Forecasting Covid19 County Level Infection Case and Death in The United States using Logistic Regression and Recurrent Neural Networks

Litian Liang, Yaosheng Xu, Yilong Huang

Donald Bren School of Information and Computer Science,

University of California, Irvine

1. Introduction and Problem Statement

Problem Statement:

COVID-19 is a respiratory disease caused by a type of coronavirus named SARS-Cov-2. It is highly infectious and has a much higher mortality rate than seasonal influenza. For many reasons like allocating resources and funding or planning social distancing measures, forecasting the severity of the infection becomes an important task. Our goal is to predict the case and death increase:

given:

1. Case and death increase ($y = [case, death]$) of the past N days $\{y_1, y_2, \ldots, y_t\}$
2. Condition of the county $x = [\#Hospital, \#ICU\ beds, \ldots]$

predict:

The case and death increase of tomorrow y_{t+1}

Summary of Results:

We used mainly 2 types of machine learning models: 1. Logistic Regression, 2. Deep Recurrent Neural Network. Logistic Regression is a model that is widely used in pandemic studies. It fits a logistic sigmoid curve to the total case and death. Deep Recurrent Neural Network is a type of Artificial Neural Network that is popular in the area of AI and Deep Learning that performs well on predicting sequential data like forecasting weather conditions. The performance of the models we have tested are recorded in section 6 table 2.

2. Related Work:

Yu-Group's ensemble model is one of the models that achieve state-of-the-art performance. Their model combines a county-specific exponential growth model and a shared exponential growth model through a weighted average. We evaluated this model on our test dataset. The 1-day look ahead case MSE error is recorded.

3. Data Sets

Covid19 county-level severity data[1] is collected from various institutions. Yu-Group's Covid19 severity Forecasting public repository includes these data. All data used in this project is from Yu-Group's Covid Severity Forecasting repository.

Data Overview:

1. Data shape:

   683 Columns: County condition (including county feature, case, and death of each day)

   3142 Rows: Detailed data of each county in The United States

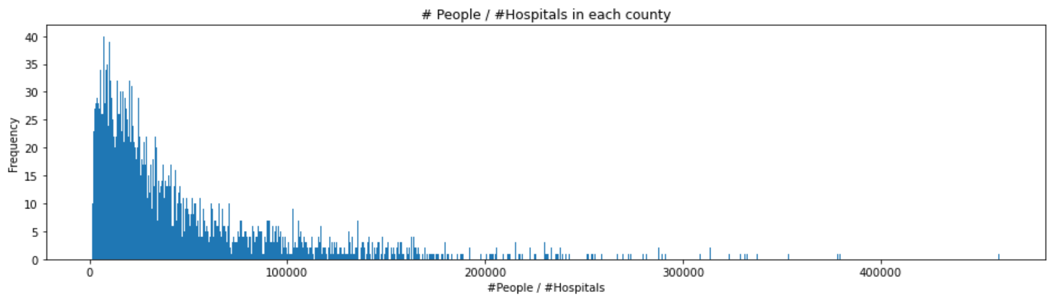
   The data of Orange County, California (not shown in full):

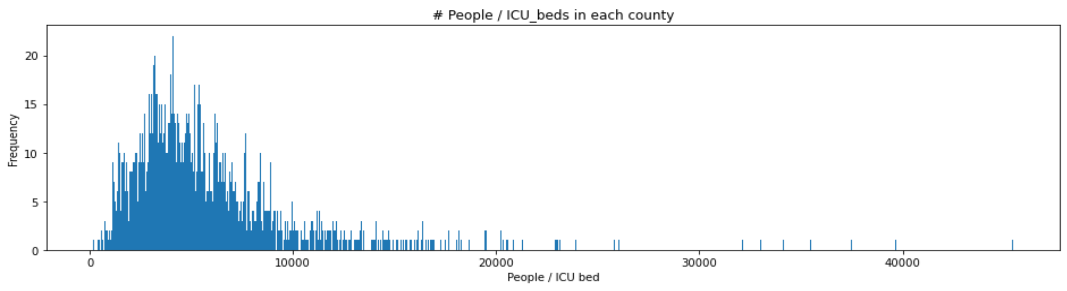
CountyName	StateName	#Cases_01-22-2020	#Cases_03-22-2020	…	#Cases_11-04-2020	#Hospitals	#ICU_beds
Orange	CA	0	95	…	60841	24.0	651.0

Table 1

2. Data Visualizations:

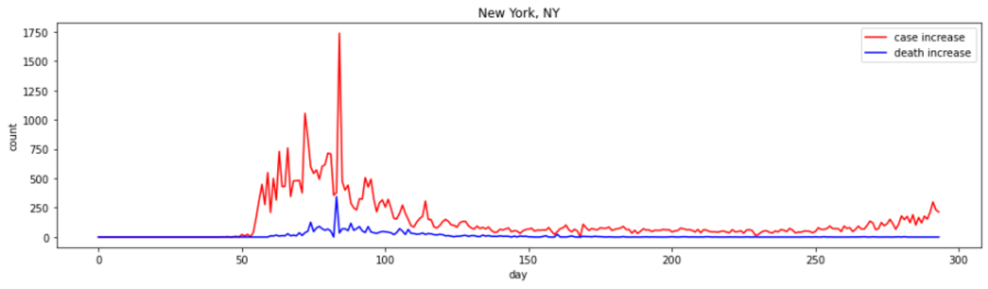
   Each county has a different level of medical resources available (more factors not shown), here are 2 example possible factors plotted in the histogram:

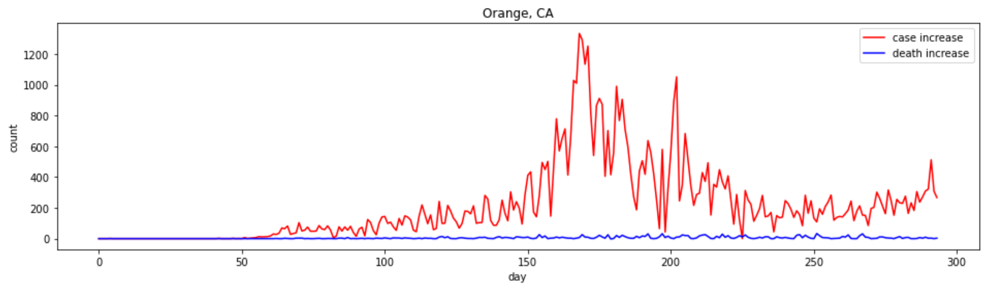




[1] Yu-Group/covid19-severity-prediction https://github.com/Yu-Group/covid19-severity-prediction

Each county also has a different rate of increase in cases and death, which could be influenced by various factors. Here is an example of 2 counties' severity plot*:





3. Explanations:

   County-level data provides a lot of details about the condition of each county in various aspects, which are related to the severity to a different extent. A model that successfully forecasts the severity should understand how the variables (both county condition and the severity of recent days) jointly contribute to the severity of a given day in the future.

