Machine Learning Engineer: Capstone
Predicting and Modeling Epidemics

Christian Urrea

Project Definition

I. Overview

The availability of data and access to it is growing by leaps and bounds with each passing day. Data, along with computerized analysis and algorithms, including machine learning algorithms, have given us a new way to gain insight of the world around us and predict, to a degree, the likelihood future events given the present and past observed data. But not all areas where major problems exist have experienced a growth of data that can help us better understand and apply machine learning to help solve it. One such area is in epidemiology, where accurate, reliable epidemiological data is extremely scarce compared to market trading data or social network data -- and we shouldn't expect to observe a significant growth of epidemiological anytime soon.

But at the same time epidemics are severe threat to any living population. Historically epidemic outbreaks such as yellow fever, dengue, measles, cholera, smallpox, tuberculosis, influenza, and more recently Ebola, have affected human populations with great consequences. Epidemics are a worthy problem to apply machine learning on. Models can be built in order to give us a means of predicting an epidemic's severity early in its cycle - in order to take the necessary actions needed to handle the epidemic and possibly even stop it early in its cycle. Something that only a century ago would be a dream. And we shouldn't let lack of data stop us.

This project tackled the severe lack of accurate and reliable epidemiological data by turning to epidemic mathematical models and, through them, generating artificial data by simulation and appropriate randomization of key epidemic parameters of the model, within realistic ranges. The data was organized into sequential timestep matrices and was used to train a recurrent neural network to predict the three key parameters of all epidemics behavior: the infection, recovery and death rate. The implication being that we can apply machine learning to predict epidemics parameters and gauge an epidemics severity early in its life cycle, before it inflicts severe loss.

II. Problem Statement

Detecting and predicting epidemics is a problem that there appears to be no clear, reliable solution for. There have been different, innovative approaches to the problem, such as clustering algorithms based on health predictive analysis to classify areas that will experience epidemics

[1], similarly deep learning models based off healthcare data that predict where the next outbreak of epidemic will be **[2]**, and even Bayesian networks trained on simulated data for epidemic detection/classification **[3]**.

All previous literature focused on classifying potential epidemic areas or classifying the presence of an epidemic. Instead of these, the aim of this project was to create a deep learning model that could *regress* the key parameters that underlie the behavior of an epidemic - and do so early from onset i.e. within the shortest possible time-span. The model's prediction of the epidemic's key parameters could then be followed up and used to simulate an epidemic through mathematical modeling to estimate the future behavior of the epidemic, and thus in a sense, predict the future behavior of any epidemic.

Briefly, we will do this in the following way:

1. Run 10,000 simulations of the epidemic model **[4]** in which the key parameters (infection rate, recovery rate, death rate) are randomly varied within realistic ranges.
2. For each simulation, collect the artificial data of the first 10 days for features such that are feasibly observable in a real world setting and save the values of the three key parameters that were used .
3. Format the time series artificial data, the feature values at each time step (X), with the corresponding key parameters values (y). *Use this as the data to train and optimize a recurrent neural network that can predict the three key parameters that describe the raw behavior of an epidemic.
4. Finally, Use those predicted key parameters to model the epidemic and gauge the expected consequences of an epidemic.

The implications of this is that we can create public health algorithms that can not just detect and flag possible epidemics latent in a population, but as well predict the key parameters of them which can be used to simulate the behavior of an epidemic on a population and gauge the epidemic's magnitude, so it can be handled appropriately, checked -- before it wreaks havoc.

III. Metrics

Considering that the problem is a multiple regression, the most appropriate error metric to train the recurrent neural network was deemed to be the mean squared error (MSE) metric: $MSE = 1/n \ \Sigma^n (Y_i - \widehat{Y}_i)^2$. At each step of training, the difference between the model's predictions and the true values were squared and summed, then averaged by the number of observations.

Data Analysis & Preprocessing

IV. Epidemic Simulations

As accurate and reliable epidemic data is scarce, not nearly enough of it exists to feasibly train a recurrent neural network. Thankfully mathematicians have studied extensively the behavior of epidemics from which they've have created mathematical models that quantify the behavior of an epidemic on a population based on a few key parameters. For this project one of the classical mathematical models, a variant of the SIR model developed by Kermack and McKendrick **[5]** was used to simulate 10,000 epidemics to generate artificial data and create our dataset.

To Kermack and McKendrick's original SIR model, the following was added to our simulations:

- Death parameter δ - important addition not found in original SIR model
- Stochasticity - add noise in the simulation that is akin to reality

The final epidemic **stochastic SIRD [6]** model was implemented follows:

$$N_t = S_t + I_t + R_t - D_t$$

Where the population is:

- N_t **:** number of still people alive at time step t

And where the variables that make up the population correspond to:

- S_t : number of people susceptible at time step t
- I_t : number of people infected at time step t
- R_t : number of people recovered at time step t
- D_t : number of people deceased at time step t

At every time step, population dynamics change as described by partial differential equations:

- $\partial S/\partial t = -\beta SI/N$
- $\partial I/\partial t = \beta SI/N - \partial R/\partial t - \partial D/\partial t$
- $\partial R/\partial t = \alpha I$
- $\partial D/\partial t = \delta I$

The key parameters underlying the partial differential equations being:

