cases was the output of both models. In this comparison, each model was trained for 50 epochs and 1000 epochs using the same source of data and the same date range (May 1st, 2020 to December 1st, 2020). The same testing dataset was used, and no GPU was used.

Table 3: Training time, MSE, and MAE for STAN and COVID Forecaster

Model	Epochs	Sec./epoch	MSE	MAE
STAN	50	31.19	4.22×10^8	10121.36
STAN	1000	16.25	4.04×10^8	9545.22
COVID Forecaster	50	0.54	3.67×10^8	9820.94
COVID Forecaster	1000	0.41	2.77×10^8	6635.76

The COVID Forecaster model shows a 98.26 percent decrease in training time compared to STAN when both models are trained for 50 epochs, and a 97.48 percent decrease in training time compared to STAN when both models are trained for 1000 epochs. We hypothesize that the large percent decrease in training time for the COVID Forecaster model is because STAN trains a separate model for each US state, whereas COVID Forecaster uses the same model to train across all US states.

Our COVID Forecaster model also shows a 13.07 percent decrease in mean squared error in comparison to STAN when both models are trained for 50 epochs. Furthermore, the COVID Forecaster model also shows a 31.44 percent decrease in mean squared error in comparison to STAN when both models are trained for 1000 epochs. To provide a fair comparison we also compared the percent change in mean squared error across the best case for each model and see again that the COVID Forecaster model has a 31.44 percent decrease in mean squared error in comparison to STAN.

We also present the mean absolute error in each experiment to provide a better picture of the differences between each model. This COVID Forecaster model shows a 2.97 percent decrease in mean absolute error in comparison to STAN when both models are trained for 50 epochs. Furthermore, the COVID Forecaster model

also shows a 30.48 percent decrease in mean absolute error in comparison to STAN when both models are trained for 1000 epochs. To provide a fair comparison we compared the mean absolute error across the best case for each model and see again that this COVID Forecaster model has a 30.48 percent decrease in mean absolute error in comparison to STAN.

4.1.3 Predicting number of cumulative future cases. A second experiment was done to show that our model can help public health officials continue to make COVID Forecasts in the absence of the number of active cases. In this case, a date range with more recent data (where the number of active cases is no longer reported) was used, and the future number of cumulative cases was the output of the model.

In this experiment, we compared the COVID Forecaster Model (with skip connections) to the COVID Forecaster Model without skip connections, as well as to a standard 2 layer graph convolutional network (GCN). We did this to show how skip connections and using a GCN with a GRU is better than using a standard GCN. In each comparison, data from May 1st, 2020 to March 7th, 2022 was used, and each model was trained for 1000 epochs over the same dataset with the same training, validation, and testing split.

In this experiment, we didn't compare the COVID Forecaster Model to STAN because recent data was used and STAN relies on the use of data that has stopped being reported as of December 15th, 2020 [3].

Table 4: MSE and MAE of different architectures for comparison

Model	MSE	MAE
COVID Forecaster	1.1942 x 10^{10}	7.0671 x 10^{4}
COVID Forecaster w/o skip connection	2.4314 x 10^{12}	9.7871 x 10^{5}
Standard GCN	5.1834 x 10^{12}	1.5120 x 10^{6}

The COVID Forecaster model (with a skip connection) shows a 99.77 percent decrease in mean squared error in comparison to a standard GCN, as well as an 95.33 percent decrease in mean absolute error in comparison to a standard GCN.

4.2 COVID Forecaster UI

Our UI is accessible at: https://covid-forecaster-ui.herokuapp.com/

When the user loads the COVID Forecaster User Interface in a web browser, the user can see our model's predictions in the top half of the page.

The bottom half of the page will initially be empty but will look similarly to the top half of the page once populated. The bottom half of the page can be populated by clicking on the "Select files" prompt in the middle of the page and uploading the user's model predictions in the form of a CSV file.

In both the top and bottom half of the page, a US choropleth map and a line graph are displayed side by side. The user can click and drag on the slider under the choropleth map to see how the intensity (represented by color) of COVID-19 is predicted to change over time across the US. When the user hovers their mouse over a state in a choropleth map, the line graph displaying the predicted number of

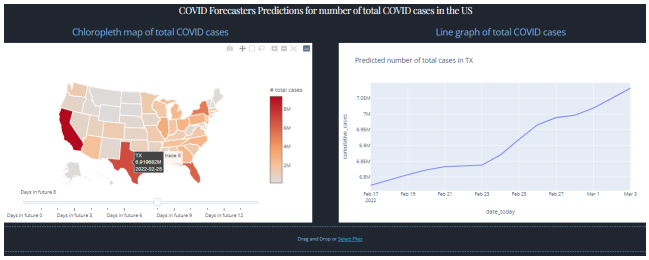

Figure 2: Top half of COVID Forecaster UI

COVID cases to the right of the choropleth map is automatically updated to correspond to the state that the user's mouse is hovering over. Furthermore, when the user's mouse hovers over a state in the choropleth map or a point in the line graph, the predicted number of COVID cases for the currently selected future date will appear next to the user's mouse.

