Attention LSTMs in Multimodal Models: A holistic approach to predicting COVID infection trends

Nuo Wen Lei

7/29/2022
Inspirit AI Research Project

Abstract

COVID-19 and its infection trends are a hot topic in the AI space due to the seeming applicability of AI, which could help with case preventions and medical resource allocations. However, too often people focus on a predetermined area and scale, overlooking the fact that COVID doesn't flow into or out of just one area but instead spreads everywhere simultaneously. Thus, we propose novel approaches to process the infection rates of all states in the United States as a whole while taking into account the interconnectivity between states through domestic flights. We introduce 1) the graph attention LSTM for analyzing domestic flights across states, 2) the image attention LSTM for understanding infection rates from United States COVID heat maps, and 3) the attention bottleneck mid fusion model that integrates graph and image attention LSTMs. Through tests that require simultaneous predictions of the infection rates in all US states, we compare the performances of our methods with those of classic sequence-processing methods and achieve significantly more accurate results.

1. Introduction

Ever since the start of the COVID-19 pandemic in early 2020, many people have attempted to use machine learning and deep learning techniques to predict infection and death trends surrounding COVID to varying degrees of success. These models often employ LSTMs (Long Short Term Memory)[4] or other kinds of recurrent neural network techniques to process the time series COVID data of a determined area and scale (Ex: United States country level). However, while these models can reach precise predictions of infection and death rates on their predetermined area and scale, these predicted rates provide little information in terms of how the infection rates of that area affect those around it because the models' predictions are restricted to a certain area. Therefore, this paper proposes a holistic approach to tackling COVID infection rate predictions. Using daily aggregated data on domestic air travel and COVID infection rates in each state, we attempt to simultaneously model all state-level infection rates in the United States.

The domestic air travel data is provided by the Bureau of Transportation Statistics, which includes details on domestic flights in the United States from January 1st, 2020 to March 31st, 2022. The COVID infection rate data is produced from a continued effort by Johns Hopkins Center for System Science and Engineering (CSSE) to publish detailed daily US COVID infection rate data. After collecting the data, we decided to format domestic air travel data as directed graphs with states as nodes and number of flights as weighted directed edges and transform COVID infection rates in states as heat maps for visual effect. The heat maps are scaled based on the highest and lowest infection rates across all dates.

Having both graph and visual data, we expand our experiments to include 6 separate models: 1) the first model is a normal LSTM model that takes the numeric COVID infection rate data as input; 2) the second model is a convolutional LSTM network[10] the implementation of which we borrowed from Keras[5] and which takes inputs in the format of COVID infection rate heatmaps; 3) the third model is a combination of models 1 and 2 by concatenating the outputs of the two LSTM models and using a dense layer at the end to make predictions, taking both graph and image data as input; 4) the fourth model is a LSTM implementation of the graph attention layer in *Graph Attention Networks*[8] which takes domestic air travel graphs as input; 5) the fifth model is a LSTM implementation of the image attention layer in *Image Transformer*[7] which takes COVID heat maps as input; and 6) the sixth model is a multimodal combination of models 4 and 5 using attention mid-fusion techniques from *Attention Bottlenecks for Multimodal Fusion*[2], which takes both the domestic flight graph and COVID infection rate heat maps as input.

The main comparison to look out for in these experiments is the difference in results between attention-based LSTM and other LSTM models. For the attention-based LSTM models, attention is often used in the context of language or sequence processing. Thus introducing yet another sequence dimension as time in the input and an extra sequence processing dimension with the LSTM structure is a new technique that's hardly explored. For our results, the graph attention LSTM model outperforms the normal LSTM model with a 35% decrease in mean absolute error (MAE), the image attention LSTM model outperforms the convolutional LSTM model with a 68% decrease in MAE, and the attention bottleneck mid fusion model outperforms the late fusion model with a 94% decrease in MAE.

2. Dataset

We use 2 free and openly available datasets taken from online sources to train our models. The first dataset is one of many COVID datasets maintained by CSSE, containing daily COVID-related statistics for all counties in the United States. The second dataset is web-scraped from the Detailed Statistics Departure page provided by the Bureau of Transportation Statistics, which contains data about domestic flights. Since both datasets are free and available online, we believe that these datasets are most realistically applicable for real-world usages.