- β : Infection Rate, in terms of probability of being infected if susceptible given proportion of infected
- $\propto$: Rate of recovery, in terms of probability of recovering if infected at each time step t
- δ : Rate of mortality, in terms of probability of dying if infected at each time step t

For each simulation the parameters of our **stochastic SIRD** model were instantiated as follows:

- The total initial susceptible population (S_0) was instantiated 100,000 for all simulations.
- The initial number of infected (I_0) was randomly selected integer between 45-55. That is, approximately 0.05% initial infected for the 100,000 pop. model.
- The infection rate (β) was randomly varied float between range of [0.15 - 0.8].
    - Range was constrained to realistic infection rate values, as β close to 1 would signify that every infected person infects a susceptible individual *almost* everyday that he lasts infected. Further, epidemics with very low infection rates $\beta < 0.1$ never lift off and thus were not considered *practically* significant enough to add.
- The recovery rate (α) was randomly initialized dependent on the infection rate, as a float ranging between 0.1 and $\beta - 0.05$.
    - It is realistic to assume that the rate of recovery from an endemic disease is slower than rate of which you infect someone. If the recovery rate were larger than the infection rate, the epidemic would be insignificant, and thus not relevant for the problem. Further, when $\alpha > \beta$, people recover before they are likely to infect another person, and thus very few people tend to become infected throughout the cycle and no "epidemic" occurs.
- The death rate (δ) was randomly initialized in range of [0.01 - 0.05]
    - The range was decided from observations that simulation with less than 0.01 value for the death rate rarely resulted in deaths in the whole epidemic cycle, while rates higher than 0.05 usually resulted in population extinction.

We hypothesize that the pool of combinations of appropriately randomized key parameter values (infection rate, recovery rate and death rate) capture a large proportion of *realistically possible* epidemics, ranging from tepid to catastrophic. Which, from a practical standpoint, is what we wish our model to learn - and to determine.

V. Simulation Features & Dataset

For every simulation we gathered data of *14 features* each end of day observation. The particular features recorded from the simulation to build our dataset were chosen by **two criteria**:

1. **Practicality** - criteria that our dataset should be made up of features that can be easily and realistically be gathered and tracked during the onset of an epidemic. That is the features should be realistic - in so much as feasibly collected in the midst of an epidemic.

2. **Non-Autocorrelation**- important criteria to avoid bias of our generated artificial data, by omitting features which are directly related to any of the recovery, infection and death parameters our model seeks to predict.

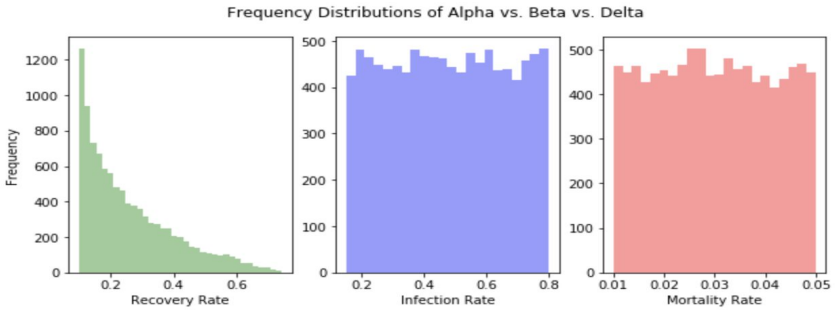
The final realistic and non-autocorrelated features, gathered of the 10 first days of an epidemic, that make up our dataset are cataloged, in the table below.

Feature	**Description**
RI^{t}_{ratio}	Ratio of recovered over infected at time t
IS^{t}_{ratio}	Ratio of infected over susceptible at time t
RS^{t}_{ratio}	Ratio of recovered over susceptible at time t
DI^{t}_{ratio}	Ratio of deceased over infected at time t
DS^{t}_{ratio}	Ratio of deceased over susceptible at time t
RD^{t}_{ratio}	Ratio of recovered over deceased at time t
Num_Scspt^{t}	Total # of susceptible at time t
Num_Inf^{t}	Total # of infected at time t
Num_Recov^{t}	Total # of recovered at time t
Num_Dsc^{t}	Total # of deceased at time t
$delta\_Scspt^{t}$	Daily change in total # susceptible at time t
$delta\_Inf^{t}$	Daily change in total # infected at time t
$delta\_Recov^{t}$	Daily change in total # recovered at time t
$delta\_Dsc^{t}$	Daily change in total # deceased at time t

VI. Targets: Epidemic Parameters

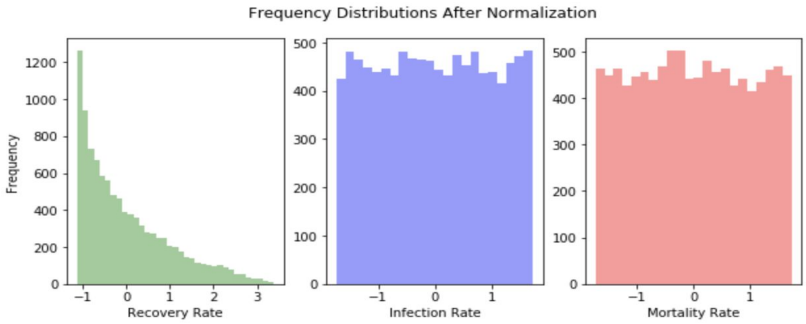
The model will aim at predicting the parameters of an epidemic, the recovery rate (α), infection rate (β), and death rates (δ) of the epidemic based on the 14 indirect and easily

observable features gathered, from the first 10 days of an epidemic. The following are the frequency distributions for the parameters used to initialize the training simulations:



	beta	alpha	delta
count	10000.000000	10000.000000	10000.000000
mean	0.475884	0.260844	0.029921
std	0.187783	0.142652	0.011494
min	0.150030	0.100000	0.010005
25%	0.313995	0.144388	0.020122
50%	0.474230	0.220060	0.029726
75%	0.636730	0.344262	0.039759
max	0.799860	0.743040	0.049990