*: We also attach a T-SNE plot in the appendix

4. Description of Technical Approach

- Logistic Regression (widely used in pandemic studies)

  We are using the SI model as a representative of the traditional logistics model that is widely used in epidemic infection prediction. The SI model will predict the number of infections over the days. In the SI model, the population will be divided into two categories: Susceptible and Infectious; and assume that the infection rate(r), the initial infections(x_0), and the maximum infections(x_m) remain unchanged.
  The Sigmoid function is shown below:

$$x(t) = \frac{x_m x_0 e^{r_0 t}}{x_m + x0(e^{r_0 t} - 1)}$$

Figure 1

The main point of the SI model is to determine the three key parameters: r_0 represents the initial infectious rate of the epidemic; x_m represents the maximum infections, which should be greater than the real maximum infections; and the $x_0$represents the initial number of infections. Since it is hard to get the real data of the three key parameters at the beginning of the epidemic, we are using the least-squares approximation to fit the logistic curve.

Since the least square regression would tend to find the local optima of the function to fit the case or the death curve, we also apply the differential evolution to find the best global initial parameters for the least square algorithm.

- Recurrent Neural Networks (popular deep learning approach on forecasting)

Recurrent Neural Networks (RNNs) is a type of Neural Network that is shown to be effective at forecasting sequential time data. It is previously widely used in natural language processing and weather forecasting.

The RNN block takes in case and death each day of a given county as a sequence Y. The Encoder takes in only the county feature X. RNN encodes its understanding of the sequential data as a hidden state h. The encoder encodes its understanding of the condition of the county as a latent space r. Then h and r are concatenated to jointly represent the current covid severity of this county. 2 residual decoders are then used to decode the joint representation $[h, r]$ to predict case and death increase.

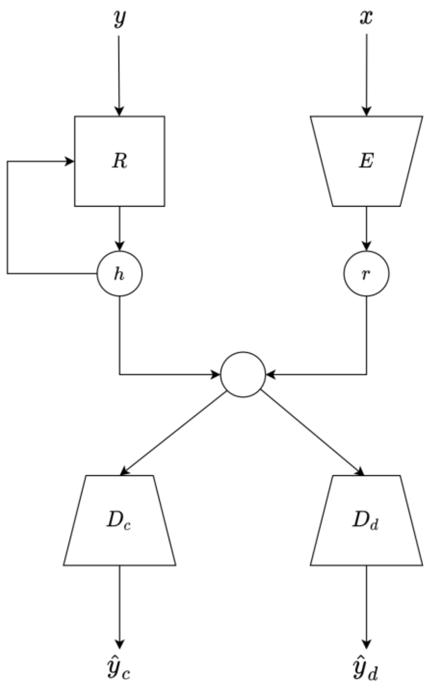


Figure 2 LSTM Structure with Encoder-Decoder

5. Software

- Data Loading and Sampling:

1. Original Data: from Professor Yu’s Covid Severity Prediction Project

2. Data Filtering: Filtering out data that has invalid feature values or records and selecting necessary features only for each valid data by using *Pandas.DataFrame*

3. Data Sampling: Sampling and shuffle data to form training and validation dataset by using *sklearn.utils.shuffle*

4. Consistency and Optimization: Store processed datasets in binary encoded files by using pickle to guarantee the consistency of the data that will be applied to different models, as well as reduce the loading time of the data.

- A High-level description of the SI Logistic Regression Model

  1. Best Initial Parameters: Find the best initial parameters by using *differenetial_evoluation* function from *scipy.optimize*

  2. Curve Fitting: Using the logistic regression function to fit the cases and deaths curve by using *curve_fit* function from *scipy.optimize* with the best initial parameters.

- A High-level description of our training pipeline of the Recurrent Neural Network

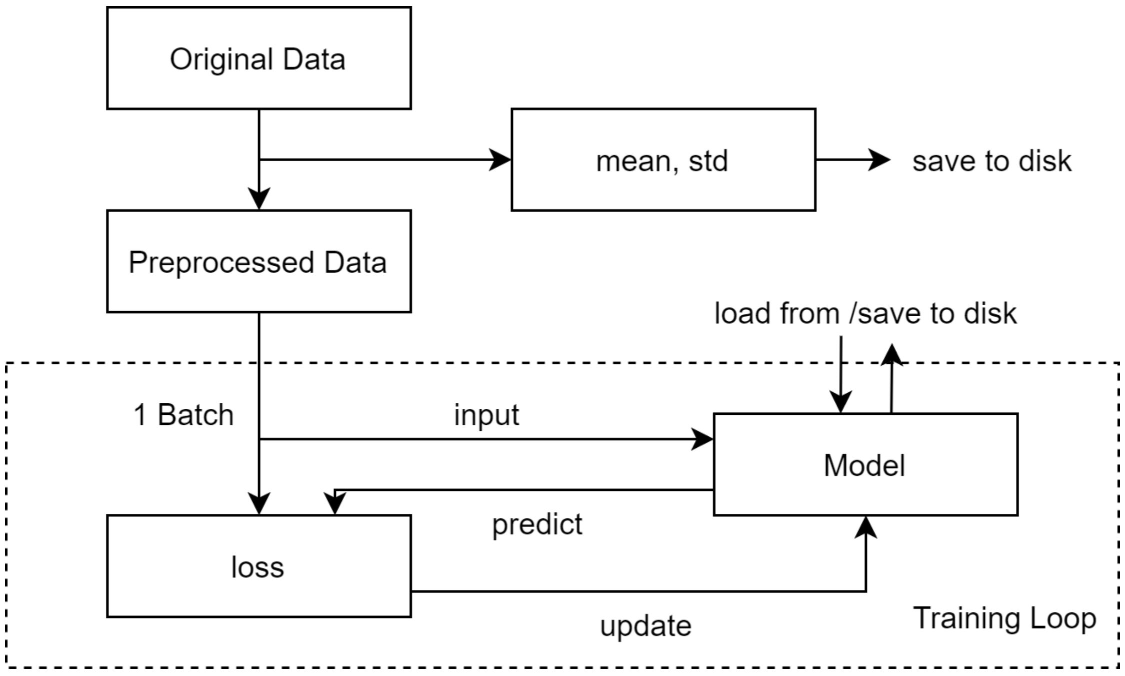


Figure 3 RNN Training Pipeline

1. Preprocessed Data: data is loaded using pandas, cast to torch tensor and normalized column-wise, mean, and standard deviation of each column is saved for inference

2. Data is cut in batch using *torch.utils.data.Dataloader*

3. Training the model involves 5 steps: forward pass, backward pass, record loss, parameter update, save the model to disk

6. Experiments and Evaluation

We experimented with many variations of models, here are some mile-stone models that we have experimented with.

- Logistic Regression

The SI Logistic Model will take the number of days t since the epidemic as the parameter to predict the total number of cases or deaths. According to our experiment, the model is underfitting the actual data especially in the late stage of the epidemic with the second peak of infections. The result is in line with our expectation since the SI model will only indicate the correlation between days t and the number of infections y, regardless of other variables, such as weather, local medical condition, recovered cases, etc., with potential influence, which will cause the neglect of key information in the model. Also, since the concept of the SI Logistic Model is that the epidemic will eventually be controlled and the increment rate of infection and death number will decrease and approach 0, the model is not suitable to predict epidemic with multiple outbreak waves.