2.1 Features

In the COVID dataset provided by CSSE, the statistics for each day are provided in separate files, each including country name, state name, county name, longitude, latitude, county-level population, county-level number of COVID infections, county-level number of COVID-related deaths, and several other geographical indicators. Given our goal to create and interpret sequential US map data on the state-level, we selected the following features: state name, county-level population, county-level number of COVID infections, and county-level number of COVID-related deaths.

In the domestic flight dataset created from the Bureau of Transportation Statistics, some features included airline, airport origin, airport destination, and several other flight statistics like scheduled departure time, delays, elapsed time, etc. Since we wanted to present domestic flights as graphs to best demonstrate the connections between different airports and states, we only selected the relevant features, which include airport origin, airport destination, and date. However, due to delays in flight statistic updates, data is often 2-3 months behind.

2.2 Map Data Processing

After selecting our COVID dataset features, we aggregated the county-level statistics by state. Then by dividing the state-level number of infections and COVID-related deaths with the state population, we created two new features, state-level infection rate and death rate. Since most of the models that we are comparing will be using image data that represent these statistics, we decided to create images of the US map divided by states that are color-mapped based on either infection or death rate.

To create these images, we joined our CSSE COVID dataset with US states geolocational data provided by the maps package[9] in R. Then using the ggplot2[6] R package, we plotted the infection and death rates for every day. These map images are then aggregated and saved as numpy arrays for storage ease.

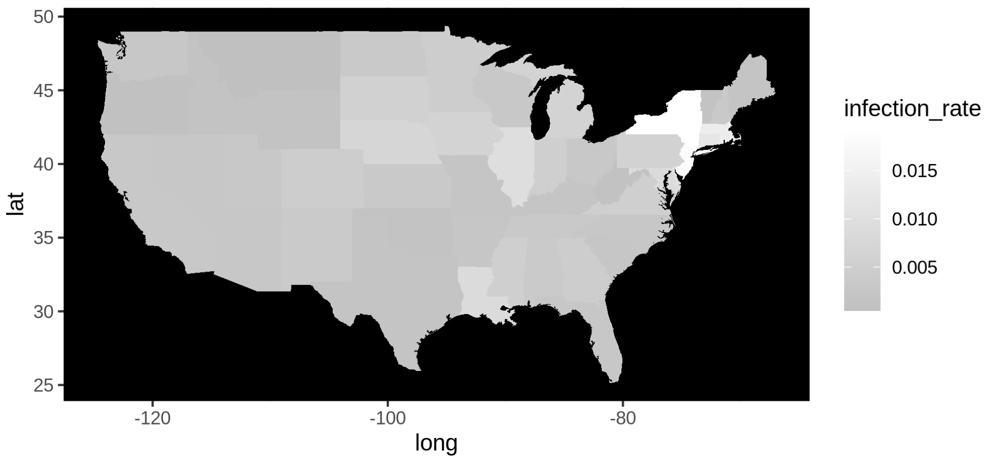

Figure 1: United States state-level heat map of COVID infection rates on May 31, 2020.

2.3 Graph Data Processing

Using the features collected from the domestic flights data, we aggregated the number of flights by origin, destination, and date. Then for every date, we used the NetworkX python package[1] to create graphs with airport state origins and destinations as nodes, flights as edges, and number of flights as edge weights. We then saved these networks as adjacency matrices, which will be the desired input format for the graph attention network.

