Using Distillation to Learn from an LSTM Model for COVID Hospitalization Prediction

Chad Weatherly

Advanced Machine Learning – COSC 7362

May 9th, 2022

Abstract

As deep learning continues to grow in its use and application, there is a need for less complex models. While large organizations with large amounts of resources (large university groups, national laboratories, Google, Amazon, Microsoft, etc.) may have the computational capability needed for training and deploying complex models, it is often infeasible for many applications. Distillation is a form of transfer learning that aims to teach a less complex student model the same decision boundary as a larger complex model. It has been shown that distillation can reduce model size while maintaining accuracy. In this project, we have a large swath of COVID-19 data from Google at different days. The goal is to train a larger deep learning model using a Long Short-Term Memory (LSTM) layer that learns the data in a time series format, but then transfers this knowledge to a less-complex student model without an LSTM layer. If the student model can learn successfully without an LSTM layer, then it would show that the time series information can be learned without a specific time-specific component implemented in the architecture itself.

Introduction

While deep learning models have been shown to be extremely powerful, they also have become increasingly computationally complex. This means that large amounts of computing power are needed to train large models. For large corporations and groups with large amounts of funding, this is doable. However, for smaller groups and most applications, these computational resources are not available to them. This presents a challenge. How can we reduce the model size while also maintaining accuracy in our models? Ba et al. proposed the question, do deep neural networks really need to be that deep?[1] Often, computational complexity is not given as much thought in research. Accuracy is the metric given the most value. However, this comes at an extreme cost, not only financially but also in terms of energy. Kamperis et al. found that training the BERT language model on a GPU had the same carbon footprint as a transatlantic flight. Training a large transformer neural network has almost five times the carbon footprint as the average car creates over the course of its lifespan, including gas consumption.[2] If this is the case, there is certainly a necessity for learning networks to be incredibly smaller moving towards the future.

Many applications also need smaller models. There are many scenarios where having a smaller model is a must. Many embedded systems, such as cell phones or hearing aids, can benefit from the aid of neural networks, but these models must meet size constraints. Many models may need to be trained or do inference across a network or remotely, meaning they must be able to work under bandwidth constrictions. Other times, a quick inference or training is necessary, such as in medical emergency situations where a quick diagnosis is needed.

A little over two years ago, COVID-19 completely changed the way the world worked, throwing much of our ordinary lives into disarray, and causing millions of deaths worldwide, up until this writing. Google compiled a complete data set consisting of information regarding the pandemic, in an effort to have a comprehensive repository of information for different regions each day since the beginning of 2020.[3] Each observation consists of information referencing demographics, hospitalizations, deaths, vaccinations, emergency mandates, epidemiology, mobility trends, search trends, weather, and more for each region on each day. The aim of this project is to separate the data into all 24 consecutive day time periods. Using all the information mentioned above for the first 10 days, we want to predict hospitalizations for the next 14 days, or two weeks.

Previous Work

Caruana et al. were the first to propose a method for neural network model reduction, as best we can tell. In their paper, they advocate for model compression, a method of reducing model size by training a student model by matching the Softmax outputs of a trained teacher model. They introduced the concept of distillation.[4] In their research, they were able to use distillation to reduce image classification model parameters by 88% or more and still maintain accuracy.

Hinton et al. introduced the Learning Using Privileged Information (LUPI) paradigm, where information is available at training that is not available at testing. This is called Privileged Information (PI). Information that is available at testing and training is Observable Information (OI). In the case of the paper, privileged information might be extra annotation of images to aid in future image segmentation and classification. Or it might be information about patient recovery after a patient has had a disease. This information can be used to train a teacher's classifying boundary and then taught to a student learner who only has access to OI. The paper introduces the algorithm SVM+, which works like a regular Support Vector Machine (SVM), except privileged information is used to introduce slack variables that give a bias to the classifying boundary.[5]

Vapnik et al. proposed a general distillation framework, based on Hinton et al.'s work, that is model-agnostic and works under the LUPI paradigm. It was a joining of the two concepts, LUPI and Distillation, into one. It also expanded the domain that distillation could be used in, not just as a method to compress models but also as a type of transfer learning.[6] [7]

Xu et al.'s research is a great example of how to use distillation effectively. In it, they used the BERT language model, mentioned above. Using Pruning of layers and distillation at every layer, a BERT student model was able to reduce its parameter size by more than 75% while only losing ~3% accuracy, quite a remarkable feat. So, it has been shown that distillation is an effective method to reduce model size and even teach a student model with less information as well as a highly complex and accurate model.[8]

Methodology

The data was pulled directly from Google, using their own API. For pre-processing, the data set was read in and features selected. The entire data set contains more than 1 million rows, so to reduce data set size, only data for the 'Texas' region was used. All columns that had only one unique value were removed (like region, country code, etc.), and all NA values were replaced with zero. All features were scaled using a MinMax scaler, standardizing all features to be in the interval [0, 1]. Finally, the data set was ordered by date, so we could capture the time series sequential data. The data set was re-ordered, so that each consecutive 24-day period is one time sequence. This sequence was split into the first 10 days and the final 14 days. The first 10 days contained all the features (472 in total) for each day, giving us 472 * 10 = 4720 input values for each time period. The target values were the hospitalizations for the following 14 days, giving us 14 target values, in sequential order. In total, there are 837 observation periods, so N = 837.

