Is COVID-19 Worth Paying Attention To?

Using Attention Mechanisms to Understand COVID-19 Mortality Trends

Anil Naidu
University of Toronto
anil.naidu@mail.utoronto.ca

Elaine Sui
University of Toronto
elaine.sui@mail.utoronto.ca

Abstract

The on-going COVID-19 pandemic has been the worst public health crisis of the twenty-first century. As a result, many computer science researchers have created machine learning models to forecast the virus' effect on numerous populations. As the data is collected as a time series, most researchers have attempted to model the dynamics of COVID-19 with variants of recurrent neural networks (RNNs). That said, little research has been done to understand what these networks actually learn. In this paper, we examine RNN variants on COVID-19 time series data and compare these models against attention-based RNNs, which offer interpretability through attention-visualization maps.

1 Introduction

The COVID-19 pandemic has been the most significant public health phenomenon of the twenty-first century. As more people are infected and fatalities rise, public health officials are pressured to implement measures to curb the spread of the virus. However, the effectiveness and severity of such measures depend on accurate forecasting of virus dynamics. Recurrent neural networks (RNNs) have become popular models for short-term time series forecasting. Though performance is relatively high, such models lack the interpretability required for public health officials to benefit from such models beyond considering them as black boxes. Our main contributions are summarized in the following:

- Compare the performance of attentional and non-attentional RNNs on k-day context/condition windows to predict cumulative deaths due to COVID-19 t-days ahead of the window
- Implement an attention mechanism that attends on exogenous time series data, i.e., time series that are not being predicted (e.g. number of tests performed, number of patients in ICU, etc. when the model is trained to predict the number of deaths)
- Present and implement a novel attention mechanism to attend on days in our condition window used for prediction
- Visualize the attentional weights for interpretability

2 Related Work

Recurrent Networks and COVID-19. Recurrent neural networks are advantageous to other neural models in their ability to capture nonlinear temporal relationships. As a result, it has become one of the most popular machine learning tools used in time series forecasting, including the task of forecasting disease dynamics. Pal et al. [4] used a shallow LSTM to predict countries' COVID-19 risk level. Wang et al. [6] compared RNN variants and proposed clustering methods to mitigate the effect of data sparsity when forecasting COVID-19 cases in United States. Arora et al. [1] uses LSTM variants to predict positive COVID-19 cases in India.

Attention Mechanisms. Attention mechanisms allow models to focus on certain input features at each timestep during learning. These mechanisms are typically applied to an Encoder-Decoder (ED) architecture built with RNNs, where attention is used to "select" important features in the encoder to generate a more accurate output from the decoder. In the past decade, attention has been widely popularized in alignment and transduction tasks such as image caption generation [7] and machine translation [3]. More recently, similar architectures have been used in time series analysis. Qin et al. [5] proposes the DA-RNN a modified version of this attention-based ED architecture. Specifically, in addition to the attention layer in the decoder, they add an input attention layer in the encoder that adaptively selects relevant exogenous series to make predictions when multiple such series are available as input. A portion of this paper will use a generalized version of the proposed architecture on COVID-19 time series data and interpret the attention weights.

3 Models

3.1 Attention Mechanisms

3.1.1 Input Attention

Originally proposed by Qin et al. [5], the idea for input attention is to feed an RNN cell with weighted input features at that time step. In other words, given a sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k)$ where $\mathbf{x}_i \in \mathbb{R}^n$ and n is the number of input features, instead of using $\mathbf{x}_i$ as the input for time step i, we use $\overline{\mathbf{x}}_i = \alpha_i^\top \mathbf{x}_i$, where $\alpha_i \in \mathbb{R}^n$ is a vector of non-negative weights that sum to one. Intuitively, the weights should indicate the importance of each input feature on a specific day in our condition window of historical data in order to make the best prediction. We compute these weights based on the entire condition window using (modified) additive or scaled-dot product attention. Refer to Appendix 8.1 for equations.

3.1.2 Decoder Attention

This attention mechanism was originally used in the decoder of Encoder-Decoder architectures for neural machine translation [3]. Its purpose is to attend to encoder hidden states, which are representations of the input sequence, generating a context vector that should help inform the next translated word in the output sequence. Contrary to its original purpose, we use this attention mechanism to allow our model a greater ability to "access" longer-term memory. We compute the attention weights using (modified) additive or scaled-dot product attention. Refer to Appendix 8.2 for equations.

3.1.3 Temporal Attention

This is our "new" attention layer that attends to decoder hidden states. In our case, it provides us interpretability in determining the importance of each day in our k-day condition window for the final t-days ahead prediction. Attention weights are generated by putting the final decoder layer's hidden states into an MLP with a softmax layer. We then compute a weighted sum of our decoder hidden states along the sequence dimension and use that as input to our final fully-connected layer for prediction.

$$\tilde{\alpha_c} = W' \left(\tanh\left(W \tilde{\mathbf{H}} \right) \right) \qquad \alpha_c = softmax(\tilde{\alpha_c}) \qquad h_c = \alpha_c^\top \tilde{\mathbf{H}}$$

where α_c is the weight vector for condition window c, $\tilde{\mathbf{H}}$ is a matrix of the decoder's final layer of hidden states and h_c is the weighted sum of the decoder hidden states. Biases are omitted for brevity.