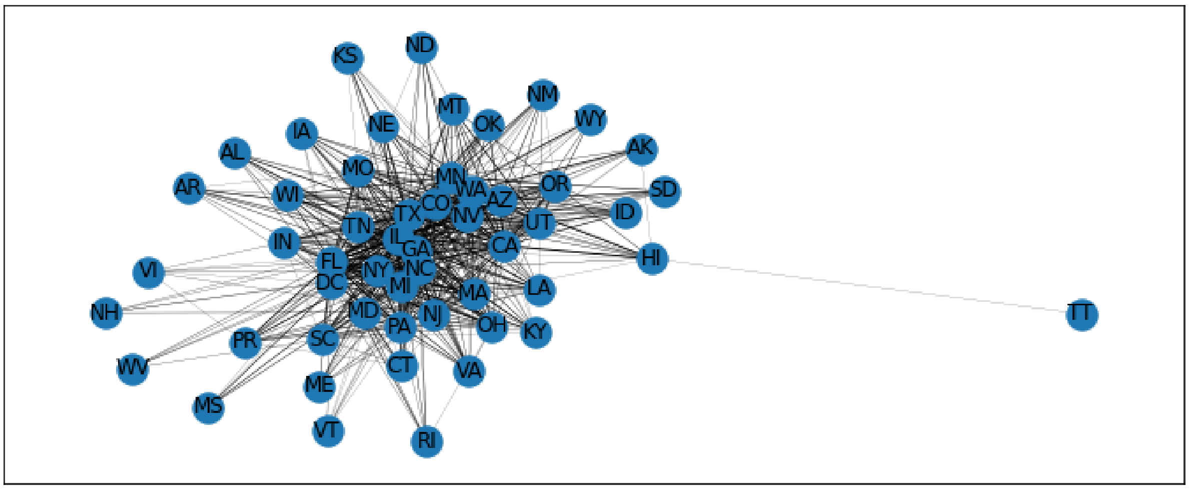

Figure 2: Graph visualization of domestic flight density between states on December 31, 2020. Line thickness represent flight density between specific states and the general closeness of nodes represent the interconnectivity of all US states. This figure is generated using NetworkX's spring layout function[1].

2.4 Sequential Data Processing

Since all models being tested are sequential models, their expected inputs all require an extra sequence dimension. Therefore, we preprocessed all data, image and graph, and made every sample contain the extra sequence dimension. We decided to make every sequence have a length of 7 to represent a week of data, and we assigned the state-level infection rates on the day after the sequence to be the target value. By adding the sequence to our data while preprocessing, it made splitting and serving data afterwards easier. For splitting training and testing data, we decided on a 80%/20% random split, which helps mix up the dates of samples.

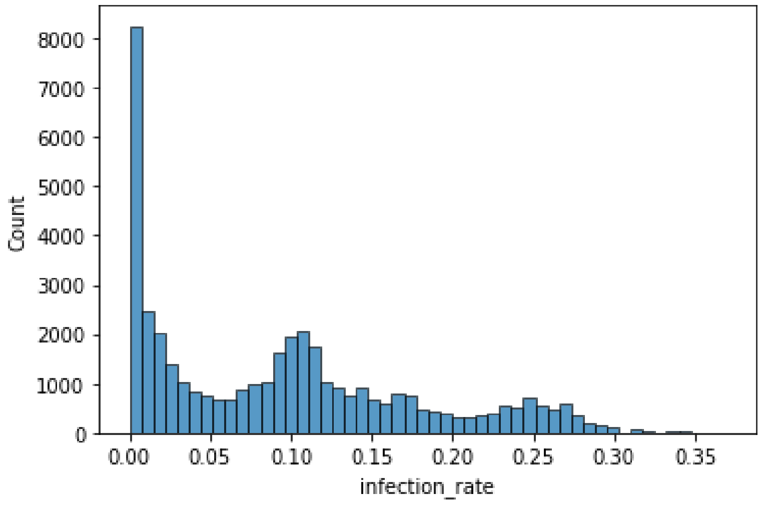

Figure 3: Histogram describing the distribution of infection rates throughout all states and days between January 22, 2020 and May 27, 2022.

2.5 Metrics

We decided to use the same metrics for both the training and evaluation process. We thought that mean squared error and mean absolute error were good metrics to start with, but we found that since we are simultaneously predicting many elements, there should be a component in our loss function that makes sure the overall magnitude is changing with the samples too. Therefore, as shown in Eq. 1, our revised loss function has a new component called squared mean error that is simply added to our mean squared error to create the final loss. Squared mean error simply finds the squared difference of the truth and prediction averages to get a value that shows the difference in magnitude between truth and prediction.

$$loss = \sum_{b=1}^{B}((true_b - pred_b)^2 + (\frac{\sum_{i=1}^{49}(true_{bi} - pred_{bi})}{49})^2)$$

Equation 1: Loss function for all experiments that adds squared mean error with mean squared error where B is the batch size.

3. Models

The experimented six models are the following: 1) LSTM model that processes COVID tabular data, 2) Convolutional LSTM model that processes infection rate heat map images, 3) multimodal model that combines encoded vectors models 1 and 2 to predict with a final dense layer, 4) Graph Attention unit as an LSTM model that processes COVID tabular data and flight paths as graph nodes and edges, 5) Image Attention unit as an LSTM model that processes infection rate heat map images, 6) Attention Bottleneck model that combines models 4 and 5 and applies attention bottleneck technique on resulting encoded vectors to make predictions.