To measure success of the models, simple loss is used. For training the teacher model and student models, Mean-Squared Error (MSE) was used to match the outputs to the target values. Accuracy was not considered, since it would be nearly impossible to guess the exact hospitalizations for each day. Also, we want models that predict trends well. If our models perfectly matched the 14 days, then it would be more likely that the LSTM was simply learning the pattern of a specific signal, rather than the general direction that hospitalizations are happening.

For training and testing, the data was split using K-Fold Cross Validation, with five folds being chosen. Therefore, each fold is 20% of the dataset. Five teacher models were produced, and each model trained using four of the folds while testing on the fifth, so that each model tests on a separate testing set. The teacher model consists of an initial LSTM layer. LSTM models have proven to be an improvement upon simple RNN layers, due to the fact that LSTM contains a short-term hidden memory state as well as a long-term memory state, which has the ability to forget and remember information over a sequence.[9] The LSTM layer is able to learn the sequential time series information as part of our data set. The LSTM layer is then followed by 3 linear, fully-connected layer, each with 25 neurons, and a ReLU activation function in between each layer. The final output is 14 neurons, which are each sent through a Sigmoid activation function, since all the target values have been standardized. The teacher model that has the best average of training and testing loss is used to train the student model.

There are two student models that are created. Let's call them S1 and S2. S1 is trained solely on matching the outputs of the best teacher model. S2 is trained to match the correct target values. Both consist of a simple feed-forward neural network, with 4720 inputs, and 3 fully-connected layers, each with 50 neurons. In between the layers are ReLU activation functions, with a Sigmoid Activation function after the final layer. The student models are both trained using the same optimizer as the teacher model: the Adam optimizer. The teacher model has a learning rate of 0.0001, while the student models have a learning rate of 0.00001.

Experiments

For each model, we ran 5 different models, one for each fold of the K-Fold splits. Each of those models were trained for 100 epochs, as it was found that 100 epochs were enough for the loss to converge to a reasonable valley. Below are the results for the teacher and student models. The “Student” model references S1. The “FNN” model references S2.