3.2 Summary of our Models

We will be comparing the performance of different RNN models with three types of attention, namely (encoder) input, decoder and our novel temporal attention. We generalize the typical one-layer attention-based architectures by stacking multiple layers and applying attention at each layer. Unlike Qin et al. [5], we use modified form of additive attention and scaled-dot product attention as our attention layers. Detailed descriptions of the models used and the training pseudocode are found in the Appendix 8.3 and 8.5, respectively.

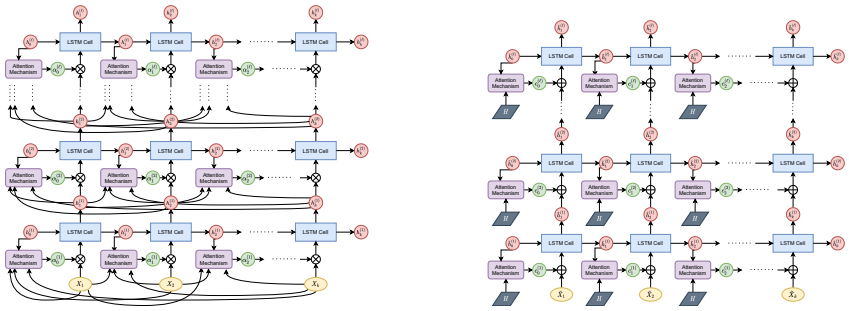

Figure 1: Right: Stacked RNN with Input Attention (Unidirectional); Left: Stacked Decoder with Temporal Attention (H denotes encoder hidden states)

4 Dataset and Setup

We use the **Status of COVID-19 cases in Ontario** [1] which contains new daily case, recovery and death counts, in addition to other counts related to coronavirus testing, hospitalizations and long-term care homes. We only use data from May 19, 2020 onwards. We choose the remaining 2020 data to be our training set, January and February of 2021 to be our validation set, and March onwards to be our test set. At the time of conducting the experiments in this paper, our test set was from March 1, 2021 to April 16, 2021. Refer to Appendix 8.4 for more details. Refer to Appendix 8.6 and 8.7 for hyperparameter tuning details.

5 Experiments, Results and Discussion

5.1 Experiment 1: Attention vs. Non-Attention

We hypothesize that a model with any of the three attention mechanisms will outperform its non-attentional counterpart. We expect this as the use of attention mechanisms reflects the real world where certain reported counts and days in a given condition window are more relevant to understanding and predicting the progression of COVID-19.

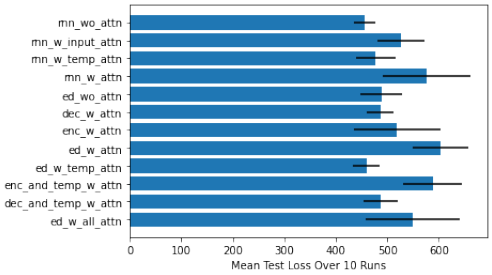

Figure 2: Model performance plot based on MSE

Table 1: Model Test Performances

Model	Test MSE (± std)	Test MAE (± std)
rnn_wo_attn	**455.759 (± 21.034)**	17.366 (± 0.690)
rnn_w_input_attn	527.125 (± 45.620)	17.990 (± 1.030)
rnn_w_temp_attn	476.993 (± 38.556)	17.467 (± 0.696)
rnn_w_attn	577.086 (± 84.850)	18.938 (± 1.621)
ed_wo_attn	488.935 (± 40.316)	**16.969 (± 0.537)**
dec_w_attn	486.171 (± 26.236)	**16.862 (± 0.390)**
enc_w_attn	518.919 (± 84.076)	17.772 (± 1.843)
ed_w_attn	602.922 (± 53.904)	18.191 (± 0.637)
ed_w_temp_attn	**459.423 (± 25.860)**	**16.868 (± 0.661)**
enc_and_temp_w_attn	588.180 (± 56.458)	20.128 (± 1.238)
dec_and_temp_w_attn	487.236 (± 33.441)	17.203 (± 0.726)
ed_w_all_attn	549.194 (± 91.874)	18.857 (± 1.861)

In general, we observe that using the input attention mechanism worsens model performance. We believe this is the case as the effective feature reduction caused by the input attention may make it harder for the other parts of the model to carry out their tasks. This is especially evident when we use other attention mechanisms in addition to the input mechanism, as model performance is worse, on average, than when using the input mechanism alone.

Overall, we find that using the input attention mechanism has a negative effect on performance, while using the decoder and/or temporal attention mechanisms has a negligible or slightly positive effect on performance. Thus, our hypothesis is not entirely true. Rather, the types of attention mechanisms used determine whether the performance of models with attention are better than those without.

[1] https://data.ontario.ca/en/dataset/status-of-covid-19-cases-in-ontario

5.2 Experiment 2: Comparison Between Different Types of Attention Mechanisms

We believe that input attention is more useful than temporal attention which, in turn, is more useful than decoder attention. This is because the initial pruning of the data should simplify the model task and the last attention mechanism has the most immediate effect in generating predictions.

However, we actually observe that temporal attention is most significant, followed by decoder attention, followed by input attention, disproving our original claim. If we examine the performance of each Encoder-Decoder model in Figure 5.1, temporal attention has a greater positive effect than input and decoder attention on model performance. Further, adding decoder attention to temporal attention has a negligible effect, but adding input attention causes a large reduction in performance. Also, the decoder attention alone produces better results than input attention. Thus, the mechanisms applied at the end of the model architecture have a greater positive impact than those at the beginning.