In order to maintain some level of consistency between models, we elected a consistent structure of 3 model layers each with either 16 recurrent cells or 32 neuron units. For multi head mechanisms, we decided to use 8 heads for all such structures. We also used the same training parameters like epochs, batch size, optimizer, and loss function. We decided to uniformly train 20 epochs with a batch size of 16 using the Adam optimizer and mean squared error as the loss function.

3.1 LSTM

LSTMs are designed to properly “remember” and take into account data from earlier parts of sequences. Using the tanh function, only the essence of the “short term” hidden state context is stored in the “long term” cell state at every step of the sequence, allowing the LSTM cell to remember earlier steps more clearly than a traditional RNN (Recurrent Neural

Network) cell. Due to its consistency in storing relevant context throughout the steps of sequential data, we decided to use an LSTM model as our fundamental baseline model.

$$
\begin{aligned}
&i = \sigma(W_{ii}x + b_{ii} + W_{hi}h + b_{hi}) \\
&f = \sigma(W_{if}x + b_{if} + W_{hf}h + b_{hf}) \\
&g = tanh(W_{ig}x + b_{ig} + W_{hg}h + b_{hg}) \\
&o = \sigma(W_{io}x + b_{io} + W_{ho}h + b_{ho}) \\
&c' = f * c + i * g \\
&h' = o * tanh(c')
\end{aligned}
$$

Equation 2: LSTM cell equations for updating hidden and cell states where x is input, h is hidden state, c is cell state, and W is a linear transformation matrix.

The LSTM model takes in the COVID statistics as tabular data that has been formatted to contain a 7-day sequence of data in every sample. In the model structure, the first 2 LSTM layers return the sequence form of their encoded results so that the following LSTM layer could process the input as a sequence. The last layer however returns only the last hidden state of the last LSTM cell, making sure that the returned tensor is only of rank 2 (including batch size). Thus, the resulting tensor could be inputted to a final regression dense layer to predict a vector of length 49 with each element representing the predicted infection rate of a separate US state. For every LSTM layer, we decided to add leaky ReLU for the activation function and layer normalization to scale the outputs.

3.2 Convolutional LSTM

Convolutional LSTMs are a combination of convolutional layers and LSTM cells. The essence of this combination is that the hidden and cell states of an LSTM cell are image data instead of tabular. In changing the data from tabular to image, the size of the hidden and cell states are changed to be images and the processing before the tanh function to update the cell state becomes convolutional image processing, thus creating the combination of convolution and LSTM.

The convolutional LSTM's 2-dimensional implementation from Keras[5], ConvLSTM2D, takes a tensor of rank 5. The first dimension is batch size, second is sequence length, and the rest are the height, width, and channels of the image. In every cell, the input and hidden states all have dimensions of an image. Both the input and hidden state are put into convolutional functions with different kernels, replacing the linear transformations used in a normal LSTM cell. Then, the resulting tensors are put through the same hidden and cell state equations as a normal LSTM. Looking at the LSTM equations in Eq. 2, this change would mean switching all linear transformation matrices W with convolutional function calls that took either x or h and a tensor of unique kernel weights as inputs.

The convolutional LSTM model takes as input the COVID heat map images that have been formatted to be 7-day image sequences. The model structure, similar to the LSTM model, uses 2 ConvLSTM2D layers implemented by Keras that return sequence encoded results to the following layer. The sequence-returning layers are followed by a third ConvLSTM2D layer without sequence outputs. The final hidden state is then flattened and put through a regression dense layer that predicts a vector of length 49 representing the predicted infection rate of each US state. For every ConvLSTM2D layer, we added a max pooling layer that reduced both the height and width of the images in the sequence by half, quartering the overall image size. And similar to the LSTM layer, we decided to add leaky ReLU for the activation function and layer normalization to scale the outputs.

3.3 Late Fusion Model Combining LSTM and Convolutional LSTM

In models designed to solve multimodal problems, there are often different components that process different kinds of inputs. In this late fusion model, for example, a convolutional LSTM model is used to process the image sequences and an LSTM model is used to process the COVID tabular sequences. Then, in order to make predictions based on both kinds of inputs, the encoded results must be combined in some way. We will refer to these methods of combining encoded results from different processings as multimodal fusion techniques. Late fusion is a basic multimodal fusion technique that combines context from different modalities through concatenation.