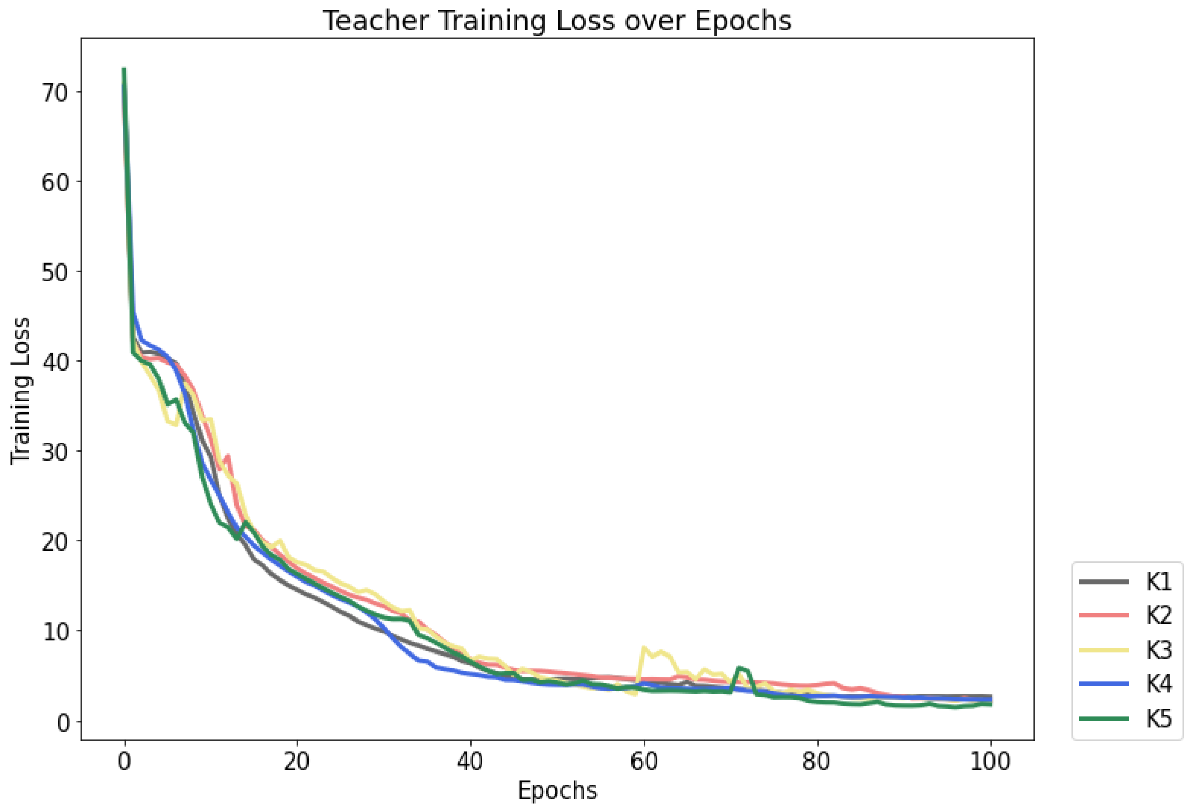

Figure 1: The teacher training loss across epochs

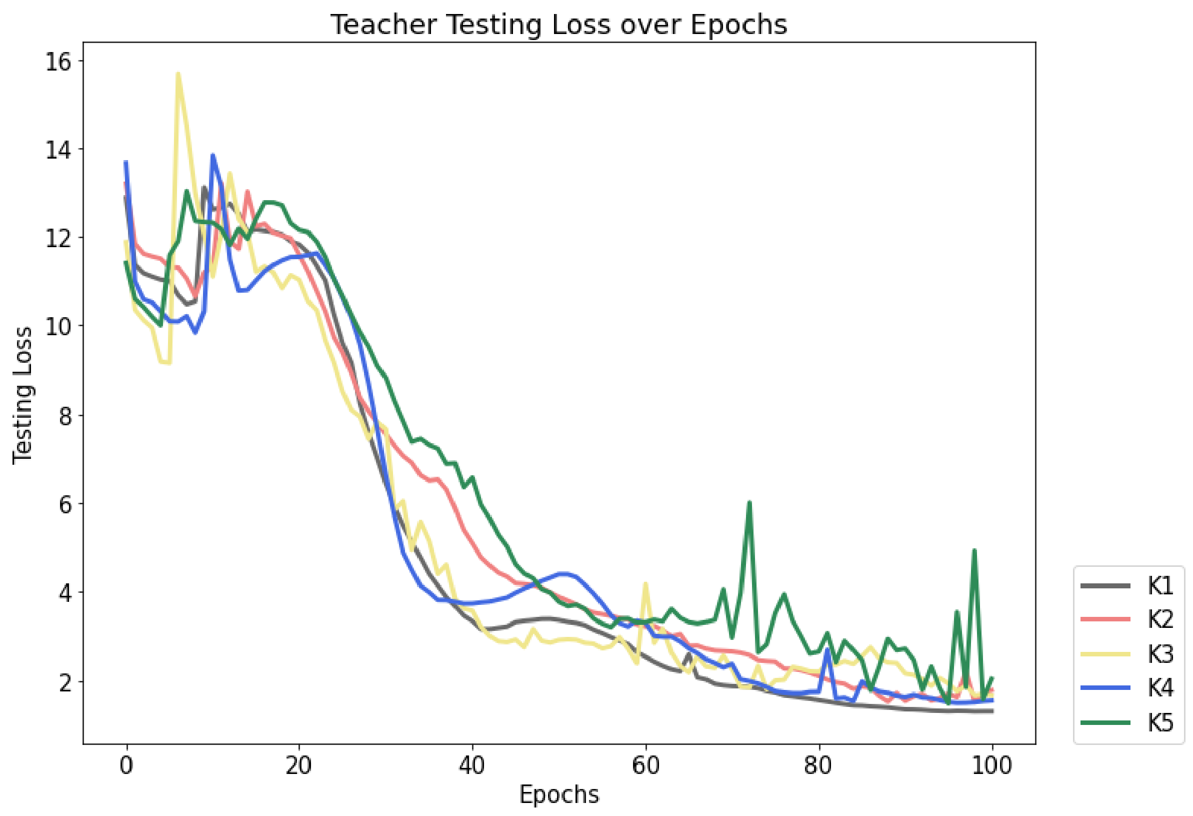

Figure 2: The teacher testing loss across epochs

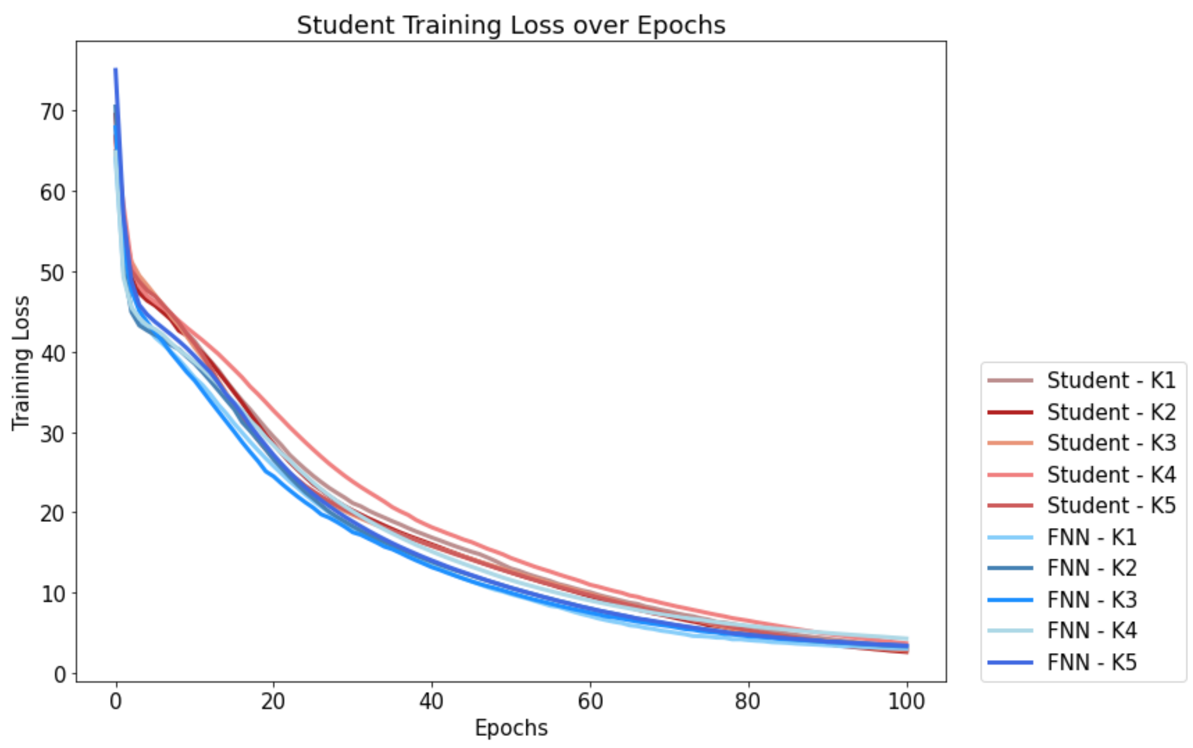

Figure 3: The student training loss across epochs

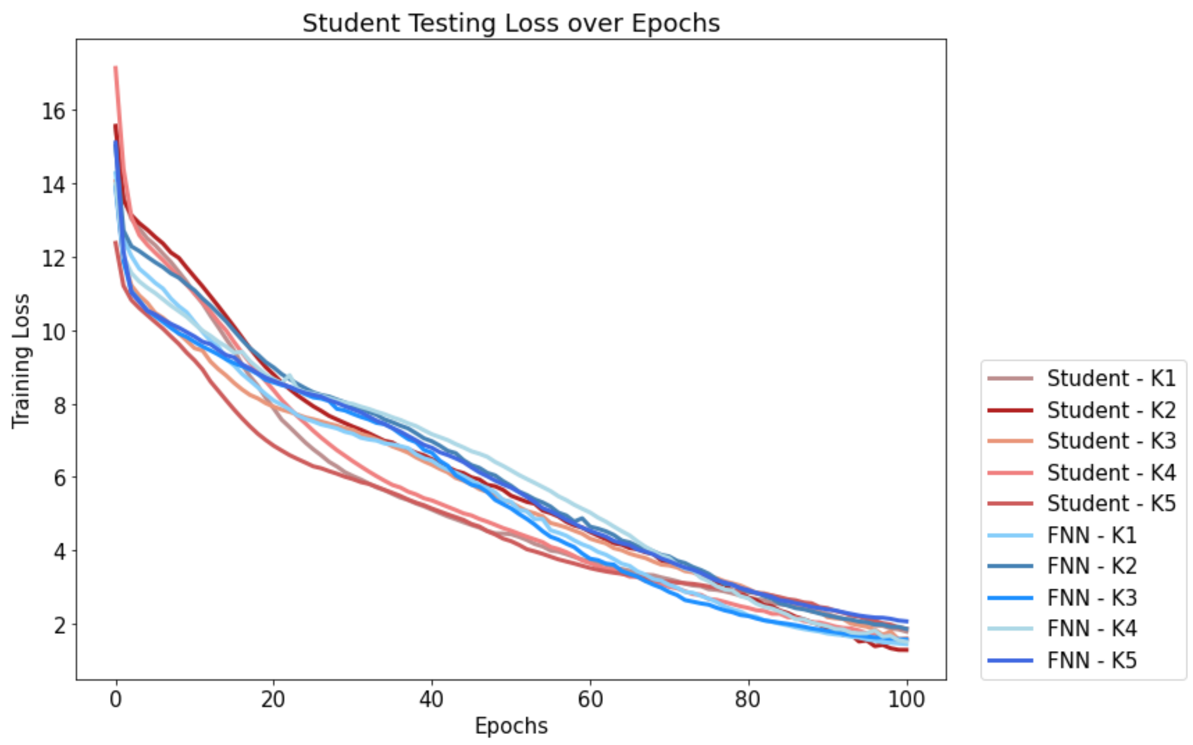

Figure 4: The student testing loss across epochs

Average Loss	*Teacher Model (LSTM)*	*Student Model 1 (Learn from Teacher)*	*Student Model 2 (Learn from real targets)*
Average Training Loss	2.22	2.85 (+0.63)	3.58 (+1.36)
Standard Error of Mean	0.09	0.14	0.22
Average Testing Loss	1.64	1.91 (+0.27)	1.66 (+0.02)
Standard Error of Mean	0.26	0.09	0.14

Figure 5: Comparing mean loss values (across 5 models) for different architectures

Conclusions

Looking at the above figures, it is apparent that all models were able to learn fairly well and converged to small loss points. From figure 5, we see that the LSTM teacher model was able to learn the best, which is to be expected, because it captures the most information from the data set. The student model that learns from the teacher learns the second best, while the student model that learns only from target values learns the worst out of the three architectures. This is to be expected, because the student model