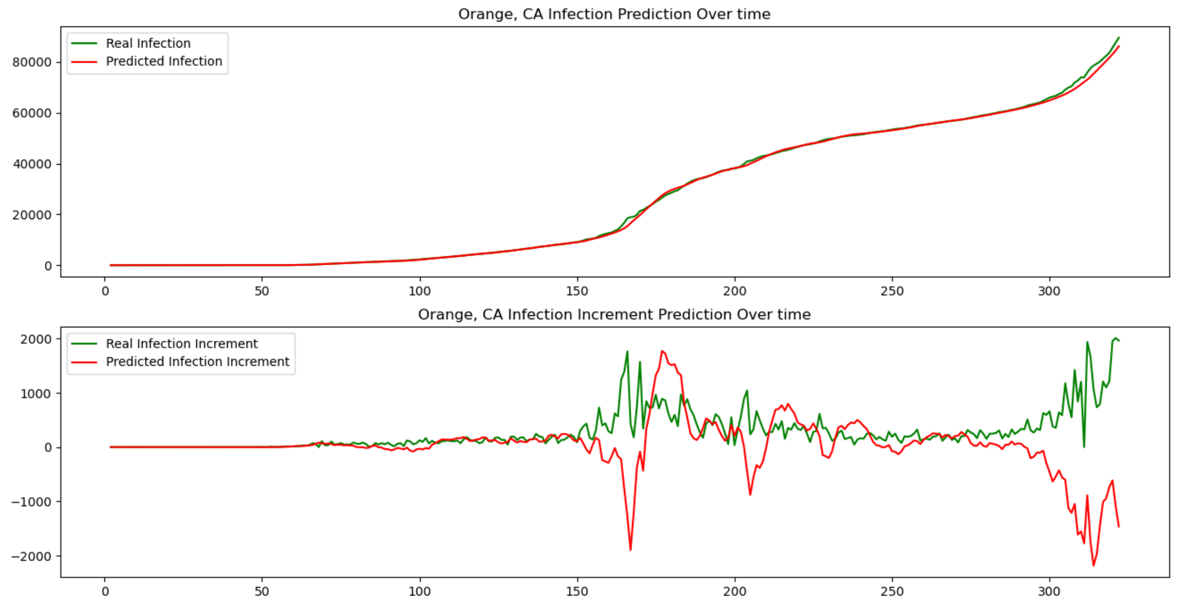


Figure 4 Orange County Infection Prediction

- All-in RNN with vanilla decoder

  The RNN takes in both county features X which is repeated and concatenated with case and death each day Y. The network predicts the case and death given consecutive days of data of case and death. This model is underfitting the data, especially the case number. We then think that to separately tune the case and death prediction; it is necessary that we have 2 decoders that have different complexity.

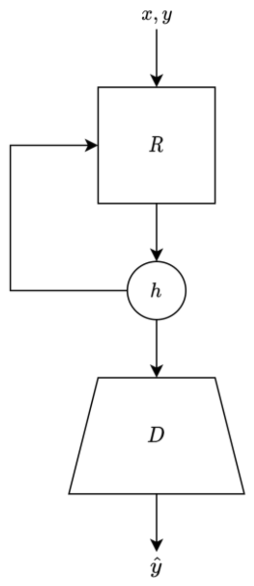


Figure 5 RNN Structure with Vanilla Decoder

- All-in-RNN with separate residual decoder

  The RNN takes in both county features X which is repeated and concatenated with case and death each day Y (same as the previous model). To achieve more sophisticated fitting, 2 residual decoders (different depth) are added to replace the fully connected layers to fit the data better and to counter the diminishing gradient problem. Residual blocks give the gradient an uninterrupted flow to the earlier layers of the network. To disjointly tune the fitting of case count and death count, 2 decoder's hyperparameter can be tuned with less influence on the other compared to 1 decoder structure. This network easily achieves better fitting than the previous one, but it does not fit well in populous counties. We only want to increase the complexity without interfering with the recurrent block which processes the sequential data well.

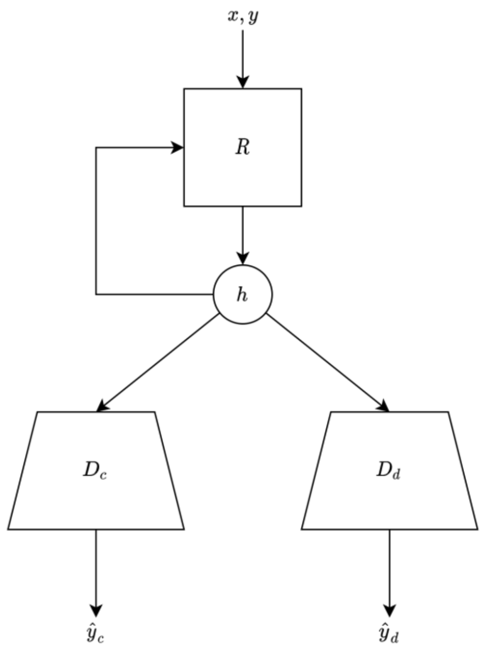
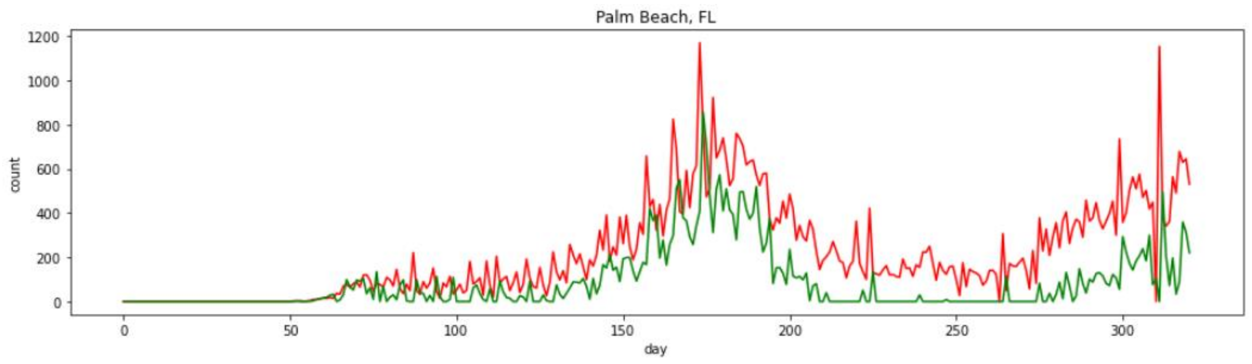


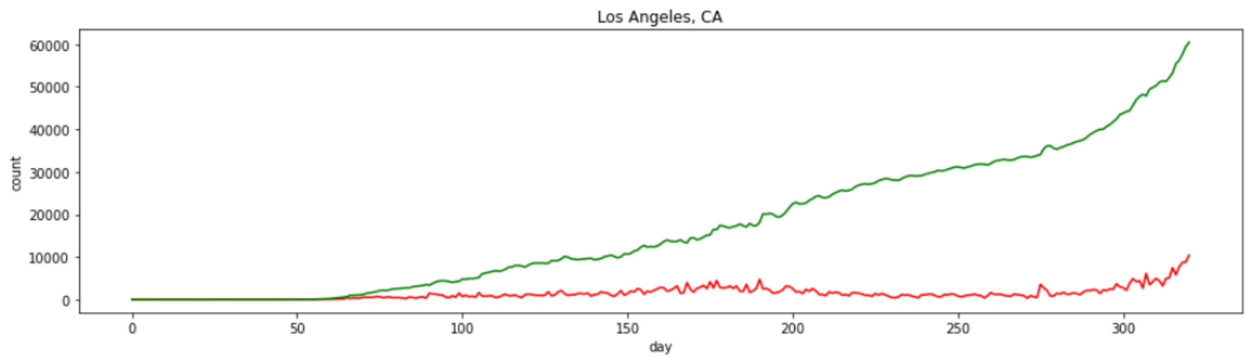
Figure 6 RNN Structure with Separated Residual Decoder