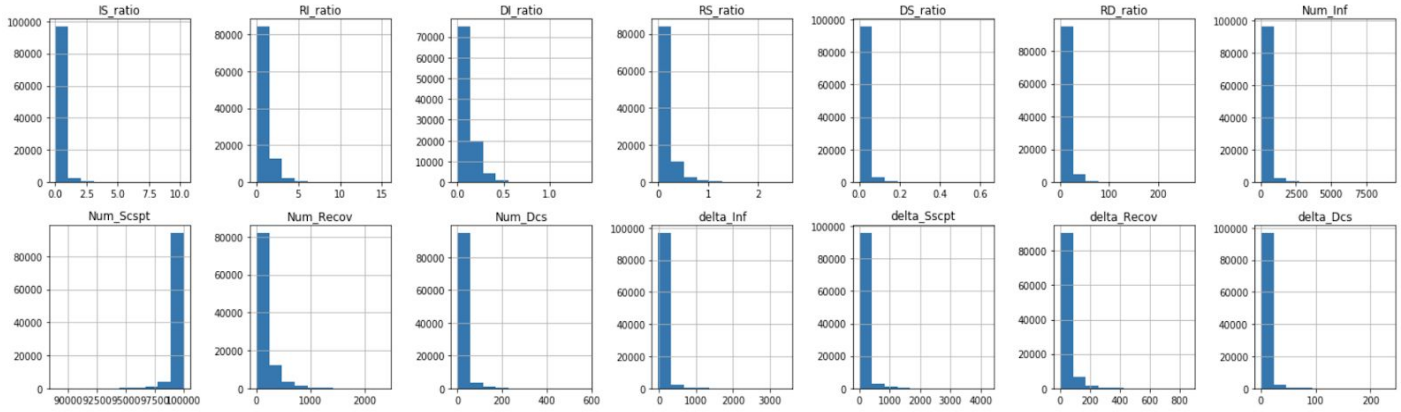
After encountering issues** in training with feature values of different ranges, standardization was applied to our feature values to approximate them to with the same range window. The results of standard transformation of the target variable were as follows:



	beta	alpha	delta
count	1.000000e+04	1.000000e+04	1.000000e+04
mean	-1.897593e-16	-7.773782e-17	-3.800016e-17
std	1.000050e+00	1.000050e+00	1.000050e+00
min	-1.735353e+00	-1.127584e+00	-1.732745e+00
25%	-8.621479e-01	-8.164093e-01	-8.525536e-01
50%	-8.807236e-03	-2.859135e-01	-1.695133e-02
75%	8.565958e-01	5.847979e-01	8.559754e-01
max	1.725354e+00	3.380395e+00	1.746085e+00

VII. Data Exploration & Transformations

The initial distributions for each of our raw features, before transformations, was as follows:

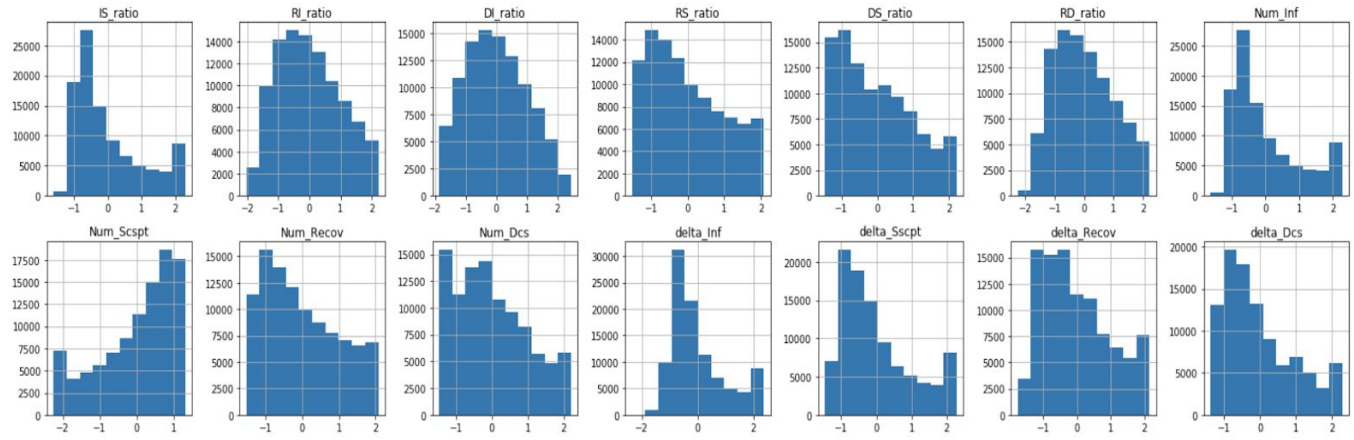


It was observed that all our features contained extreme skews towards one direction. As well, all features had the with majority of data points concentrated around one end of the range,

and other data points sporadically found along the extreme skew. Outlier removal was not considered as it would result in the omission of the data observations we are most interested, those of high magnitude epidemics that are responsible for the extreme high/low feature values. All in all, our feature distributions were such that a neural network would be unable to learn because of the nature of the variance. Thus the following series of transformations was applied to the features to make them amenable for training our neural network:

1. *1st Standardization:* Features were standardized with mean of 0 and standard deviation of 1 to reduce skew and stabilize variance.
   Result: The extreme skews and high leptokurtosis remained present.
2. *Min-Max + Box-Cox Transform:* A combination was used of 1) min-max scaler to put features in a positive range between (1,2) which was then followed with a 2) box-cox power transformation to normalize and further stabilize our features' variance **[7]**.
   Result: The extreme skew and high leptokurtosis was significantly reduced but absolute
   values now varied to a great degree.
3. *2nd Standardization***:** Finally, A second standardization was applied to scale the features to approximately the same range of values, mean of 0 and standard deviation of 1.
   Result: All features were in approximately similar range of values, approximately normal with little sign of skew or high leptokurtosis

The final distributions for each feature, after the series of transformations was applied:



.

Model & Final Results

VII. Model Training

The total training set of 9,000 stochastic epidemic simulations generated from randomized parameters as described in IV. Epidemic Simulations. Each simulation resulted in an observation matrix that consisted of 10 time steps x 14 feature values, attached with corresponding epidemic parameters used to generate the simulation as targets, as described in V. Dataset Features.

Considering the data was a time series, we modeled a recurrent neural network, made up of bi-directional layers, as in experiments it consistently showed superior results than a unidirectional RNN **[8]**. Briefly, a bi-directional layer is simply a recurrent neural network that concatenates the outputs of two RNNs, one processing the inputs from future to past, the other one from past to future. The concatenated outputs are the predictions of the layer.

Many setbacks that were encountered in order to get our model learning and achieve the final results. The most important points being:

- When we trained the model without the series of feature transformations described in VII. Data Exploration & Transformation, the model outputted predictions near the mean, in other words the model failed to learn from the inputs and minimized error by making hard predictions near the mean. The problem was that the extreme outliers within the feature distributions was hindering the model's ability to learn. And considering that the outliers were representative feature values of catastrophic epidemics, it would have inhibited our model's practical use if they were removed. After research, experimentation, numerous trial and error runs, the model best learned when the inputs were transformed in the order as described:

1st Standardization → Min-Max Transform + Box-Cox Transform → 2nd Standardization

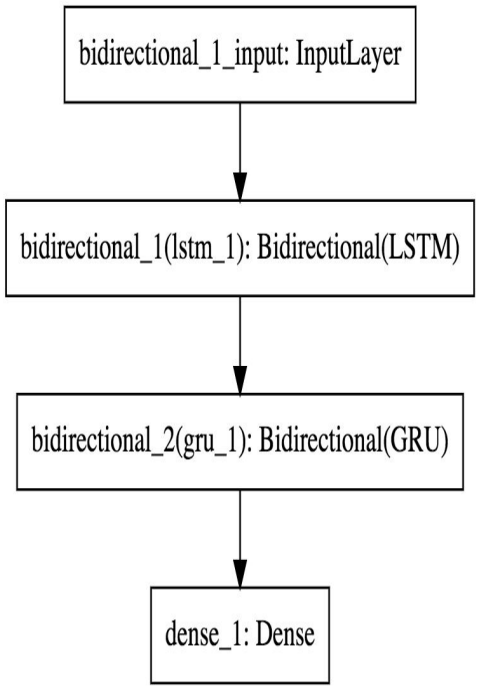
- Initially, the model failed to learn the parameter delta and did little better than the benchmark MAPE score. I then researched that standardization of y-values is necessary for neural network when learning multiple regressors as the regressors whose absolute values are larger will hold more influence in the backpropagation algorithm. Since the neural network will optimizes weight such that it minimizes *total* MSE error, it'll thus prioritize optimization of weights to predict the regressors with the highest absolute values at the cost of neglecting the ones with lower absolute values. This ws case the death rate parameter with range [.01 - .05] compared with infection and recovery rates

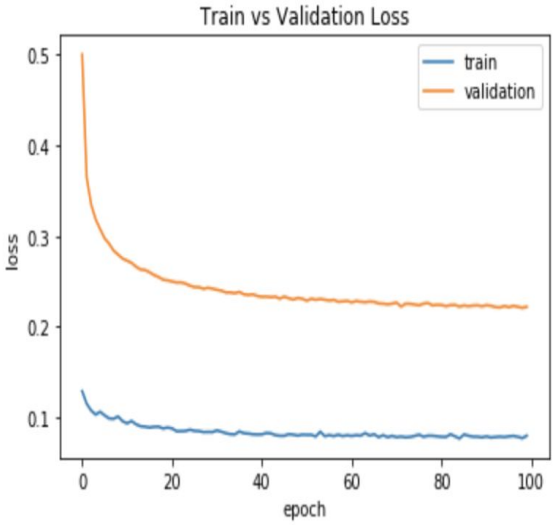
which ranged from [.15 - .8] and [.1 -.75], respectively. Thus applying standardization of to targets was *necessary* in order for the model to appropriately learn the delta parameter.