The late fusion model takes both COVID heat map image sequences and COVID tabular data sequences as inputs. The multimodal model structure includes both unimodal models (convolutional LSTM and LSTM) without their regression dense layer at the end. Therefore, the convolutional LSTM component returns the flattened image hidden state and the LSTM component returns its final hidden state at the end of its third layer. The inputs are passed into their respective unimodal model (image sequence -> convolutional LSTM; tabular data sequence -> LSTM), and then the encoded results from both modalities are joined through a concatenation layer on the last dimension. This concatenated tensor is then passed to a regression dense layer to yield predictions.

3.4 Graph Attention LSTM

Graph attention LSTM cells are a novel combination of the graph multi head attention mechanism proposed in *Graph Attention Networks*[8] and a normal LSTM cell. To construct the cell, we need the proposed attention mechanism and making that requires graph attention heads, which handle the majority of the processing for the mechanism.

In a graph attention head, the node features of the graph are passed in as tabular data and the edge weights as an adjacency matrix. In order to transform the tensors into the desired output size, the node features are first processed through a linear transformation matrix to become node feature tokens. Then, by separately applying 2 linear transformation matrices, transposing 1 of them and summing the results together, we created a simple attention score matrix. To highlight the edges with stronger connections, we added the adjacency and attention score matrices. And after processing the resulting score matrix with the softmax function, we applied the attention scores onto the node feature tokens through matrix multiplication. And finally, we added the attention-applied feature tokens to a separately processed set of node feature tokens in order to achieve a residual effect.

The graph multi head attention mechanism individually processes the input graph with each head and then averages the results. An interesting point to note is that we decided to let each head of the attention mechanism process the entire graph input without splitting the input into smaller query chunks. Though this choice differs from the original multi head attention proposed in *Attention Is All You Need*[2], we decided to stick with the implementation provided in *Graph Attention Networks*[8].

Our graph attention LSTM cell takes heavy inspiration from the Keras implementation of convolutional LSTM cells. All gates in the graph attention LSTM cell are processed outputs from a separate graph multi head attention mechanism, and then using the resulting gates, we applied the original LSTM equations to produce the final outputs of a graph attention LSTM cell. From the equation standpoint, the LSTM equations in Eq. 2 would change by switching all linear transformation matrices W with graph attention mechanism calls that took either x or h and a tensor of unique layer weights as inputs.

The graph attention LSTM model takes in sequential graph nodes formatted as sequential tabular data and sequential graph weights formatted as adjacency matrices. The model structure is the standard 3 layers with LSTMs that are made of 16 graph attention LSTM cells each, and the first 2 layers return outputs in the form of sequences for subsequent LSTM layers to process. However, instead of a dense layer in the end to produce outputs, the final layer of the model acts as the predictor layer by returning its hidden state with the output size of a 49-element vector.

3.5 Image Attention LSTM

Image attention LSTM cells are a new combination of the image attention mechanism proposed in *Image Transformer*[7] and a normal LSTM cell. We decided to implement the 2-dimensional image attention from the paper as we believed that would allow the mechanism to understand regions on the US map better.

In the 2-dimensional image attention mechanism, the input differs from that of usual attention mechanisms that have separate query, key, and value inputs. Instead, this mechanism is only for image self attention and thus all 3 inputs are the same. To mimic having 3 different inputs, we put the image inputs through 3 separate linear transformations to create query, key, and value features. Each of these features are split into separate heads similar to a multi head attention mechanism. Further, the images are split over the height and width dimensions into small kernels, creating a tensor of rank 6. The dimensions represent as follows: batch size, number of heads, number of kernels, query size per head, kernel height, and kernel width. To create our attention tensor, we matrix-multiply the query and key features by their query size per head, kernel height, and kernel width. Then, after scaling the attention tensor with softmax, we finally apply the attention tensor onto our value features through matrix-multiplication again and join the kernels back together.

Our image attention LSTM cell is implemented similarly to our graph attention LSTM cell, again taking heavy inspiration from Keras' convolutional LSTM cell. All gates in the image attention LSTM cell are processed outputs from a separate image attention mechanism, and then using the resulting gates, we applied the original LSTM equations to produce the final outputs of an image attention LSTM cell. From the equation standpoint, the LSTM equations in Eq. 2 would change by switching all linear transformation matrices W with image attention mechanism calls that took either x or h and a tensor of unique layer weights as inputs.