Figure: model not fitting well on more populous counties

1. Real (red) vs 1-day look-ahead prediction (green) of case increase of Palm Beach, FL



2. Real (red) vs 1-day look-ahead prediction (green) of case increase of Los Angeles, CA

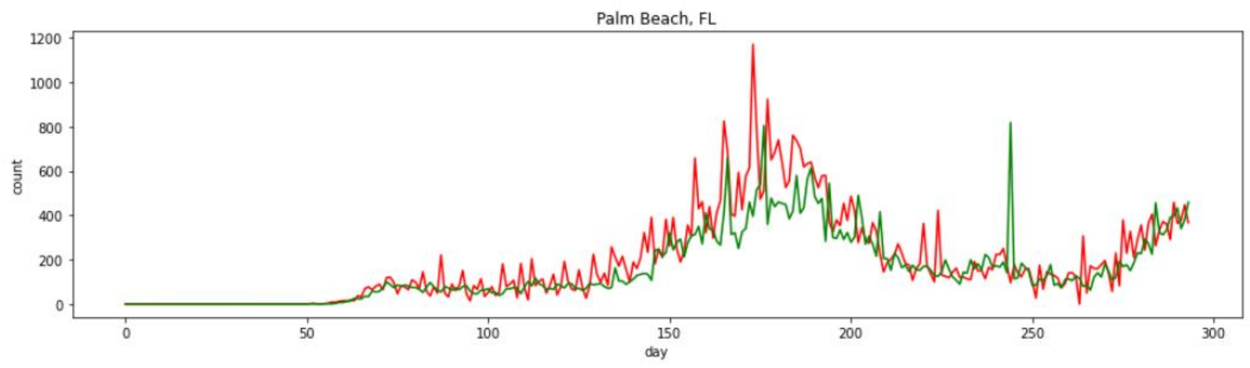


- **LSTM with encoder-decoder (final model network structure: See section 4 figure 2)**

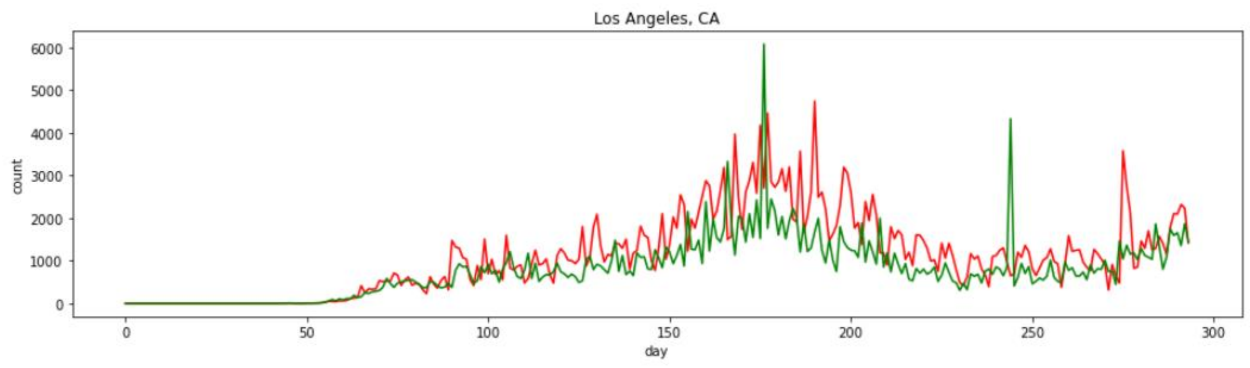
  In this variation, the recurrent block only takes in the sequential part of the data. The encoder encodes the fixed-length county feature to a latent space r. This way, the encoder, and recurrent block can be tuned separately to fit the complexity of the data. Our result shows that this fits better in both large and small counties. In this case, the RNN block captures the important information only from the sequential part of the data, while the encoder only encodes the county feature. This design eliminates the interference from fixed data to the sequential network, which should give a better hidden state that represents the intensity on its own. The decoder then accounts for both the past severity and the county's condition to generate the prediction. This network easily overfits the training data if no regularization method is used. In our experiments, early stopping at 70-100 epochs with learning rate $1e-4$ achieves good performance on both training and testing data.

Figure: model fitting better on more populous counties

1. Real (red) vs 1-day look-ahead prediction (green) of case increase of Palm Beach, FL



2. Real (red) vs 1-day look-ahead prediction (green) of case increase of Los Angeles, CA



The overall performance (table 2) of some of the mile-stone models, along with the state-of-the-art model performance is recorded. More detailed data are available in the appendix.

Model Name	Test avg 1-day-ahead MSE (case increase)	Test avg 1-day-ahead MSE (death increase)
Logistic Regression.	3873.018	3.857
Vanilla LSTM	6207.78	1.785
LSTM with Encoder-Decoder	399.186	0.898
Yu-Group's CLEP Predictor	8.427	Not evaluated

Table 2

7. Discussion and Conclusion

1. The SI logistic regression did not perform as well at the county level as expected due to county condition variation are not considered.

2. The SI logistic regression has parameters that are more explainable than the deep learning approach.

3. The separation and different depth design of decoder heads compensate for the difference in complexity of case and death data.
4. The separation of sequential and feature parts of the data increased model performance.
5. The current performance of our model captures the trend of change, but not precise.
6. Further decomposition of the NN structure may have positive effects on performance.*
7. LSTM may be used to do gradient boosting on the error of logistic regression.*
8. Professor Yu's CLEP predictor is currently state-of-the-art.

*: interesting things to try if we can work on this in the next year or two in a lab

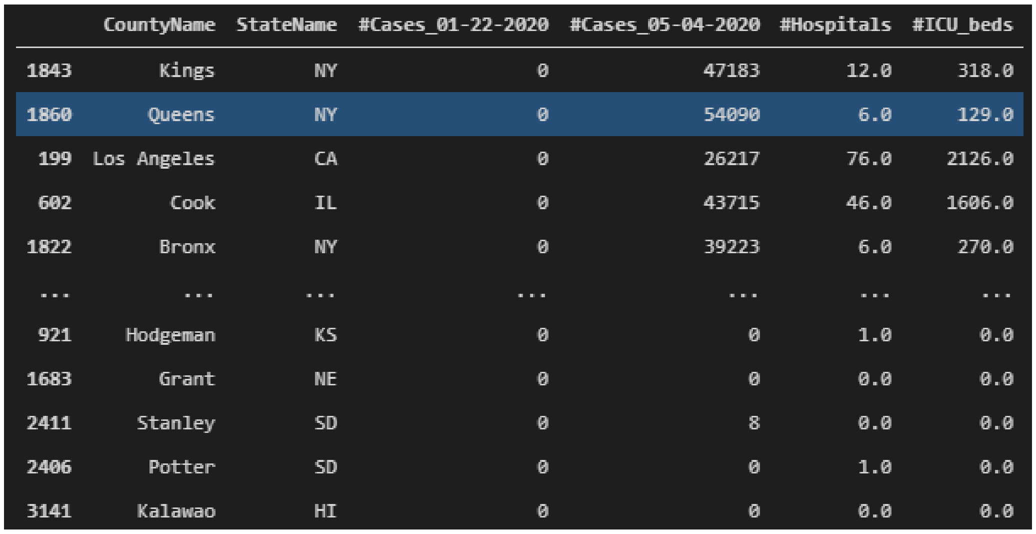
8. A separate page on Individual Contributions