VIII. Final Model Architecture & Hyperparameters

After numerous experimentation runs, the best performing architecture & hyperparameters were found to be the following:

- Two layers, both bidirectional, and each bidirectional layer consisting of 14 nodes - coincidentally the same as the number of features.
- For all layers, the Xavier normal initializer was used to initialize the weights **[9]**.
- The first bidirectional layer was made up of LSTM cells and the second bidirectional layer of GRU cells - the theory being that less "memory" was needed to understand the first LSTM layer output. The results showed this was true *for this case*.
- For regularization, recurrent dropout regularization was used - set at 50% of nodes being randomly dropped at each batch for both layers.
- The bidirectional layers were merged in mode 'sum', instead of 'concat'.
- For training the network, the gradient descent variant, Adam Optimizer **[10]** was used with learning rate initialized at 0.001,
- Mean squared error was used as the loss function in model training.
- A batch size of 32 was used





Twenty percent (20%) of the training set was used to validate the model during training. The model was trained for 100 epochs. The validation vs. training plot showed signs of the model overfitting on the training data. Many values of dropout, as well as the combination recurrent dropout and dropout regularization, were applied in order to ameliorate the over fitting - to little success.

The best regularization found was recurrent dropout set at 0.5 for both layers. But the train/validation loss plot still reflect that this is an area where more work could be done to decrease the gap between training and validation errors, and consequently improve the model's generalization.

VIIII. Results

Although the model was trained with the mean squared error metric, it is problematic to report model's MSE score when target values range $0 < y < 1$, as they do in this case. For example say the mean error of an observation is 0.3, the mean squared error would then be 0.09 - a third of the real error, which can easily be misinterpreted. To avoid confusion, the model results will be evaluated in terms of the mean absolute percentage error (MAPE) metric **[11]**.

$$MAPE = 100/n \times \Sigma(Y_i - \widehat{Y}_i) / Y_i$$

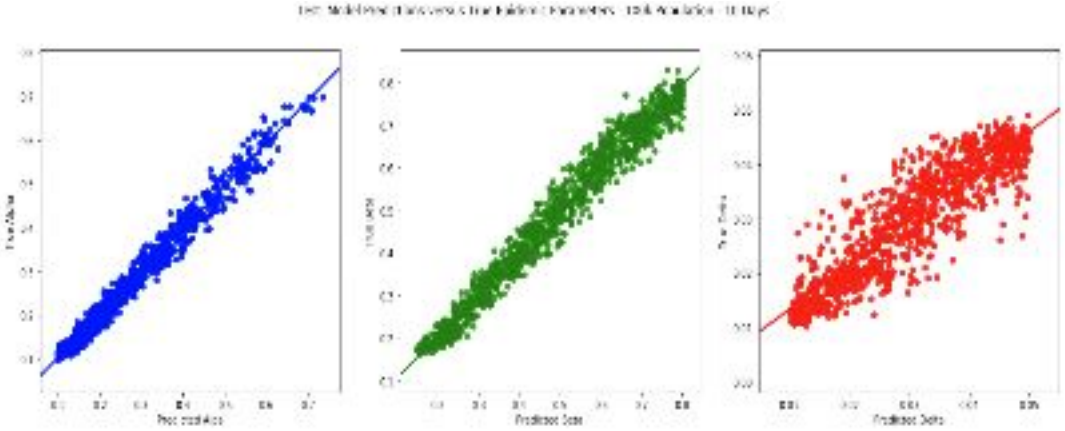
The model was evaluated on a separate test set consisting of feature values of 1,000 simulations along with their corresponding key parameter values. The benchmark proposed to compare this model was simply the naive mean prediction on all three parameters. The mean absolute percentage Error (MAPE) of the **naive model's predictions** on the test set were as follows:

- **MAPE** for parameter **beta** : **45.796**%
- **MAPE** for parameter **alpha** : **54.019**%
- **MAPE** for parameter **delta** : **43.637**%
- **Average MAPE** for **naive model** across the 3 parameters: **47.82**%

The prediction results of our bidirectional RNN on the test set, whose parameters and training are explained in VIII. Model Architecture & Hyperparameters, were as follows:

- **MAPE** for parameter **beta** : **5.891**%
- **MAPE** for parameter **alpha** : **6.202**%
- **MAPE** for parameter **delta** : **14.58**%
- **Average MAPE** for **our model** across the 3 parameters: **8.891**%

A visualization of the bi-directional RNN results, plotting predictions versus real values:



Overall, it can be observed that our model was able to learn the epidemic parameters much better than our benchmark model, obtaining an average MAPE across the three parameters of 8.891% versus the naive which obtained an average MAPE of 47.82% .

In particular our model learned infection β and the recovery rate α of epidemic quite well, both with MPAE for α and β $< 6.5\%$. But it is also observed that our model struggled to learn an epidemics death rate parameter as it obtained an MPAE for $\delta > 14\%$.