The image attention LSTM model takes in sequential COVID heat map images. The model structure is the standard 3 layers with LSTMs that are made of 16 image attention LSTM cells each, and the first 2 layers return outputs in the form of sequences for subsequent LSTM layers to process. The result from the last image attention LSTM layer is averaged across its height and width dimensions and then passed through a regression dense layer to predict 49-element output vectors.

3.6 Attention Bottleneck Mid Fusion Model Combining Graph and Image Attention LSTMs

Attention bottleneck mid fusion is a multimodal fusion technique first proposed in *Attention Bottlenecks for Multimodal Fusion*[2]. To explain the technique, we will define the 2 key components, mid fusion and attention bottleneck.

Unlike the late fusion technique described in section 3.3, mid fusion joins the encoded results from different modalities at a midpoint of the overall layers of the model. For example, this attention bottleneck mid fusion model processes the image and graph sequences with their respective attention LSTM for 2 layers each, then the encoded results are joined and processed further. In contrast to late fusion, which strictly separates the modalities until prediction time, mid fusion allows the model to understand the different modalities in a mixed context. While this mixed context allows for new levels of understanding that late fusion cannot provide, mid fusion's lack of information on the individual modalities hinders prediction accuracy.

Attention Bottlenecks for Multimodal Fusion[2] addresses the problem with mid fusion's loss of individuality between modalities using attention bottlenecks. Instead of only concatenating the encoded results like in the late fusion model of section 3.3, a kind of padding is added between the encoded results. An important deviation from the originally proposed model is that the format of our 2 modalities are different, which means our concatenation do not evenly distribute the same amount of unimodal information on each end of the resulting tensor. We did not see this change cause any significant issue and thus chose to continue with this difference. After concatenation, we pass the resulting tensor into multi head attention layers for post-fusion processing. Thus, in subsequent attention layers, the padding acts as a bottleneck that limits the exchange of modality-specific attention, allowing the preservation of information for individual modalities on the ends of the concatenated tensor.

The attention bottleneck mid fusion model takes as inputs sequential COVID heat map images, sequential graph nodes formatted as sequential tabular data, and sequential graph weights formatted as adjacency matrices. The multimodal model structure includes 2 of each of the unimodal attention LSTM layers (graph attention LSTM layers and image attention LSTM layers), and only the first of each modality's attention LSTM layer returns the results in sequence form. After the pre-fusion processing for each modality, we concatenated the results with 64 bottleneck tokens in between. Then for post-fusion processing, we passed the tensor through a multi head self attention layer, after which we extracted the ends of the tensor that contained the most modality-specific information. These extracted pieces of information were then passed into separate regression dense layers that each predicted a 49-element vector. And the resulting vectors were averaged to produce the final vector result.

4. Results and Discussion

The metrics presented in the results, mean squared error (MSE) and mean absolute error (MAE), are based on the model's performance on the evaluation dataset, which the model does not see during training. The scale of the errors may seem extremely low, which is because of the nature of the dataset. As seen in Figure 3, the infection rate of any state in any day has rarely exceeded 0.30. Therefore, models' predictions and thus calculated error values are also very small.

Model	Modal	Eval Loss	Eval MSE	Eval MAE	Trainable Weights	Average Time Per Epoch
LSTM	Tabular	7.79E-05	7.51E-05	6.70E-03	21,921	0.5
Convolutional LSTM	Images	1.36E-05	1.12E-05	2.70E-03	95,825	7.8
Late Fusion Model	Tabular, Images	9.23E-06	9.10E-06	2.40E-03	117,697	7.1
Graph Attention LSTM	Graphs	3.69E-05	3.37E-05	4.30E-03	235,392	189.05
Image Attention LSTM	Images	2.02E-06	1.06E-06	8.59E-04	65,745	96.05
Attention Bottleneck LSTM	Graphs, Images	3.93E-08	3.75E-08	1.29E-04	317,475	148.90

Table 1: Evaluation results of different models using COVID heat map images and domestic flight graphs as data.

Additionally, due to the difference in layer structures, the hyperparameters we have selected may have unfairly favored some models over others in performance. Thus, in our results, we also provided the number of trainable model weights and average training time per epoch for analysis. And in order to see if any overfitting has taken place, we provided the performance metrics of the models in their last epoch of training.