- Yilong Huang
  - Data preprocessing
  - Designed and tuned Logistic Regression learner
  - Evaluated the performance of forecasting of the Regressor
- Yaosheng Xu
  - Data preprocessing, Data visualization
  - Designed and tuned RNN and LSTM module
  - Evaluated the performance of forecasting of the Neural Network
- Litian Liang
  - Designed and tuned decoder and feature encoder
  - Recurrent neural network hyperparameter tuning
  - Evaluated the performance of forecasting of the Neural Network

Appendix

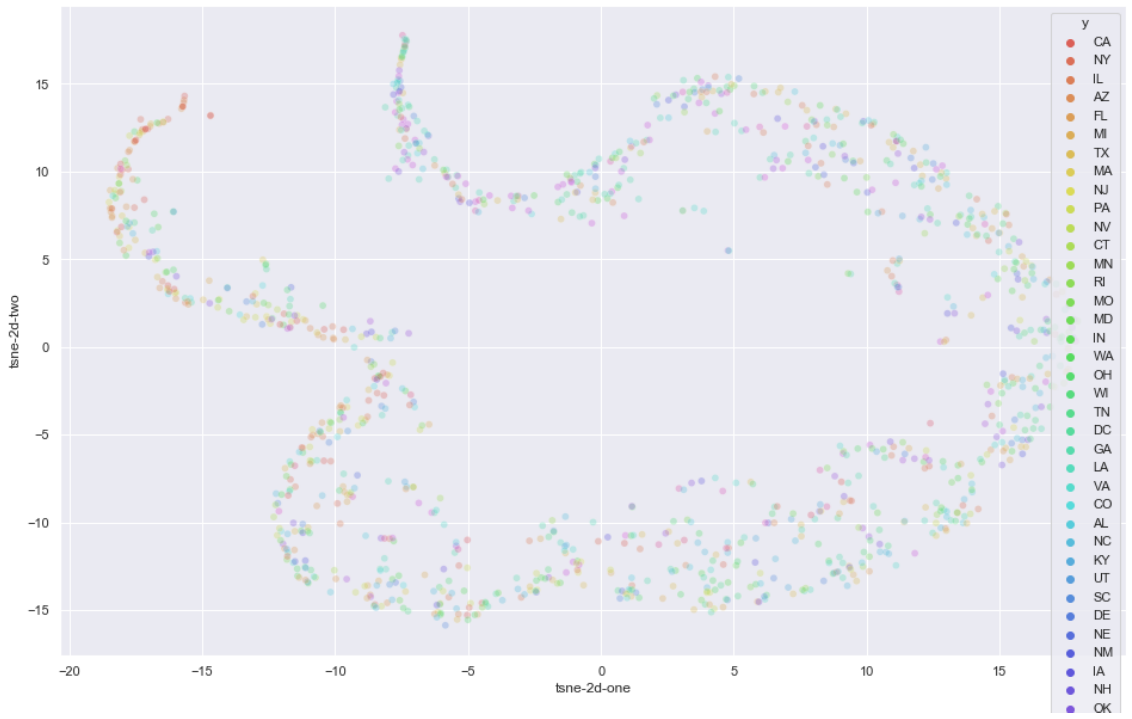
Data Visualization

An example of some columns and rows of data (not shown in full):

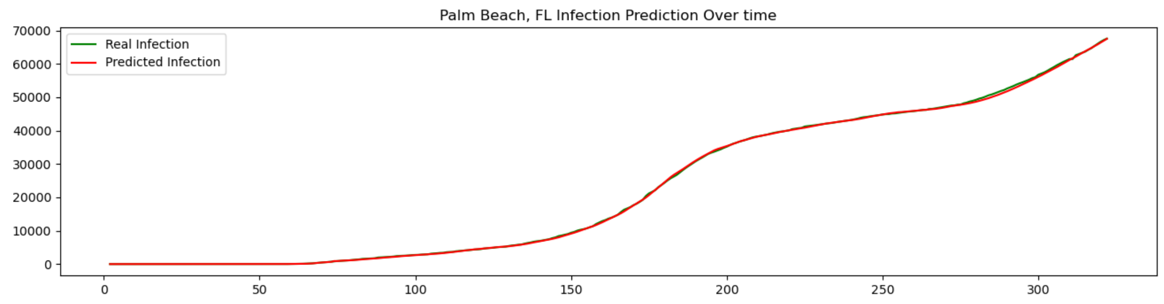


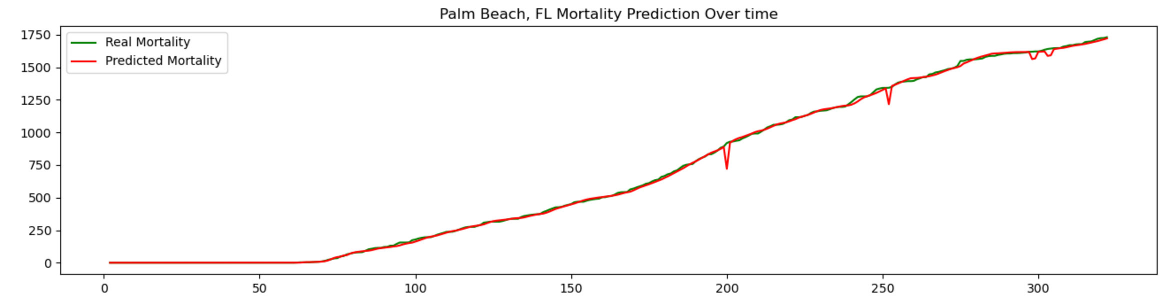
	CountyName	StateName	#Cases_01-22-2020	#Cases_05-04-2020	#Hospitals	#ICU_beds
1843	Kings	NY	0	47183	12.0	318.0
1860	Queens	NY	0	54090	6.0	129.0
199	Los Angeles	CA	0	26217	76.0	2126.0
602	Cook	IL	0	43715	46.0	1606.0
1822	Bronx	NY	0	39223	6.0	270.0
...	...	...	...	...	...	...
921	Hodgeman	KS	0	0	1.0	0.0
1683	Grant	NE	0	0	0.0	0.0
2411	Stanley	SD	0	8	0.0	0.0
2406	Potter	SD	0	0	1.0	0.0
3141	Kalawao	HI	0	0	0.0	0.0