5.3 Experiment 3: Features Attended by Input Attention Mechanism

We expect that if we find a subset of input features with which a non-attentional model performs best, those same features would be attended to by the input attention mechanism. Since it would be intractable to run each non-input-attention model on every combination of input features, we run them on power sets of feature categories. The feature categories that result in the best performance over the models are reported in the second columns of Table 5.3. Refer to Appendix 8.8 and 8.10 for descriptions of features used and our experiment procedure.

Table 2: Non-input attention models and their best feature subsets

Model	Best Input Features	Input-Attended Features
RNN w/o Attn	Testing, Hospital, LTC	TPLTCHCW, Hospital features in general
RNN w/ Temporal Attn	Basic, Testing, Hospital, LTC	TPLTCHCW, Hospital, Positive, ICU
ED w/o Attn	Testing, Hospital, LTC	Fairly equal throughout
ED w/ Decoder Attn	Basic, Testing, Hospital, LTC	Fairly equal throughout
ED w/ Temporal Attn	Testing, Hospital, LTC	TPLTCHCW, ICU, Positive
ED w/ Decoder and Temporal Attn	Basic, Testing, Hospital, LTC	Hospital, ICU, TPLTCHCW, Positive, TTC, UI

The third columns of Table 5.3 describes the features the input-attention mechanism attends to most highly when such a mechanism is added to the corresponding model in column 1. Refer to Appendix 8.10 and 8.11 for how these were determined. Comparing columns two and three in Table 5.3, we see that for the most part, there is at least some correspondence between the best input features for a non-attentional model and the features attended by its attentional counterpart. Thus, this shows that overall, our input attention mechanism is able to find the input features that are the best predictors for cumulative deaths.

5.4 Experiment 4: Days Attended by Temporal Attention

We believe that the temporal attention mechanism will attend to more recent days in the condition window, as trends in more recent data are generally more predictive of what will happen in the immediate future (in our case, one week ahead). We conduct this experiment by comparing the attention maps for the temporal attention mechanism. Refer to Appendix 8.12 for the attention maps.

From the visualizations, we see that more recent days in condition windows are attended to by all models with temporal attention except the RNN with temporal attention and the Encoder-Decoder with temporal attention. These models are the only ones using no other attention mechanism besides the temporal one. This suggests that these mechanisms interact with each other and their interactions dictate which areas are focused on. That said, overall, our attention maps point to recency being an important aspect in formulating more accurate predictions, as hypothesized.

6 Conclusion

In summary, we investigated the effects and interpretations of different attention mechanisms on various RNN models for predicting Ontario COVID-19 deaths 7 days ahead. We found that overall, using temporal and decoder attention mechanisms have negligible or slightly positive effects, while using the input attention mechanism has a negative effect on performance. Though we have attempted to outline reasons as to why this might be the case, more research must be performed to examine its underlying causes, especially in the interactions among attention mechanisms.

References

[1] P. Arora, H. Kumar, and B. K. Panigrahi. Prediction and analysis of covid-19 positive cases using deep learning models: A descriptive case study of india. *Chaos, Solitons Fractals*, 139(110017), 2020. doi: 10.1016/j.chaos.2020.110017.

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*. ICLR, 2015. arXiv:1412.6980.

[3] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, page 1412–1421. Association for Computational Linguistics, 2015. doi: 10.18653/v1/D15-1166.

[4] R. Pal, A. A. Sekh, S. Kar, and D. K. Prasad. Neural network based country wise risk prediction of covid-19. *Applied Sciences*, 10(18), 2020. arXiv:2004.00959.

[5] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, page 2627–2633, 2017. https://www.ijcai.org/proceedings/2017/0366.pdf.

[6] L. Wang, A. Adiga, S. Venkatramanan, J. Chen, B. Lewis, and M. Marathe. Examining deep learning models with multiple data sources for covid-19 forecasting. arXiv:2010.14491, 2020.

[7] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2048–2057. PMLR, 2015. http://proceedings.mlr.press/v37/xuc15.pdf.

7 Contributions

Contributor	Coding	Paper
Anil	• Attempted training code and attention visualization code • Additive addition (modified from PA3) • Encoder-Decoder models hyperparameter tuning • Initial Basic RNN model	• Experiments 1 and 2 • Conclusion • Overall editing • Proposal Method/Algorithm and Summary
Elaine	• Data preprocessing • RNN with input attention; all Encoder-Decoder models • Scaled-dot product attention (modified from PA3) • Final training code and attention visualization code • RNN hyperparameter tuning • Experiment code; Plotting code	• Abstract, Introduction, Related Work, Models, Methods, References • Experiments 3 and 4 • All diagrams, pseudocode, equations and tables

8 Appendix

8.1 Input Attention Equations

Additive Input Attention:

$$\tilde{\alpha}_l^{(t)} = W_{l,2}(ReLU(W_{l,1}[\mathbf{H}_l^{(t-1)}; \mathbf{X}_l])$$
$$\alpha_l^{(t)} = softmax(\tilde{\alpha}_l^{(t)})$$

Scaled-Dot Input Attention:

$$\tilde{\alpha}_l^{(t)} = W_{p,l}\Big(\frac{(W_{q,l}\mathbf{H}_l^{(t-1)})^\top (W_{k,l}\mathbf{X}_l)}{\sqrt{k}}\Big)$$
$$\alpha_l^{(t)} = softmax(\tilde{\alpha}_l^{(t)})$$

where $\alpha_l^{(t)}$ is the weight vector for timestep t of layer l, k is the sequence length, $\mathbf{H}_l^{(t-1)}$ is the previous encoder hidden state in layer l repeated k times and $\mathbf{X}_l$ is the encoder input to layer l. Note that the two aforementioned matrices are of size $(m \times k)$, where m is the input size if $l = 1$ and the hidden size otherwise. Biases are omitted for brevity.