Model	Modal	Last Epoch Loss	Last Epoch MSE	Last Epoch MAE	Trainable Weights	Average Time Per Epoch
LSTM	Tabular	8.29E-05	7.99E-05	6.90E-03	21,921	0.5
Convolutional LSTM	Images	6.53E-06	6.27E-06	2.00E-03	95,825	7.8
Late Fusion Model	Tabular, Images	9.08E-06	8.95E-06	2.30E-03	117,697	7.1
Graph Attention LSTM	Graphs	4.84E-05	4.09E-05	4.80E-03	235,392	189.05
Image Attention LSTM	Images	1.97E-06	1.04E-06	8.49E-04	65,745	96.05
Attention Bottleneck LSTM	Graphs, Images	4.96E-08	4.47E-08	1.45E-04	317,475	148.9

Table 2: Results of different models in their last epoch of training using COVID heat map images and domestic flight graphs as data.

4.1 Comparing LSTM to Graph Attention LSTM

We compared the performance between the LSTM and graph attention LSTM models due to the similarity of their inputs since the tabular form of COVID statistics is formatted the same way as the graph nodes are. From the results, we observed significantly lower errors (35% decrease) from the graph attention LSTM model and no significant signs of overfitting from either model. While the error metrics for the graph attention LSTM model are significantly lower than that of the LSTM model, we cannot overlook 3 major differences between the models and their performances.

Firstly, the modalities that the models receive are different. While the graph attention LSTM model received the graph data in the form of both graph nodes and edge weights, the normal LSTM model only received the data of graph nodes in tabular form due to the model's incompatibility with processing edge weight data. Secondly, in terms of computing resources required, the normal LSTM model at this scale is already more than 10 times more resource-efficient than the graph attention LSTM model. If scaled up, this resource difference could increase even more drastically. Thirdly, the training time differs greatly at this scale as well. Out of all models tested, the graph attention LSTM model was the slowest model to train. This difference in training time is largely shared between all LSTM and attention LSTM models, and this could be due to the inefficiencies in customizing layers with Keras[5]. However, such a difference in training efficiency should be noted when selecting a model.

4.2 Comparing Convolutional LSTM to Image Attention LSTM

Due to both models taking image sequences as input, we decided to compare their performances. From the results, we observed significantly lower errors (68% decrease) from the image attention LSTM model and no overfitting from either model. In contrast to the resource difference between the LSTM and graph attention LSTM models, the image attention LSTM model requires less weights than the convolutional LSTM model while achieving better results. However, even so, the training time difference is still drastic between Keras' natively implemented convolutional LSTM and the custom image attention LSTM layers.

Other than training speed, image attention LSTMs outperform convolutional LSTMs in both evaluation metrics and computing resources. And in terms of overall metric rankings, image attention LSTMs ranked second, behind only the multimodal fusion method that utilizes image attention LSTMs in its model structure.

4.3 Comparing Late Fusion to Attention Bottleneck Mid Fusion

Finally, we compared the performances of our multimodal fusion models. We must first mention that the attention bottleneck mid fusion model was the largest model that was tested and it trained the second slowest. However, in terms of results, the attention bottleneck mid fusion model performed far and away the best out of all models in the experiment. Comparatively, the late fusion model's MAE metric was a 1760% increase from that of the mid fusion model, meaning that late fusion model predictions on average deviated 17 times more than the mid fusion model from target values. The late fusion model underperformed to the point that even the convolutional LSTM model achieved better and lower error metrics. This underperformance raises the implication that even if a multimodal model can have all the layers that a unimodal model has, it is not guaranteed that the multimodal model can outperform the unimodal model.

From our results, it seemed that the quality of data in the graph modality was not as high as the quality in the image modality, resulting in better performance from models using images as input. While the attention bottleneck mid fusion model did not have trouble with the difference in information quality, the late fusion model certainly did, which was why the model couldn't even outperform the convolutional LSTM model during training. Therefore, when it comes to multimodal models, the chosen multimodal fusion technique could play a crucial role in determining how well the information between different modalities reinforce each other.

5. Conclusion

In our search for a way to simultaneously predict all state-level COVID infection rates in the United States with COVID heat maps and domestic flight graphs, we propose novel methods of processing graph and image sequences with attention-based LSTM layers as well as evaluate the effectiveness of different multimodal fusion techniques. Across the board, attention-based LSTM models outperformed the LSTM models of similar modality. And between the 2 multimodal fusion techniques examined, late fusion and attention bottleneck mid fusion, the latter performed significantly better than all other models in this experiment. Therefore, among the models tested, we conclude that the attention bottleneck mid fusion model, which uses graph attention LSTM and image attention LSTM, is the model most suited to tackle the task of holistically predicting infection rates across the United States.