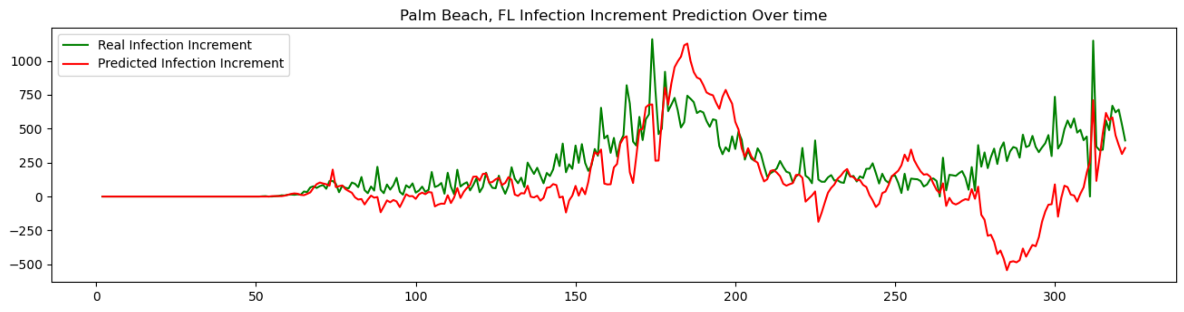
t-SNE visualization

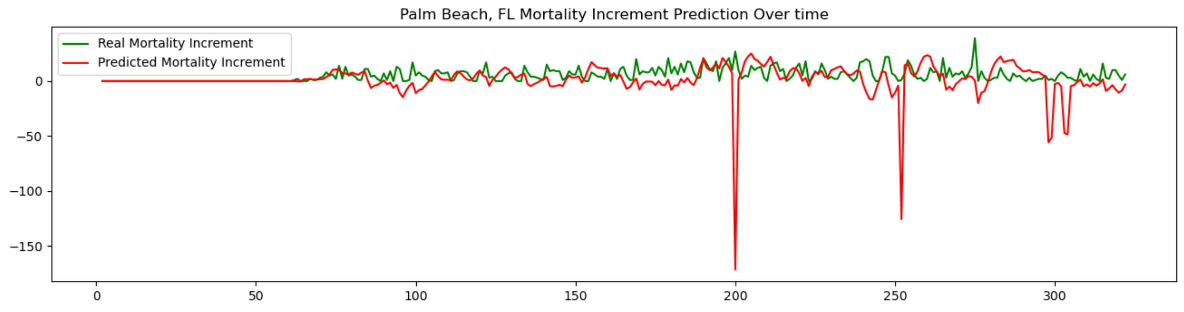


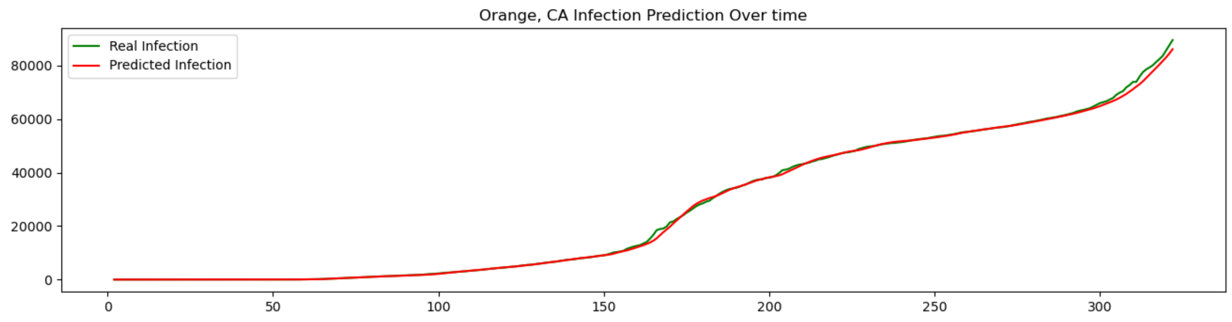
SI Logistic Model

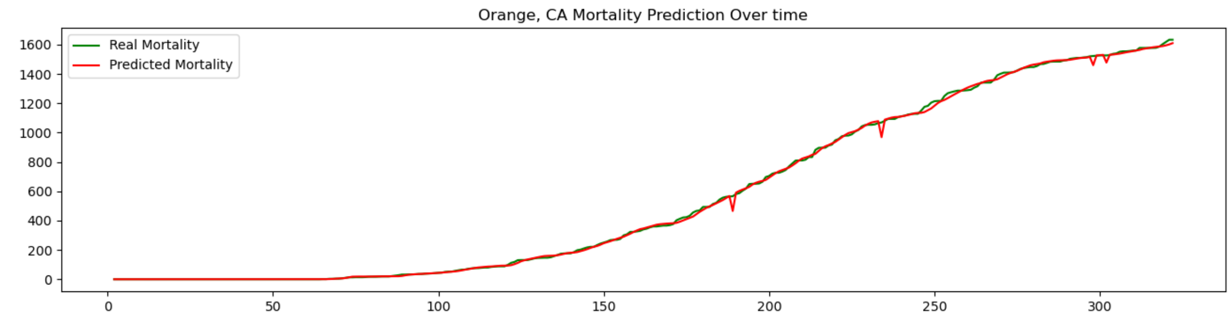


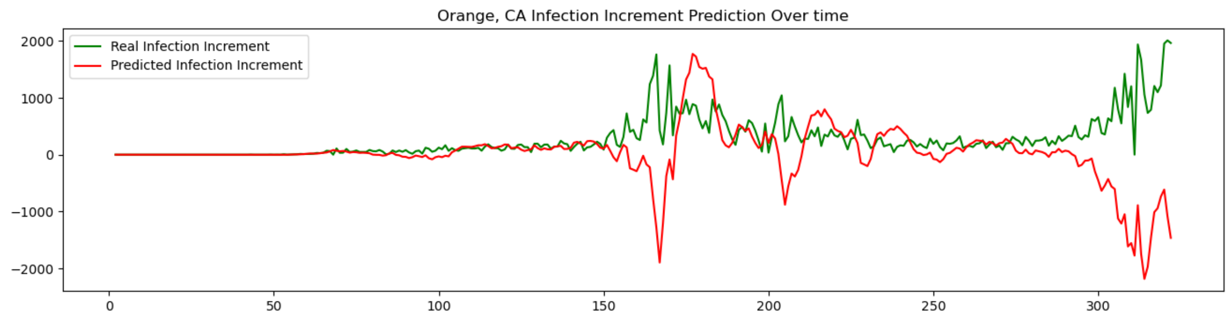


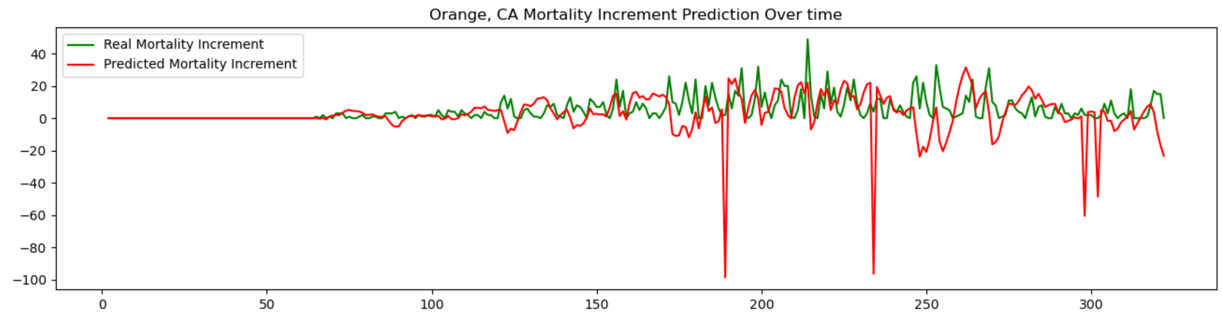






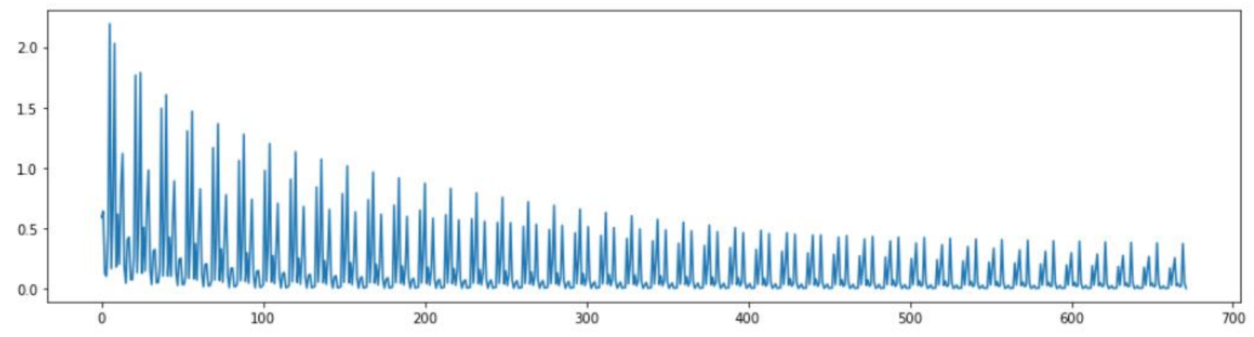