8.2 Decoder Attention Equations

Additive Decoder Attention:

$$\tilde{\alpha}_l^{(t)} = W_{l,2}(ReLU(W_{l,1}[\tilde{\mathbf{h}}_l^{(t-1)}; \mathbf{H}])$$
$$\alpha_l^{(t)} = softmax(\tilde{\alpha}_l^{(t)})$$
$$c_l^{(t)} = (\alpha_l^{(t)})^\top \mathbf{H}$$

Scaled-Dot Decoder Attention:

$$\tilde{\alpha}_l^{(t)} = \frac{(W_{q,l}\tilde{\mathbf{h}}_l^{(t-1)})^\top (W_{k,l}\mathbf{H})}{\sqrt{d}}$$
$$\alpha_l^{(t)} = softmax(\tilde{\alpha}_l^{(t)})$$
$$c_l^{(t)} = (\alpha_l^{(t)})^\top W_{v,l}\mathbf{H}$$

where $\alpha_l^{(t)}$ is the weight vector for timestep t of layer l, $\tilde{\mathbf{h}}_l^{(t-1)}$ is the previous decoder hidden state in layer l, $\mathbf{H}$ is a matrix of encoder's final layer of hidden states and d is the decoder hidden size. Biases are omitted for brevity.

8.3 Model Names and Their Descriptions

Model Name	Description
rnn_wo_attn	RNN without any attention
rnn_w_input_attn	RNN with input attention only
rnn_w_temp_attn	RNN with temporal attention only
rnn_w_attn	RNN with both input and temporal attention
ed_wo_attn	Encoder-Decoder without any attention
dec_w_attn	Encoder-Decoder with decoder attention only
enc_w_attn	Encoder-Decoder with input attention only
ed_w_attn	Encoder-Decoder with input and decoder attention only
ed_w_temp_attn	Encoder-Decoder with temporal attention only
enc_and_temp_w_attn	Encoder-Decoder with input and temporal attention only
dec_and_temp_w_attn	Encoder-Decoder with decoder and temporal attention only
ed_w_all_attn	Encoder-Decoder with input, decoder and temporal attention

8.4 Note on data pre-processing

Some of the features in the Ontario COVID-19 dataset were deemed as legacy features. These are features that the government reported only when a pandemic was not officially declared by the WHO. As a result, we do not use these as possible input or output features. We also linearly impute any missing values and smooth the data by using a moving 3-day average. We only use data from May 19, 2020 onwards, as all the non-legacy statistics are reported consistenly thereafter. We further choose to divide our dataset into training, validation and test by date in order to prevent the model to unintentionally learn temporal dependencies that would not generalize. Specifically, our training set includes data from May 19, 2020 to December 31, 2020, our validation set includes data from January 1, 2021 to February 28, 2021 and our test set includes data from March 1, 2021 onwards. This gives a roughly 67/17/16 split of our data.

8.5 Training Procedure

Algorithm 1 Training

model = RNN or Encoder-Decoder
for $epoch = 1, 2, \ldots, n$ **do**
  for $\mathbf{X}$, $\tilde{\mathbf{X}}$, $\mathbf{y}$ in DataLoader **do**
    # $\mathbf{X}$ *is condition window of input features*, $\tilde{\mathbf{X}}$ i*s condition window of output features*
    # $\mathbf{y}$ *is target*
    $\mathbf{X} = \mathbf{X}$.normalize
    $\hat{\mathbf{y}}$ = model($\mathbf{X}$)
    $\hat{\mathbf{y}}$ = $\tilde{\mathbf{X}}$.mean + $\hat{\mathbf{y}} \cdot \tilde{\mathbf{X}}$.std
    loss = MSELoss($\hat{\mathbf{y}}, \mathbf{y}$)
    θ = gradient(loss)
    update(model.params, θ)
  end for
  if validation loss increases **then**
    lr = lr · lr_decay
  end if
  if validation loss does not decrease for m consecutive epochs **then**
    break
  end if
end for
return best model

8.6 Hyperparameter Tuning and Training Procedure

We tune our hyperparameters while predicting $t = 7$ days and fixing our input features to all available ones and our output feature to cumulative deaths. We use a grid search to find the optimal parameters for each model, which includes the optimal condition window size of $k = 5$ for encoder-decoder models (we keep $k = 5$ for RNN models as well, to compare the models fairly).

We train our model using minibatch gradient descent with the ADAM optimizer [2] and mean-squared error as our objective function. As the scale of our inputs vary widely, we also normalize each minibatch of data along the sequence dimension before inputting the data into the model and "unnormalize" the outputs. We evaluate the performance of our models by the RMSE and the MAE. Everything is run on the CPU and GPU on Google Colab.