is trying to learn the boundary of the teacher, giving it access to more information about the data set. The second student model does not receive any of this information from the data set. Each observation is treated as independent, so naturally it loses the time component in its training. It’s also interesting that the student models have higher standard errors, which is to be expected. The student models have access to less information, so their variation will be more spread out in their ability to predict correctly. The LSTM is able to have more bias because it has a better grasp of the data. Based on the average testing values and their standard errors, it is shown that training a student model using distillation from an LSTM model has better results than training a student model from target values alone. The student model trained using distillation was able to capture some sort of sequential information from the LSTM model, even though not as efficiently as the LSTM model itself. This is a promising result and confirms the hypothesis positively that a simple, fully-connected neural network can capture time series or sequential data.

However, how well do any of the model architectures actually predict hospitalizations for the final 14 days of each time period? While we cannot look at each observation, we can see a small sample of how some of the models predicted different time periods. Figures 6-10 are below in the **Appendix** section. In general, the teacher model is plotted in shades of green, while the student model trained using distillation is plotted in reds, and the student model trained on target values is in blues. What can be seen is that the teacher model obviously performs the best, but in general, the predictions are often quite off if there is not a good pattern to follow. For the earlier days of the pandemic, there were no new hospitalizations, and the models performed poorly (Figure 6). What’s seen is that the models tend to follow the trend very well, but the bias can be inaccurate, guessing hospitalizations to be way above or below what they actually ended up being. If there was not a clear trend in hospitalizations, the models tended to vary more in their predictions as well (Figure 7). If the trend is very clear, then the predictions are extremely clustered (Figure 10). Overall, the findings from these plots show similar results to what we see in Figure 5, which tells us that distillation, even for different types of models, can be an effective way to teach new, simple architectures for future applications.