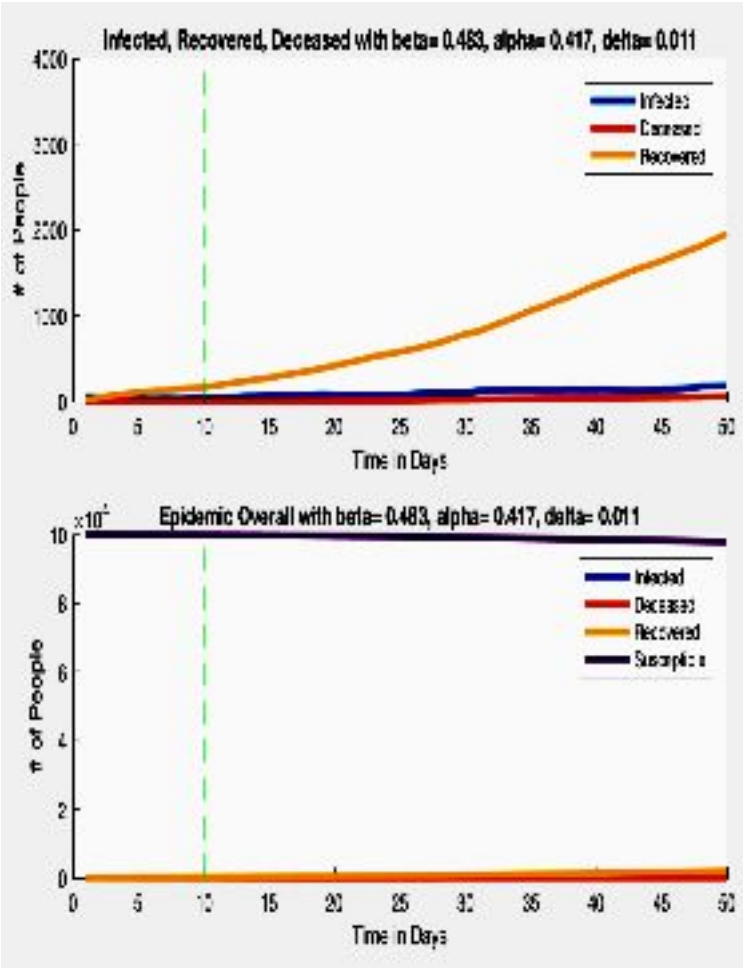
Shown below are **the top ten test observations that the model produced the most error:**

	true_beta	true_alpha	true_delta	pred_beta	pred_alpha	pred_delta	mape_beta	mape_alpha	mape_delta	mape_avg
550	0.48431	0.41722	0.011212	0.546743	0.449602	0.027269	12.891069	7.833248	144.106558	54.943625
662	0.31312	0.20968	0.012983	0.324656	0.228452	0.029060	3.684195	8.949028	123.829475	45.454233
522	0.16525	0.10327	0.018996	0.178708	0.121346	0.037447	9.142801	17.503484	97.132122	40.926069
263	0.47579	0.39600	0.013306	0.440205	0.329971	0.031299	7.479069	10.795938	103.823304	40.699020
95	0.16625	0.11159	0.011677	0.189798	0.118896	0.022878	14.164236	6.574029	95.919914	38.886060

Unsurprisingly, we observe that the lion's share of the error for these observations came from the delta parameter. Interestingly, the highest errors came when the true death rates δ was in its lower range of values [0.01167-0.01899]. Below are the simulation plots contrasting the true and predicted epidemic for the observation which model was obtained the highest error:

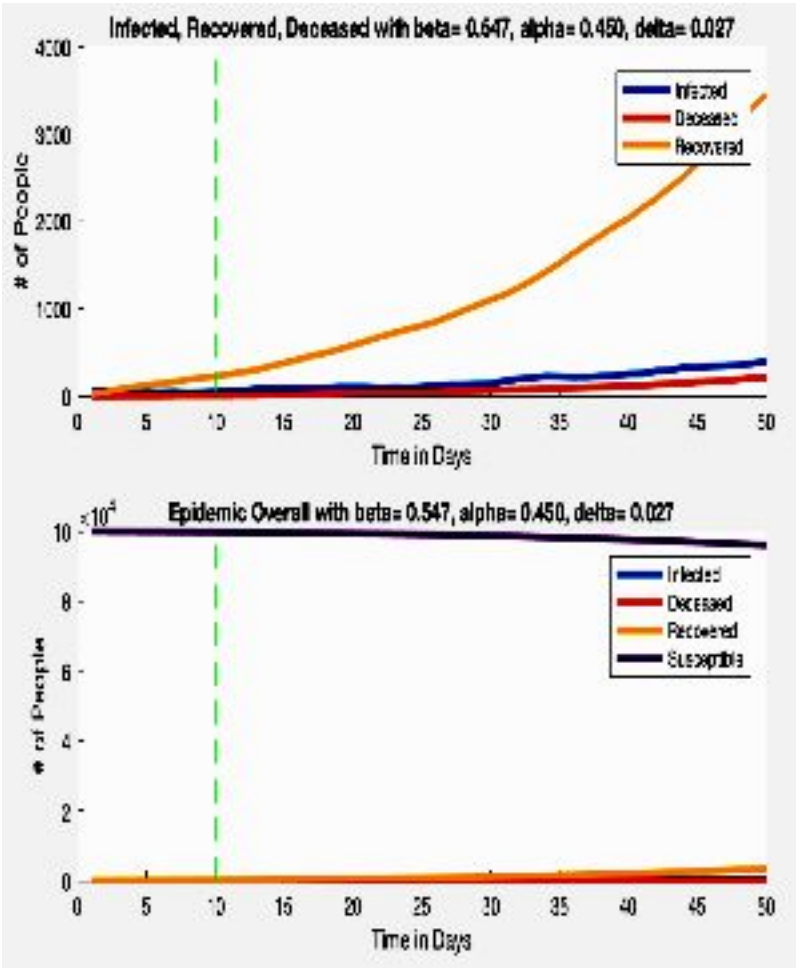
True Epidemic

$\beta = 0.48431;\ \alpha = 0.41722;\ \delta = 0.109$



Predicted Epidemic

$\beta = 0.547;\ \alpha = 0.45;\ \delta = 0.027$



It can be observed that the true epidemic and the model predicted epidemic appear to be slightly dissimilar, particularly after ~15 day mark. But note that the model only received inputs from *within the first 10 days*, delineated by the dashed green line. If we restrict our observation to within the first 10 days, we observe a negligible difference, only a slightly higher number of recovered, between the true epidemic and the model's predicted epidemic.

Due to the restricted time window of 10 days, and that the population of 100k is initialized with a very low number of infected, ~50 sick or .05%, not many deaths were observed within the model's time window. Further our epidemic simulation model is **stochastic**, adding a noise factor to death data - it is possible that there was simply insufficient data or not enough time steps for the model to learn the particular death rate parameter δ. Thus, for slower epidemics with low death rates, the model would thus need a longer time window or data from later steps, to be able to discern the death rate parameter.

Conclusion

X. Free-Form Visualization

The purpose of this project was to design a viable deep learning model that could be trained off artificial data generated from the stochastic epidemic model, that can be feasibly deployed, *due to features that are easily observed and tracked*, to predict the magnitude of an epidemic very early in its process. The real-world ramification of such a model being that it could at least predict a baseline of the reality of an epidemic - a predicted infection, death and recovery rate of the epidemic within 10 days, prior to the observing the deleterious effects of an epidemic so that the epidemic could be handled appropriately to its magnitude.

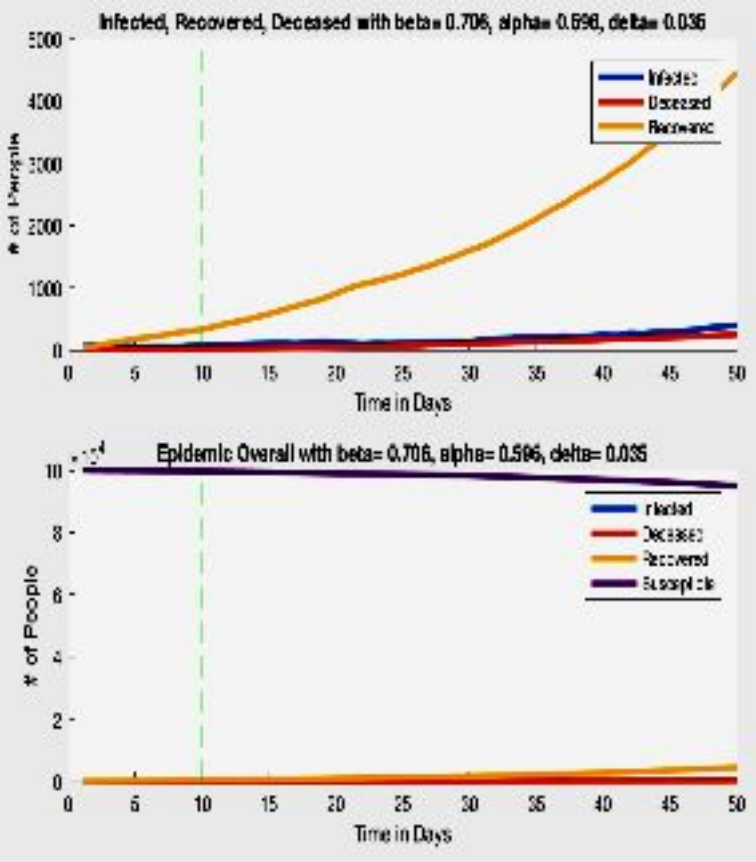
The deep learning model's predicted epidemic parameters could then be followed up with a simulation of the stochastic epidemic model, which would describe the probable behavior of the epidemic on a population - essentially allowing us to know, within 10 days of ~0.05% of the population being sick, approximately what epidemic to expect. It is nice to note that this is also shows that perhaps machine learning and mathematical modeling, quite contrary to being mutually exclusive, can actually be combined powerfully.