8.7 Hyperparameter Settings

Model	Hidden Size	Num Layers	lr	lr Decay	Batch Size	Num Epochs	Early Stopping Patience	Input Attn	Decoder Attn
rnn_wo_attn	16	1	1e-3	0.999	16	70	5		
rnn_w_input_attn	16	1	1e-3	0.999	16	100	5	Additive	
rnn_w_temp_attn	16	1	1e-3	0.999	32	100	5		
rnn_w_attn	16	1	1e-3	0.999	16	100	5	Additive	
ed_wo_attn	20	1	1e-3	0.8	22	300	20		
enc_w_attn	20	1	1e-3	0.97	32	200	20	Additive	
dec_w_attn	26	1	1e-3	0.8	32	200	20		Additive
ed_w_attn	16	1	1e-3	0.95	32	200	10	Additive	Scaled-Dot
ed_w_temp_attn	16	1	1e-3	0.85	32	100	20		
enc_and_temp_w_attn	48	1	1e-3	0.925	10	100	20	Additive	
dec_and_temp_w_attn	18	1	1e-3	0.7	12	100	20		Additive
ed_w_all_attn	32	1	1e-3	0.99	32	150	10	Additive	Additive

8.8 Input Features Included In Each Category

Category	Abbreviation	Feature Description
Basic	Positive	Number of active confirmed cases on the reported date
Basic	Total Cases	Cumulative number of confirmed cases on the reported
Basic	Resolved	Cumulative number of resolved cases on the reported date
Basic	Deaths	Cumulative number of fatalities
Testing	AFT	Total patients approved for testing as of Reporting Date
Testing	TTC	Total tests completed in the last day
Testing	PPT	Percent positive tests in last day
Testing	UI	Number of tests under investigation
Hospital	Hospital	Number of patients hospitalized with COVID-19
Hospital	ICU	Number of patients in ICU due to COVID-19
Hospital	Ventilator	Number of patients in ICU on a ventilator due to COVID-19
LTC	TPLTCR	Total Positive LTC Resident Cases
LTC	TPLTCHCW	Total Positive LTC Health Care Worker (HCW) Cases
LTC	TLCTRD	Total LTC Resident Deaths
LTC	TLTCHCWD	Total LTC HCW Deaths

8.9 Actual vs. Predicted Cumulative Death Plots

The following are the true and predicted cumulative deaths for our four best performing models. The plots are split into the training, validation and test sets.

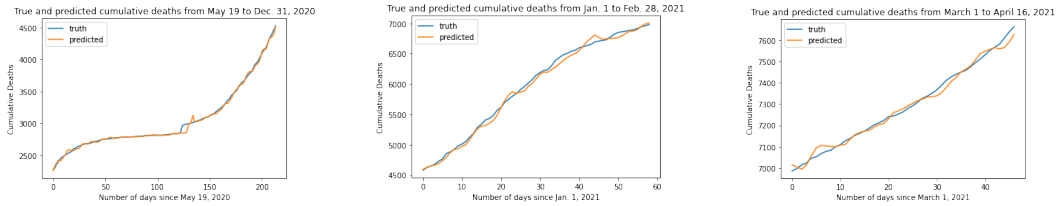

Figure 3: Encoder-Decoder With Temporal Attention

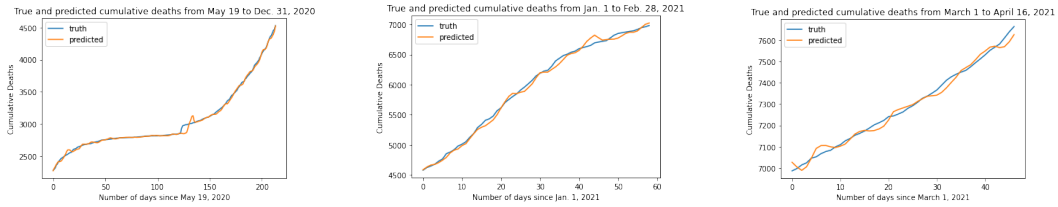

Figure 4: RNN Without Attention

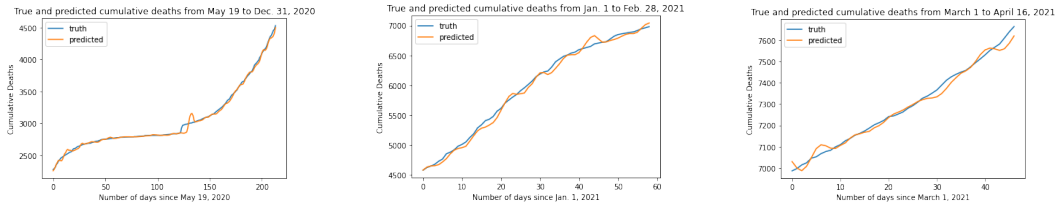

Figure 5: Encoder-Decoder With Decoder Attention

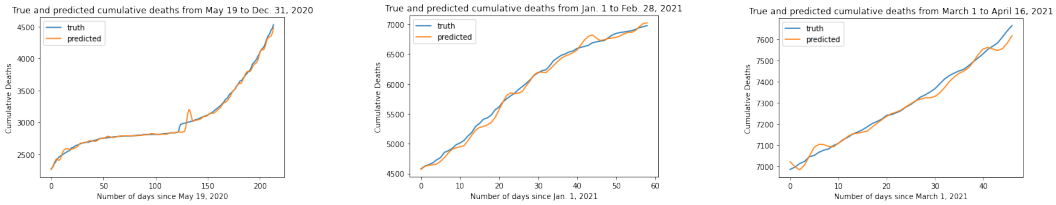

Figure 6: Encoder-Decoder Without Attention

8.10 Experiment 3 Preliminary Note and Performance Plots

We expect that if we find a subset of features that a non-attention model performs best on, those same features would be attended to by the input attention mechanism. This is because a higher performance of the non-attention model means that those features are more important in predicting cumulative deaths. As a result, the input attention mechanism should weight those same features more heavily than others. We conduct this experiment by first finding the best feature subsets for each model without input attention and then visualizing their with-attention counterparts' attention maps.