5 FUTURE WORK

Future work for the COVID Forecaster model could involve integrating vaccination data into the model, as well as looking for other datasets that report the number of active infections and merging them with the JHU dataset so models that rely on the SIR model can have a complete set of SIR data to use for training. Future work on the COVID Forecaster UI could involve generalizing the choropleth map to different locations (such as those outside of the US), as well as creating greater color variation from plotting the change in the number of new cases (as opposed to total cumulative cases), and plotting the normalized (instead of raw) number of cases for each US state.

6 CONCLUSION

In this work, we present a deep learning model that can predict future cumulative COVID case numbers, and we also present an interface that allows users to visualize and compare COVID forecasts between different models. To implement the deep learning model, we used PyTorch Geometric's GConvGRU implementation, linear layers, skip connections, and ELU activations. To implement the interface, we used Dash and Plotly as the framework, CSV files to store the data, Heroku for hosting the UI, and crontab in Linux with a Bash script to regularly update the interface after retraining our model with new data. We hope our model and our interface can help government and public health officials make more well informed decisions to control the COVID-19 pandemic.

REFERENCES

[1] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. https://doi.org/10.48550/ARXIV.1409.1259

[2] João L. D. Comba. 2020. Data Visualization for the Understanding of COVID-19. *Computing in Science Engineering* 22, 6 (2020), 81–86. https://doi.org/10.1109/MCSE.2020.3019834

[3] CSSEGISandData. 2020. *US Recoveries have been nullified #3464.* Retrieved May 19, 2022 from https://github.com/CSSEGISandData/COVID-19/issues/3464

[4] CSSEGISandData. 2021. *Recovery data to be discountinued #4465.* Retrieved May 19, 2022 from https://github.com/CSSEGISandData/COVID-19/issues/4465

[5] CSSEGISandData. 2022. *COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University.* Retrieved May 20, 2022 from https://github.com/CSSEGISandData/COVID-19

[6] Dash. 2021. *Dash Python User Guide.* Retrieved May 20, 2022 from https://dash.plotly.com/

[7] Junyi Gao, Rakshith Sharma, Cheng Qian, Lucas M Glass, Jeffrey Spaeder, Justin Romberg, Jimeng Sun, and Cao Xiao. 2021. STAN: spatio-temporal attention network for pandemic prediction using real-world evidence. *Journal of the American Medical Informatics Association* 28, 4 (01 2021), 733–743. https://doi.org/10.1093/jamia/ocaa322 arXiv:https://academic.oup.com/jamia/article-pdf/28/4/733/36642145/ocaa322.pdf

[8] GeeksforGeeks. 2022. *'crontab' in Linux with Examples.* Retrieved May 20, 2022 from https://www.geeksforgeeks.org/crontab-in-linux-with-examples/

[9] Heroku. 2022. *Cloud Application Platform | Heroku.* Retrieved May 20, 2022 from https://www.heroku.com/

[10] JHU. 2020. *COVID-19 Map - Johns Hopkins Coronavirus Resource Center.* Retrieved May 23, 2022 from https://coronavirus.jhu.edu/map.html

[11] NumPy. 2022. *numpy.convolve - NumPy v1.22 Manual.* Retrieved May 20, 2022 from https://numpy.org/doc/stable/reference/generated/numpy.convolve.html

[12] The COVID Tracking Project. [n.d.]. *Data FAQ | The COVID Tracking Project.* Retrieved May 19, 2022 from https://covidtracking.com/about-data/faq

[13] Benedek Rozemberczki. 2022. *PyTorch Geometric Temporal.* Retrieved May 20, 2022 from https://pytorch-geometric-temporal.readthedocs.io/en/latest/modules/root.html

[14] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2016. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. https://doi.org/10.48550/ARXIV.1612.07659

[15] David Smith and Lang Moore. 2004. *The SIR Model for Spread of Disease - The Differential Equation Model.* Retrieved May 20, 2022 from https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model

[16] NY Times. 2021. *Covid-19 World Map: Cases, Deaths and Global Trends - The New York Times.* Retrieved May 23, 2022 from https://www.nytimes.com/interactive/2021/us/covid-cases.html

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. https://doi.org/10.48550/ARXIV.1710.10903