Appendix

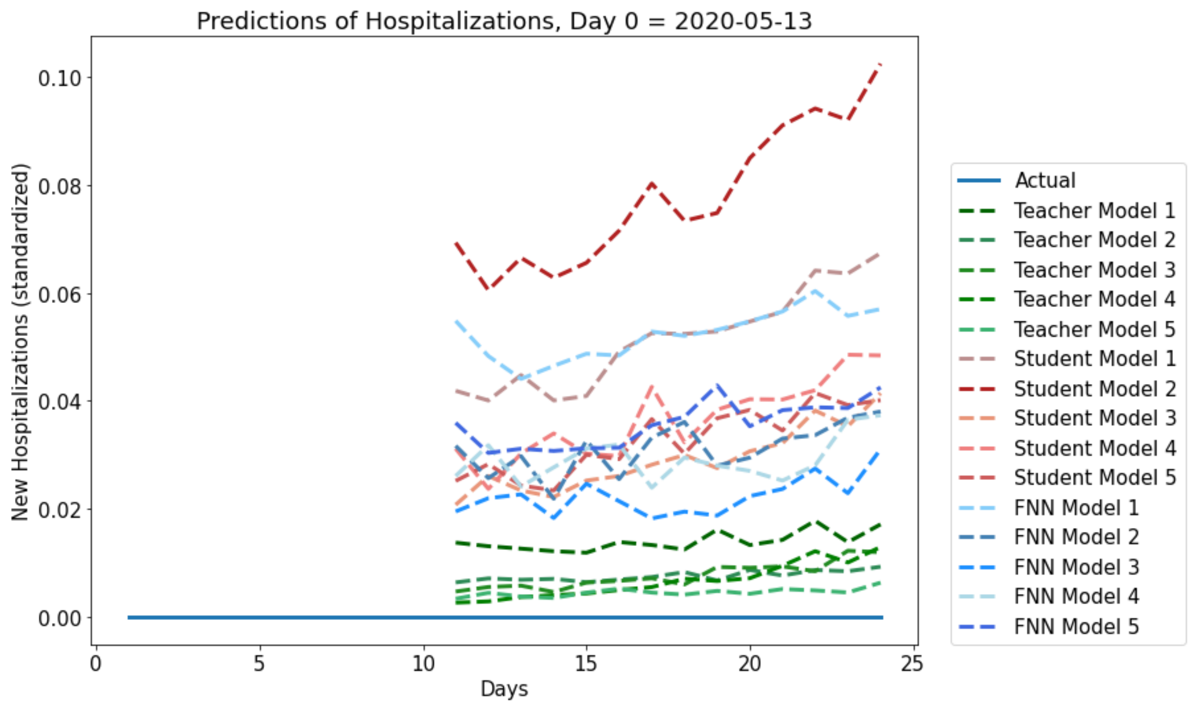

Figure 6: Hospitalization Predictions for all 3 architectures across all 5 folds/models

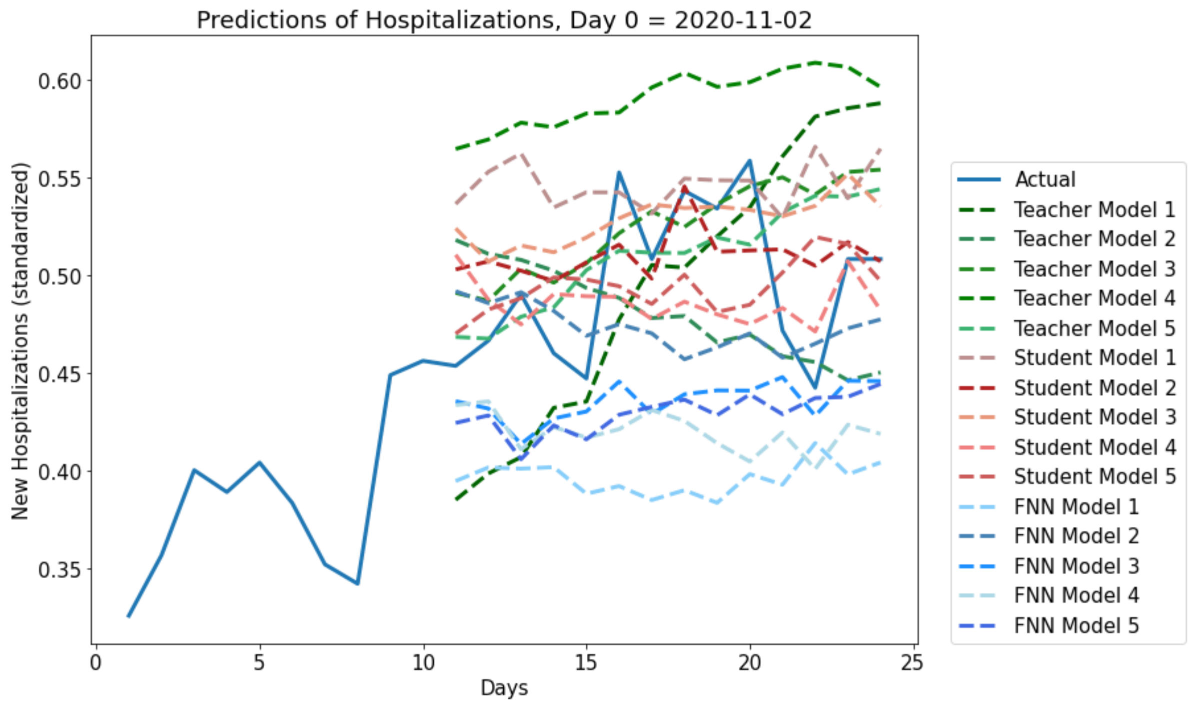

Figure 7: Hospitalization Predictions for all 3 architectures across all 5 folds/models