The following are plots of the means and standard deviations across five runs for each model and each power set of these feature categories.

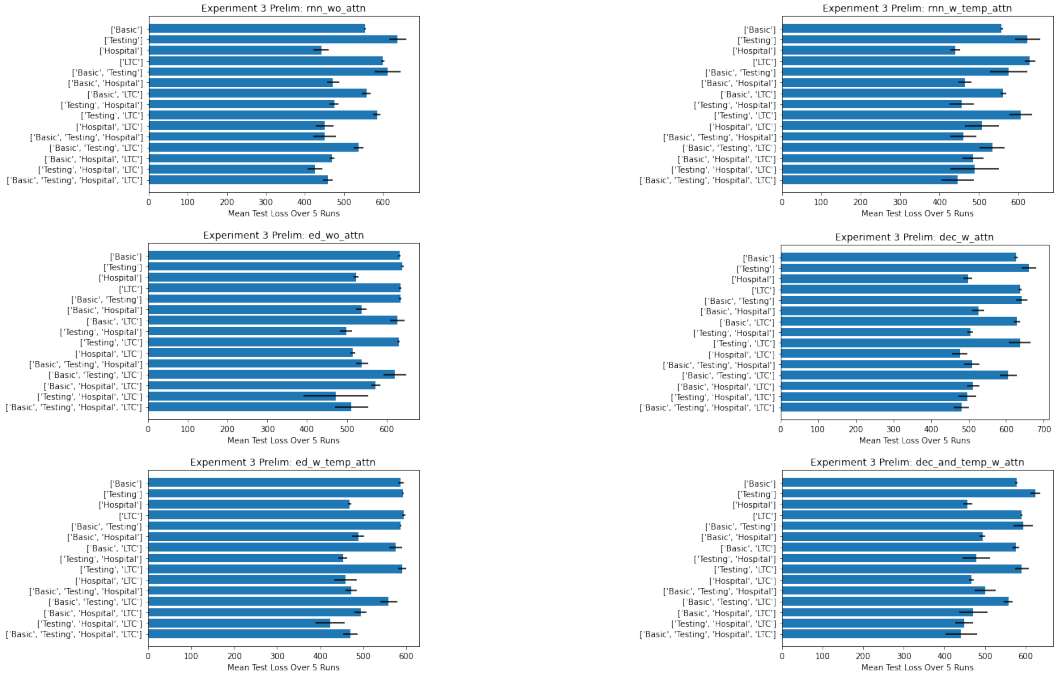

Figure 7: Model Performance on Different Input Feature Categories

8.11 Experiment 3 Input Attention Maps

The findings in 5.3 are from observations of the forward and backward input attention weights for the corresponding models shown below. We visualize these for three days, each in different stages of the pandemic: June 6, 2020, November 8, 2020 and April 7, 2021. Note: low weight = dark blue; high weight = yellow.