[18] Benjamin D Wissel, P J Van Camp, Michal Kouril, Chad Weis, Tracy A Glauser, Peter S White, Isaac S Kohane, and Judith W Dexheimer. 2020. An interactive online dashboard for tracking COVID-19 in U.S. counties, cities, and states in real time. *Journal of the American Medical Informatics Association* 27, 7 (06 2020), 1121–1125. https://doi.org/10.1093/jamia/ocaa071 arXiv:https://academic.oup.com/jamia/article-pdf/27/7/1121/34153390/ocaa071.pdf

Our code is accessible at: https://github.com/AWUIUC/CS499-SP22

A COVID FORECASTER UI

A.1 Running a local instance of the COVID Forecaster UI

To run the UI locally, please:

(1) Install Python, and create/activate a Python virtual environment.
(2) Clone the repo linked above and navigate to the "CS499-SP22/5. UI Code/" directory
(3) Install the necessary Python packages needed by running the command: "pip install -r requirements.txt"
(4) Activate and run the UI by running "python app.py" and then clicking on the link that shows up after successfully creating an instance of the web app/UI.

A.2 Expected format of uploaded files in COVID Forecaster UI

Given a CSV file that stores predictions for future COVID numbers, the COVID Forecaster UI visualizes the stored predictions at the top of the interface's web page. The COVID Forecaster UI also allows the user to upload a CSV file with predictions of future numbers of COVID cases from other deep learning and epidemiology models for visualization at the bottom of the interface's web page and for comparison to our model's predictions.

In both cases, the CSV file should have 3 columns: date_today, state, and cumulative_cases (or active_cases, depending on what is being visualized). The values in the date_today column should be of the form Year-Month-Day (Ex: 2022-02-15) and represent the day into the future for which the corresponding prediction is made. The values in the state column should be the abbreviation of a state name (Ex: IL). Lastly, the values in the cumulative_cases (or active_cases) column should be either floats or integers.

B COVID FORECASTER MODEL

As part of an effort to create reproducible results, the PyTorch random number generator seed was set to 0 in all of the experiments run and reported in this work. The data used can be found at: https://github.com/CSSEGISandData/COVID-19

B.1 Reproducing results in Table 2 (Comparing COVID Forecaster to STAN for 50 and 1000 epochs)

To reproduce the results in Table 2, please:

(1) Clone the repo to a Google Drive folder by:
  - Navigate to drive.google.com in your web browser
  - Click on the "New" button in the top left corner of the screen
  - Click on the "More" button
  - Click on "Google Colaboratory"
  - In a cell in the colab session, run the following commands to mount the colab session to your Google Drive and clone the repo:
    (a) from google.colab import drive
    (b) drive.mount('/content/gdrive')
    (c) %cd "/content/gdrive/My Drive/PATH-TO-DIRECTORY"
    (d) !git clone https://github.com/AWUIUC/CS499-SP22.git
(2) Return to drive.google.com, open each of the following files as a Google Colab session by double clicking on them. Then, run all the cells in each Colab session:
  - STAN for 50 Epochs: CS499-SP22/3. Experiments /1. STAN-Original/Experiment2-50Epochs-Colab /train_stan_old_data.ipynb
  - COVID Forecaster for 50 Epochs: CS499-SP22 /3. Experiments/2. COVID_Forecaster /train_v3.1_old_data_timed_50_epochs.ipynb
  - COVID Forecaster for 1000 Epochs: CS499-SP22 /3. Experiments/2. COVID_Forecaster /train_v3.2_old_data_timed_1000_epochs.ipynb

To reproduce the results in Table 2, please also

(1) Clone the repo to a local or virtual machine since the free tier of Google Colab will terminate training sessions when too much time has been spent in the Colab session.
(2) Install Python
(3) Create and activate a Python virtual environment
(4) Install the necessary libraries
(5) Run the file at CS499-SP22/3. Experiments /1. STAN-Original/Experiment1-1000Epochs-Workstation /train_stan_old_data.py"

B.2 Reproducing results in Table 3 (Comparing COVID Forecaster with and without skip connections to a standard GCN)

To reproduce the results in Table 3, please follow the same directions for running Colab files in section B.1 but use the following files instead:

- COVID Forecaster with skip connections for 1000 Epochs: CS499-SP22/3. Experiments/2. COVID_Forecaster /train_v3.3_new_data_COVID_Forecaster_full.ipynb
- COVID Forecaster without skip connections for 1000 Epochs: CS499-SP22/3. Experiments/2. COVID_Forecaster /train_v3.4_new_data _COVID_Forecaster_no_skip_connections.ipynb
- Standard GCN for 1000 Epochs: CS499-SP22/3. Experiments/2. COVID_Forecaster /train_v3.5_new_data_standard_GCN.ipynb