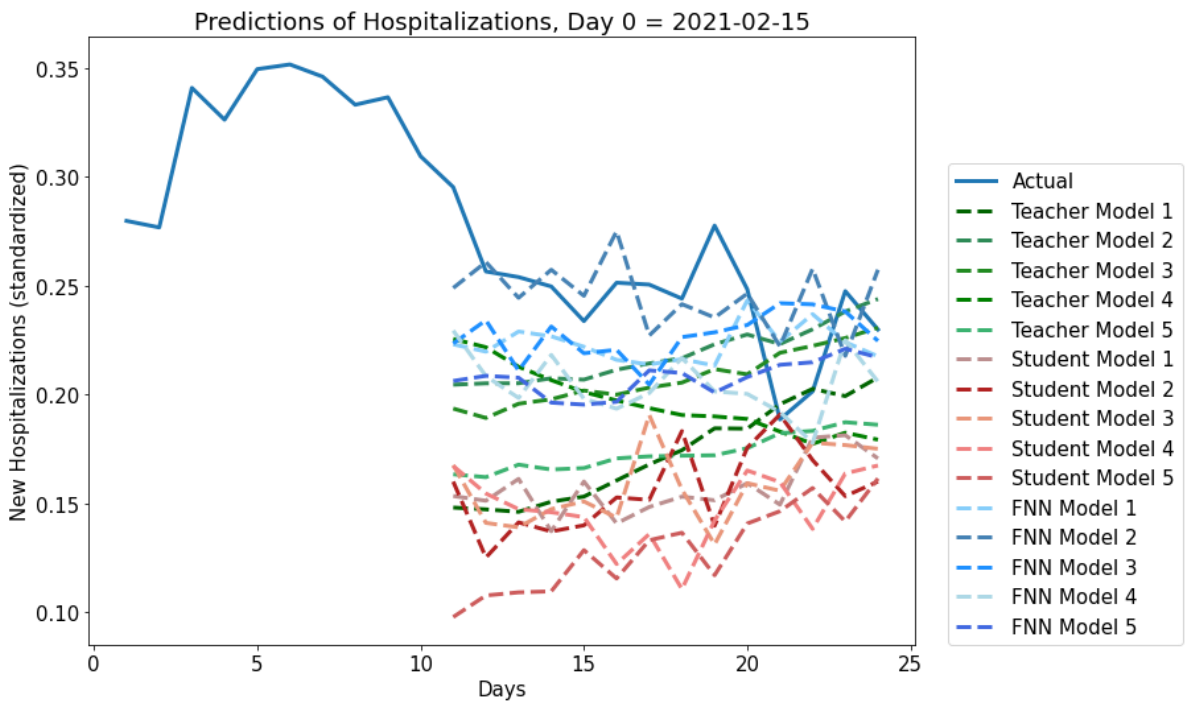

Figure 8: Hospitalization Predictions for all 3 architectures across all 5 folds/models

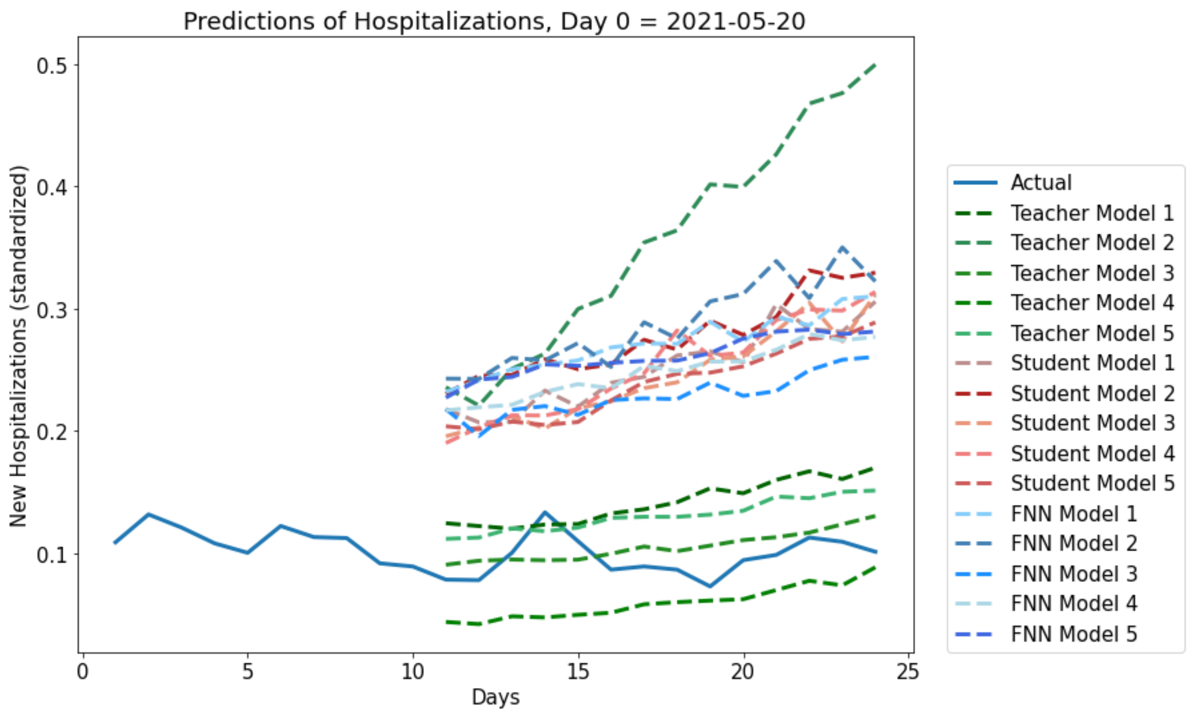

Figure 9: Hospitalization Predictions for all 3 architectures across all 5 folds/models