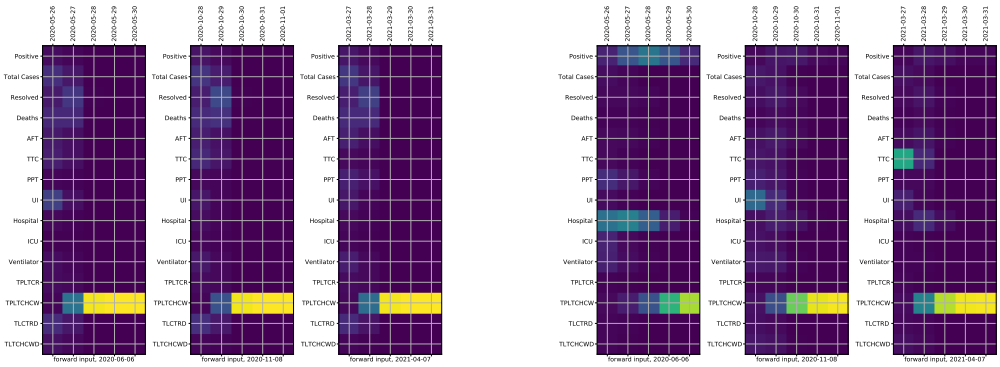

Figure 8: Left: Forward rnn_w_input_attn; Right: Forward rnn_w_attn

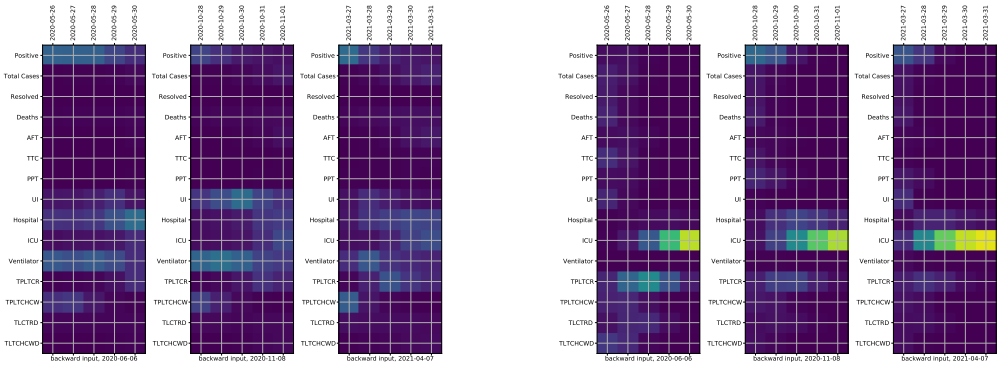

Figure 9: Left: Backward rnn_w_input_attn; Right: Backward rnn_w_attn

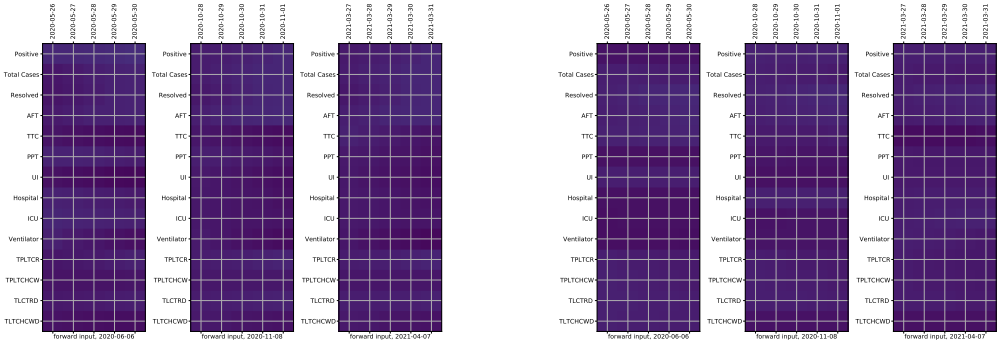

Figure 10: Left: Forward enc_w_input_attn; Right: Forward ed_w_attn

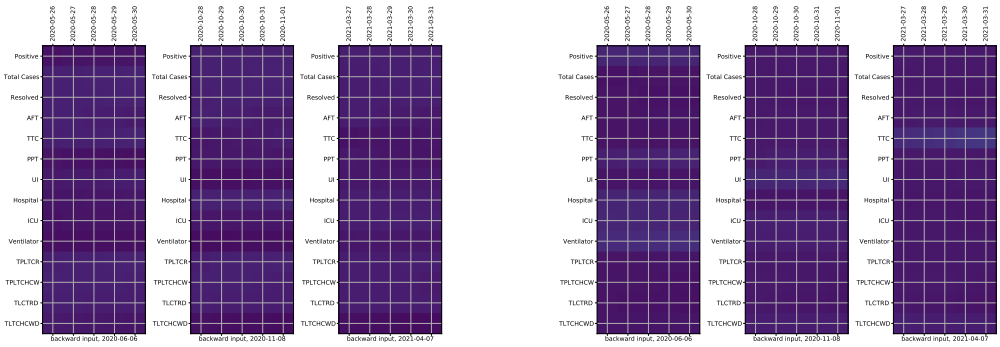

Figure 11: Left: Backward enc_w_input_attn; Right: Backward ed_w_attn

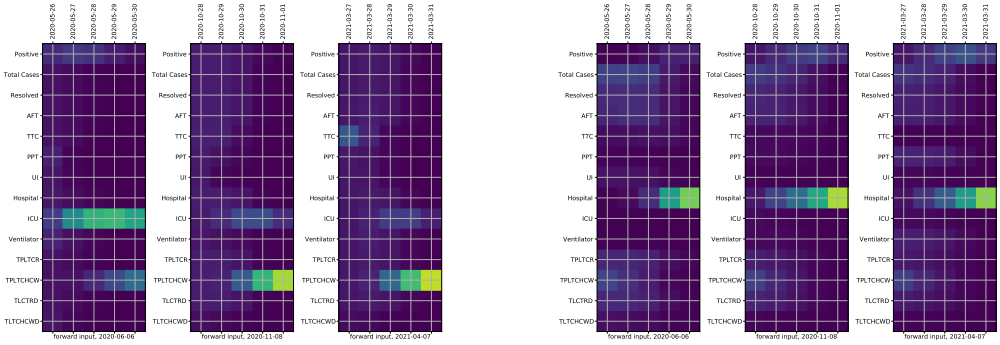

Figure 12: Left: Forward enc_and_temp_w_attn; Right: Forward ed_w_all_attn

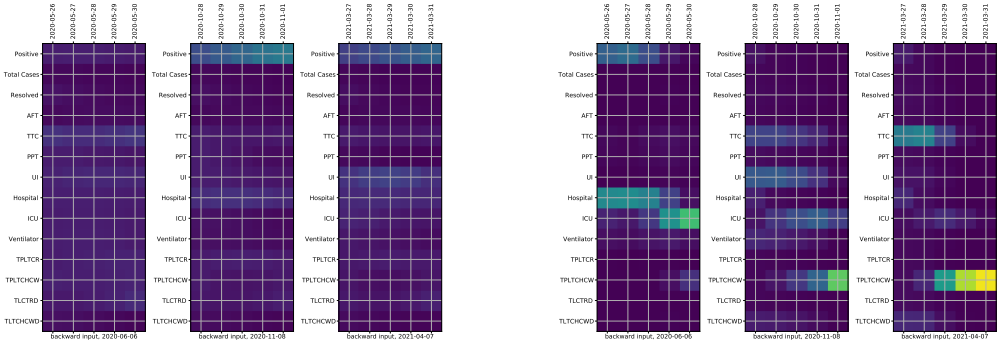

Figure 13: Left: Backward enc_and_temp_w_attn; Right: Backward ed_w_all_attn

8.12 Experiment 4 Temporal Attention Maps

The following are the temporal attention visualization maps for all the models using the temporal attention mechanism. We visualize these for three days, each in different stages of the pandemic: June 6, 2020, November 8, 2020 and April 7, 2021. Note: low weight = dark blue; high weight = yellow.