Looking towards the future, in terms of possible improvements, some important problems that need to be addressed before serving the attention bottleneck mid fusion model in a production environment are the long training time, lack of a stable data input stream, and large model sizes. We believe that the training time and model size problems could be partially due to the inefficiencies in custom Keras solutions, so exploring other frameworks could be an option to pursue. As for creating a consistent data stream, we think establishing official collaborations with organizations that have reliable domestic flight data and COVID statistics is a method to consider.

We also found an exciting implication worth building upon when considering our attention bottleneck mid fusion model's performance. While designing the model, we decided to deviate from the originally proposed attention bottleneck fusion technique in terms of input format uniformity. Instead of requiring all inputs to be of a certain modality like audio spectrogram images and visual images, we chose to allow different modalities of different input formats, which in our experiment were graphs and images. And due to the excellent performance of the resulting model, we believe it is possible to merge other kinds of modalities with drastically different input formats too.

Therefore, for possible future experiments, we believe that testing the attention bottleneck mid fusion technique on other modalities with different formats is a path worth considering. We also envision the possibility of using the attention bottleneck method to merge more than 2 modalities, which is an area that could warrant more thought. Furthermore, our testing could be more comprehensive. Since our testing of the attention bottleneck model was limited to only the COVID dataset, evaluation with more datasets is required to validate the model's capabilities, and with some hyper parameters as well as loss function tuning, the model could potentially achieve even better results. Finally, in the future, we could test both image and graph attention LSTMs against other sequential methods with similar modalities and on official datasets.

With our attention bottleneck mid fusion model and associated attention-based LSTMs, we presented methods to maximize the use of data formats. In our era of Big Data, complex data formats are often discarded in favor of simpler, more generalized data formats. However, data formats themselves are a kind of data. The difference between presenting US state-level infection rates as a heat map image versus as a table impacts results. Thus, the process of creating a machine learning pipeline is finding the balance between model, data, and data transformations.

Acknowledgments

We would like to thank Eric Bradford for guiding us through the ideology, construction, and experimentation process as a mentor. We are also grateful for Inspirit AI in their offering of the Independent Project Mentorship program, which allowed us to work with Eric Bradford.

References

[1]
A. Hagberg, D. Schult, and P. Swart, “Exploring Network Structure, Dynamics, and Function using NetworkX”, Proceedings of the 7th Python in Science Conference, 2008. 11–15.
[2]
A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, and C. Sun, “Attention Bottlenecks for Multimodal Fusion.” arXiv, Apr. 01, 2022. Accessed: Jun. 14, 2022. [Online]. Available: http://arxiv.org/abs/2107.00135
[3]
A. Vaswani et al., “Attention Is All You Need.” arXiv, Dec. 05, 2017. Accessed: Jun. 14, 2022. [Online]. Available: http://arxiv.org/abs/1706.03762
[4]
C. B. Vennerød, A. Kjærran, and E. S. Bugge, “Long Short-term Memory RNN.” arXiv, May 14, 2021. Accessed: Jul. 12, 2022. [Online]. Available: http://arxiv.org/abs/2105.06756
[5]
F. Chollet et al., "Keras", 2015.
[6]
H. Wickham, "ggplot2: Elegant Graphics for Data Analysis," R Package Version 3.3.6, 2016. [Online]. Available: http://CRAN.R-project.org/package=ggplot2.
[7]
N. Parmar et al., “Image Transformer.” arXiv, Jun. 15, 2018. Accessed: Jul. 06, 2022. [Online]. Available: http://arxiv.org/abs/1802.05751
[8]

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks.” arXiv, Feb. 04, 2018. Accessed: Jun. 14, 2022. [Online]. Available: http://arxiv.org/abs/1710.10903

[9]

R. Becker, T. Minka, A. Deckmyn, "maps: Draw Geographical Maps," R Package Version 3.4.0, 2021. [Online]. Available: http://CRAN.R-project.org/package=maps.

[10]

X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.” arXiv, Sep. 19, 2015. Accessed: Jul. 12, 2022. [Online]. Available: http://arxiv.org/abs/1506.04214

Notes

All experiments, TensorFlow model implementations, and statistical results can be found here: https://github.com/NuoWenLei/Attention-LSTMs-in-Multimodal-Models