As an example, let's sample a random epidemic simulation from the test set:

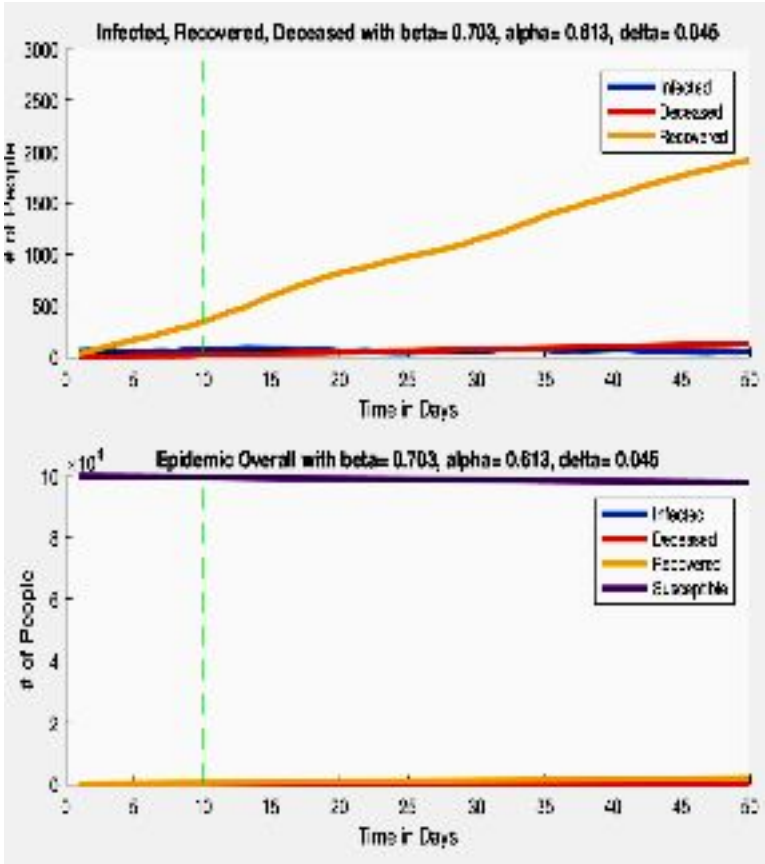
	true_beta	true_alpha	true_delta	pred_beta	pred_alpha	pred_delta	mape_beta	mape_alpha	mape_delta	mape_avg
199	0.70549	0.59647	0.034661	0.703303	0.613272	0.045281	0.30995	2.816888	30.640592	11.25581

And using the true and predicted parameters plot the true epidemic and the predicted epidemic:

True Epidemic



Predicted Epidemic



We observe that the model's predicted epidemic was an underestimation of the true epidemic. In a real case situation, believing such a prediction could lead to insufficient preparations and planning - and thus be catastrophic. At the same time we observe that the true and predicted epidemic within the first 10 days had little discernible difference between them. This shows an important limitation of our model, as the relationship between the epidemic behavior and the key parameters is extremely sensitivity, and slight deviations result in different epidemics altogether.

XI. Improvements & Reflection

Clearly the model can be improved, and should not seen as the final product but rather a step in a direction - of creating an algorithm that can predict an epidemics' parameters and gauge it's magnitude so that the potential damage it would inflict on a population can be mitigated or perhaps even avoided.

One direction for improving the model could be incrementally, through training the model on more simulations/data, better/more features, using layer normalization for overfitting and more. From a practical standpoint, the main weakness of the model is when it is wrong and we do not know it. And when facing a looming epidemic, we would naturally care very much about how uncertain our model is about in its prediction.

When dealing with epidemics, the consequence of assuming wrongly, even a little, would cost lives -- and being very wrong would be disastrous. After going through the process of creating the project and writing this paper, I think rather than incremental improvements of maximum likelihood estimator, such as the one described in this paper, though one can obtain very good results, they fail to relate crucial information about the model's uncertainty in its predictions.

A better approach for highly critical problems such as epidemics, would be to change the type of model altogether - specifically to a Bayesian recurrent neural network. BRNNs can incorporate regularisation through application Bayesian methods to training **[12]**. This is critical, since as was noted section VII. Model Training, a high dropout value was not enough to regularize training and prevent overfitting. But most of all, Bayesian recurrent neural network approach would also allow the model to express its' uncertainty in its prediction through its predicted parameters, which as stated before is *crucial* to know when incorporating an epidemic's predicted parameters as information **[12]**.

To conclude with some non-analytical thoughts, it's exciting to see that there lies the possibility of applying machine learning to potentially help solve one of the biggest threats humanity has faced in its time, epidemics. Being able to predict epidemics and their magnitude would be invaluable for governments, health authorities and individuals to take the necessary actions needed to handle the epidemic and perhaps even stop an epidemic early in its cycle. Predicting accurately the true magnitude of an epidemic is within reach, and that bodes well for humanity.

References:

[1] AIP Conference Proceedings 1891, 020064 (2017) https://doi.org/10.1063/1.5005397

[2] International Journal of Engineering Research in Computer Science and Engineering Volume 4, March 2017, Topic: A Proposal for Epidemic Prediction using Deep Learning

[3] https://publications.polymtl.ca/1659/1/2014_NastaranJafarpourKhameneh.pdf

[4] https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology

[5] Kermack WO, McKendrick AG (August 1, 1927). "A Contribution to the Mathematical Theory of Epidemics". Proceedings of the Royal Society A. 115 (772): 700–721

[6] Allen L.J.S. (2008) An Introduction to Stochastic Epidemic Models. Mathematical Epidemiology. Lecture Notes in Mathematics, vol 1945. Springer, Berlin, Heidelberg

[7] https://en.wikipedia.org/wiki/Power_transform

[8] https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks

[9] Glorot & Bengio, AISTATS 2010: http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

[10] http://ruder.io/optimizing-gradient-descent/

[11] https://en.wikipedia.org/wiki/Mean_absolute_percentage_error

[12] https://arxiv.org/pdf/1704.02798.pdf