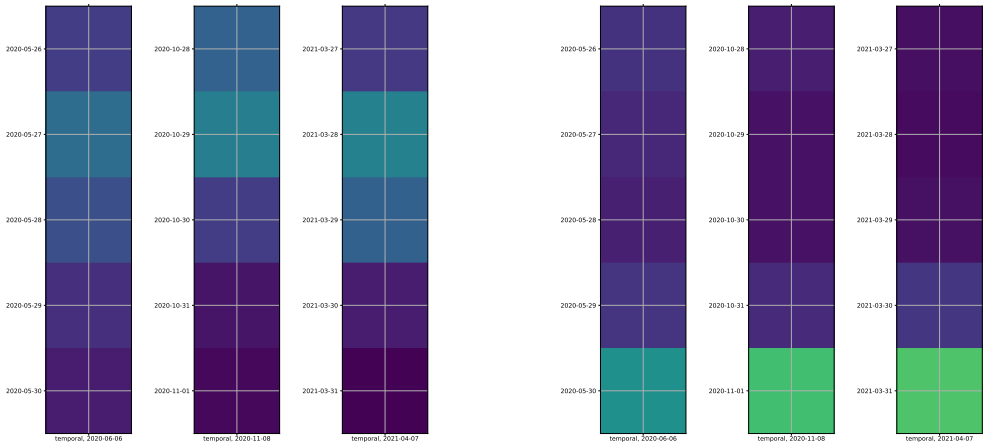

Figure 14: Left: rnn_w_temp_attn; Right: rnn_w_attn

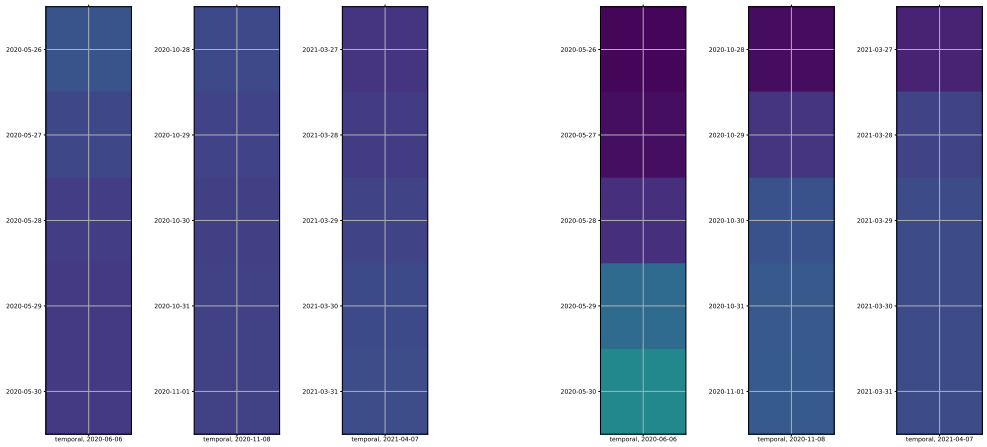

Figure 15: Left: ed_w_temp_attn; Right: enc_and_temp_w_attn

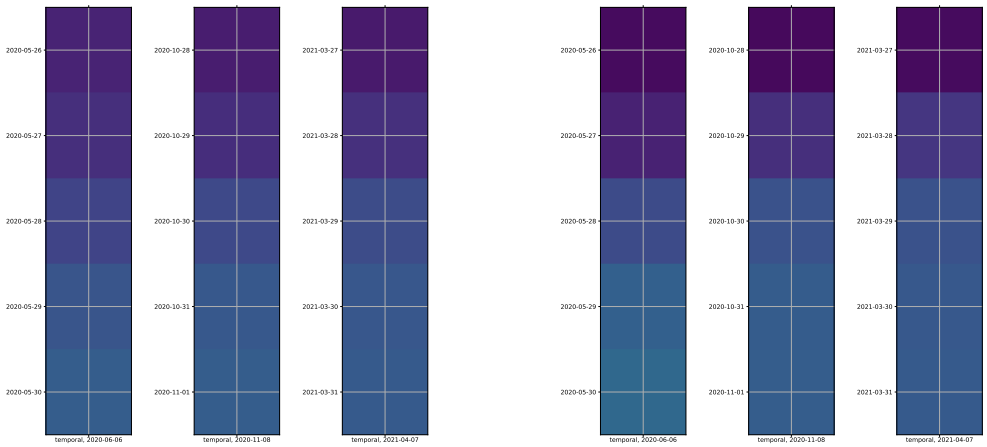

Figure 16: Left: dec_and_temp_w_attn; Right: ed_w_all_attn