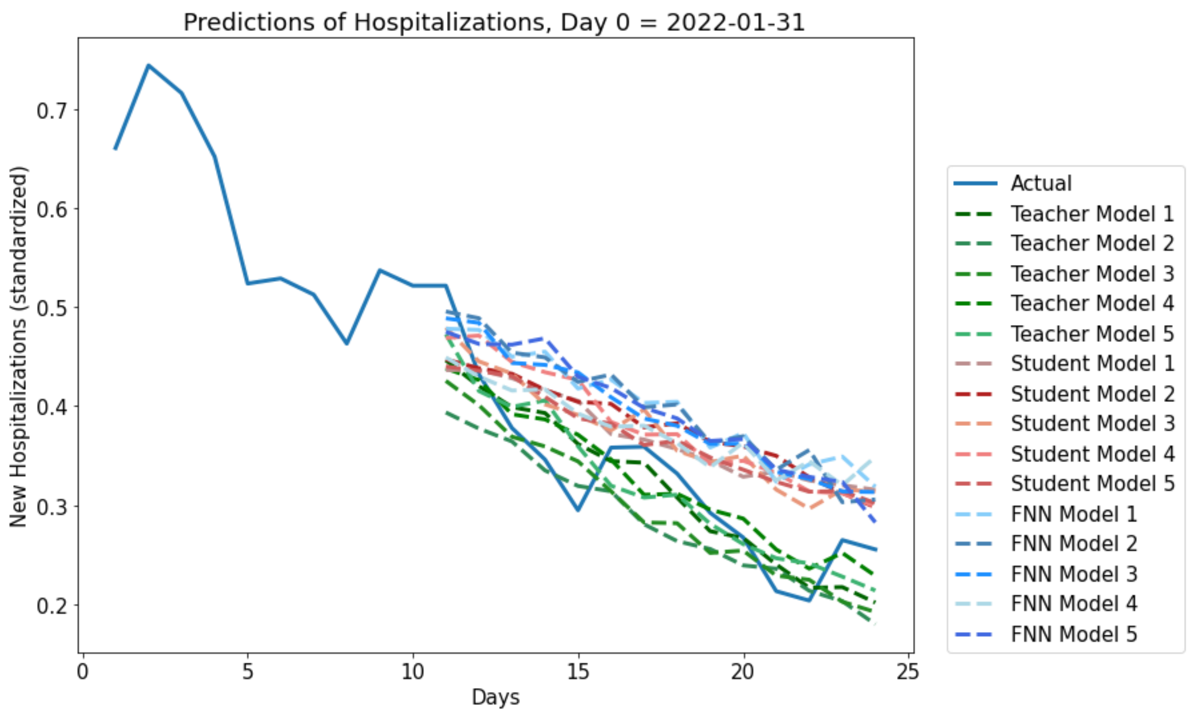

Figure 10: Hospitalization Predictions for all 3 architectures across all 5 folds/models

References

[1] L. J. Ba and R. Caruana, “Do Deep Nets Really Need to be Deep?,” *ArXiv13126184 Cs*, Oct. 2014, Accessed: Apr. 19, 2022. [Online]. Available: http://arxiv.org/abs/1312.6184

[2] S. Kamperis, “Energy considerations for training deep neural networks,” *A blog on science*, Aug. 14, 2019. https://ekamperi.github.io/machine%20learning/2019/08/14/energy-considerations-dnn.html (accessed May 08, 2022).

[3] https://health.google.com/covid-19/open-data/raw-data?loc=US_TX

[4] “compression.kdd06.pdf.” Accessed: Apr. 19, 2022. [Online]. Available: https://www.cs.cornell.edu/~caruana/compression.kdd06.pdf

[5] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network”

[6] V. Vapnik and R. Izmailov, “Learning Using Privileged Information: Similarity Control and Knowledge Transfer”

[7] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, “Unifying distillation and privileged information”

[8] D. Xu *et al.*, “AutoDistil: Few-shot Task-agnostic Neural Architecture Search for Distilling Large Language Models”

[9] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” *Neural Comput.*, vol. 9, pp. 1735–80, Dec. 1997, doi: 10.1162/neco.1997.9.8.1735.