Vanilla Long Short-Term Memory

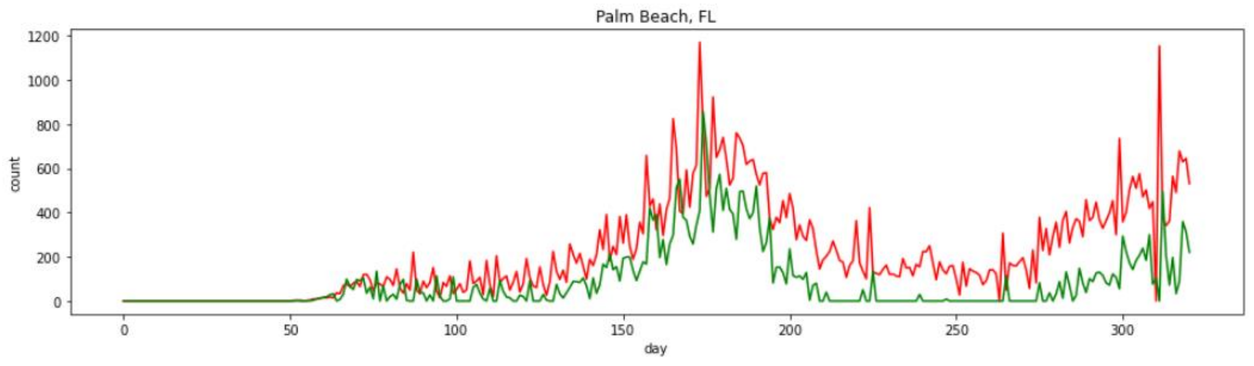
Training Loss

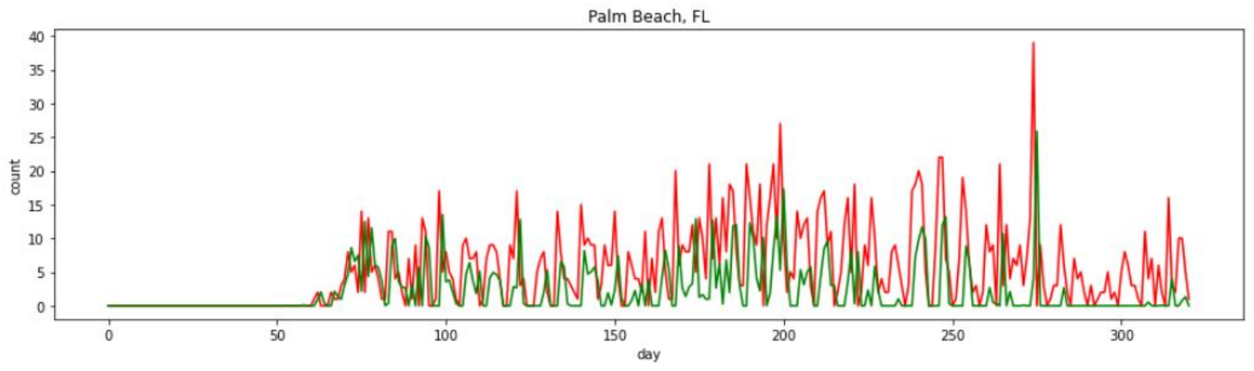


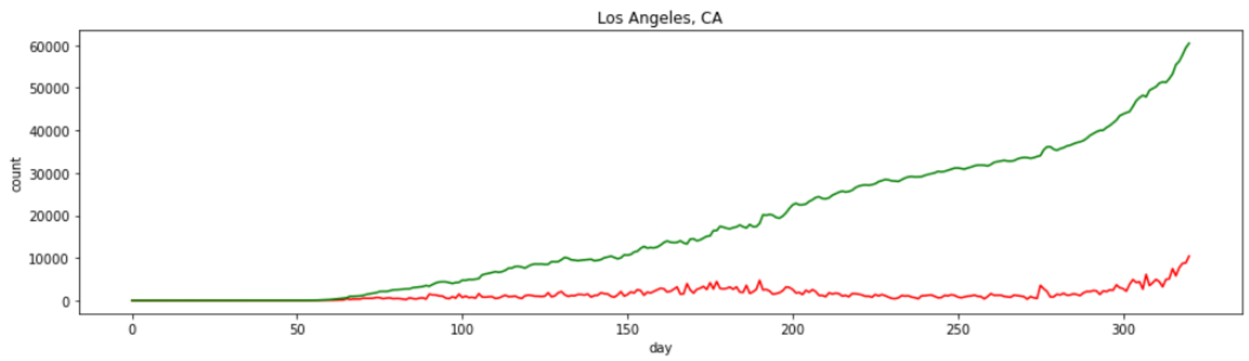
Test case, death 1-day look ahead MSE after convergence (best result)

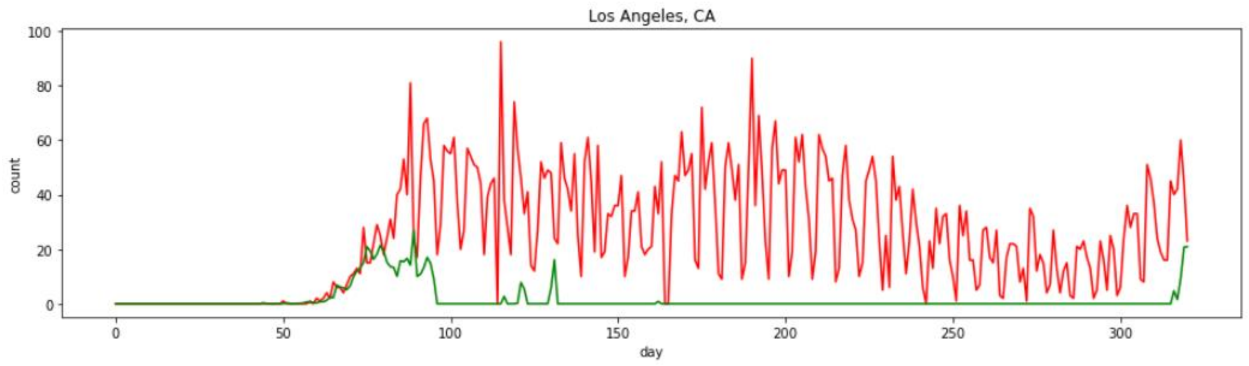
(6207.78, 1.785)

Example prediction (case, death of county):





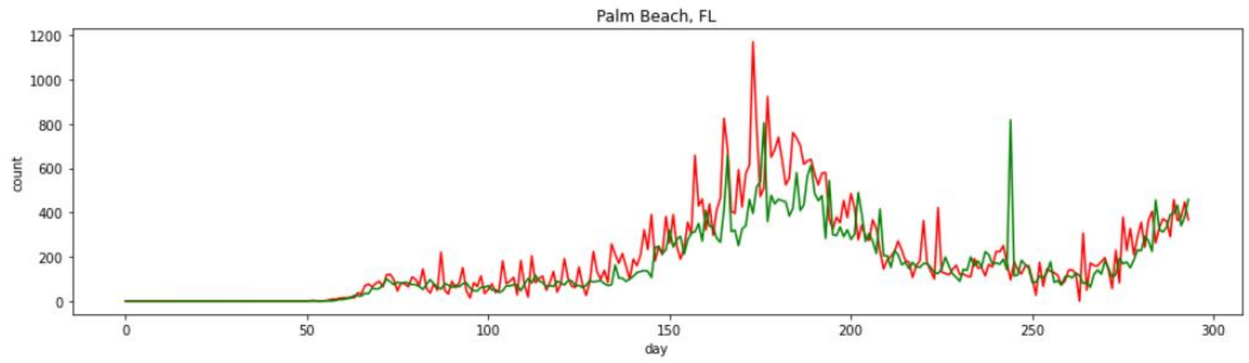


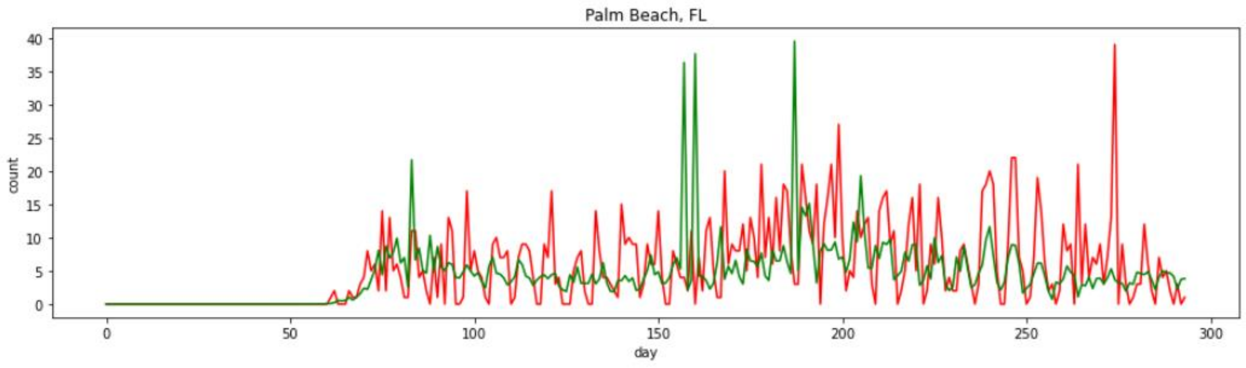


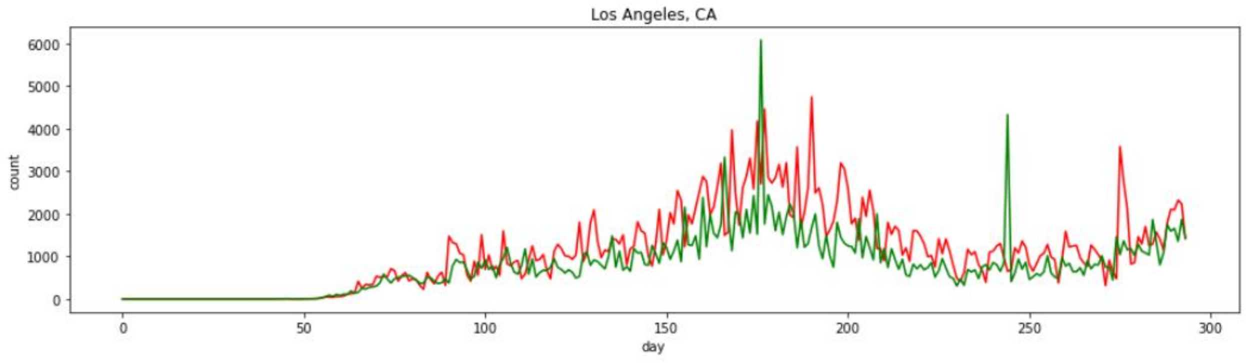
LSTM with encoder-decoder

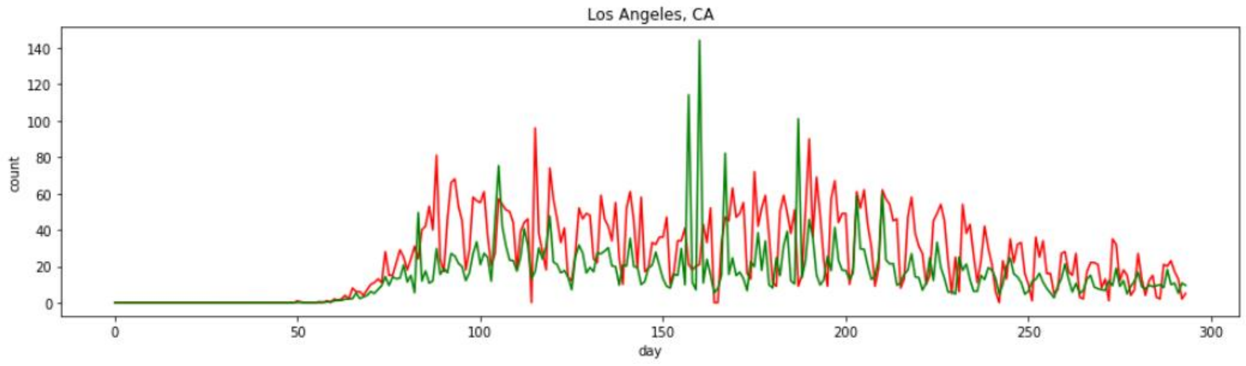
Test case, death 1-day look ahead MSE after convergence (best result)

(399.186, 0.898)




Palm Beach, FL
count
day
0
5
10
15
20
25
30
35
40
0
50
100
150
200
250
300


Los Angeles, CA
count
day
0
1000
2000
3000
4000
5000
6000
0
50
100
150
200
250
300


Los Angeles, CA
count
day
0
20
40
60
80
100
120
140
0
50
100
150
200
250
300