PHP2650 Project4 Clara Hu

Introduction

The data that I obtained record 4.2 million traffic accidents took place in the US from Feb 2016 to Dec 2020. The original traffic event data was extracted from two API sources ('MapQuest Traffic' and 'Microsoft Bing Map Traffic') including the time, location, and distance of the accidents. There are some real-time weather information such as temperature, humidity, wind speed and precipitation combined with the traffic event that are extracted based on the GPS coordinates and timestamps. Moreover, this data contains some infrastructure information corresponding to the location, such as train station, stop signs, junctions and so on. All the infrastructure information are binary entries meaning they are either true or false. The traffic events are categorized into 4 levels based on their severity which is categorized on the duration of traffic event. The original data has 4.2 million rows and 39 columns. As some columns describes similar information, such as address, or only one level in the categorical variables, I had removed those columns as well as the ones with more than half missing values. Also, some outliers are removed based on investigation of the weather information. Therefore, the final dataset includes 3.7 million records with 21 variables.

I used tree modeled for the prediction before and due to the imbalance of severity level, the prediction model did not perform well. Therefore, I decided to predict on the continuous variable, car accident duration. Since the severity level was classified based on the accident duration, predicting the duration variable would provides similar aspects. Also, the duration can provide information other than accident severity, so build a model on the car accident duration would be a better approach for this dataset. The tree models provide me some idea on the importance of different features, for example year, distance and month are important factors to predict car accidents durations. Since year is one of the leading factor of the data variance, I found that if I split my data to two subset, car accidents in 2020 and car accidents in 2016-2019. The random forest model performed differently. The random forest model performs better on the older data where the lowest test RMSE is **35.1** and the 2020 data has the lowest RMSE around **51.6**. Moreover, predictors such as distance, month, hours are more important in

2020 data, while distance, timezone, hour, pressure, windspeed and temperature are important in data before. Therefore, I decided to have two subsets separate for building future prediction model.

Modeling

For the deep learning model, I decided to fit the 2020 data and older data separately. I would use MSE as my metrics so that I can convert it to RMSE in order to compare the model performance between deep learning model and my tree models. The deep learning model, unlike tree models, do not allow missing values, and need the predictors to be in matrix. I used the same dataset that I used to train my final random forest models but removed all missing values (<1% of the data), and convert X into matrix using model.matrix() function. Independent variable, duration, was select as y in a 1D array. Both X and y are split into training and test set (80/20). Since my independent variable, y, is continuous, both X matrix and y array are scaled by minus the mean and divided by standard deviation.

To set up the initial network architecture, I started with a simple 3 layer model, one input, one hidden and one output layer (Table 1). The input layer has 26 units which is the same as the column numbers of X variables. The hidden layer inherited the same number of units. Both layers will be activate by RELU function. The output layer has only one unit and is activated by linear function because I am predicting continuous outcome. The model is compiled using 'Adam' as the optimizer, MSE as both the loss function and metrics. The first model is fit using 100 epochs without early stopping and using default batch size, 32. I use 20% of random shuffled training samples as validation set to evaluate the validation loss and MSE changes.

Model: "Initial model"

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 26)	702
dense_4 (Dense)	(None, 26)	702
dense_3 (Dense)	(None, 1)	27

```
=================================================================
Total params: 1,431
Trainable params: 1,431
Non-trainable params: 0
_________________________________________________________________
```

Table 1: parameters of the initial model

I chose a simple 3 layer model as the initial one because I would like to see how the model is learning from the training set without overfitting on training. The epoch was set to 100 for me to inspect any overfitting trend. For the initial model, I would expect to see some overfitting trend in later epochs so that I can control the learning by adding more capacities and use regularizer to prevent overfitting. However, I expect the optimization of deep learning model to be difficult because it is complex and has a lot of hyper-parameters that we can adjust, which requires thorough understanding of the data and each parameter.

I started from fitting the model on the 2020 subset. Both validation and training loss decreases as the number of epochs increases, and after 20-30 epochs the loss stabilizes (Figure 1). I transformed the scaled predicted y back to original format and calculated the RMSE which is 51.45, while the RMSE I had from random forest is 51.65, indicating similar performance. However, I think the initial model is learning too slow, and the model may still underfitting, so I decide to add more capacity to the training. First of all, I increases the batch size to allow the gradient to be updated on more samples. The training loss does not seem to change too much (Table 2) when the batch size increases, but higher batch size allows the model to fit faster and smoother training loss (Figure 2, top left). Therefore, I use batch size of 200 for the followed model optimization.

Layers (total)	Batch size	Epochs	Final Training loss	Final val loss
3	32	100	0.669	0.675
3	100	100	0.672	0.679
3	200	100	0.671	0.674
4	200	100	0.661	0.672
5	200	100	0.659	0.668
6	200	100	0.659	0.671

Table2: change of training loss at different capacity

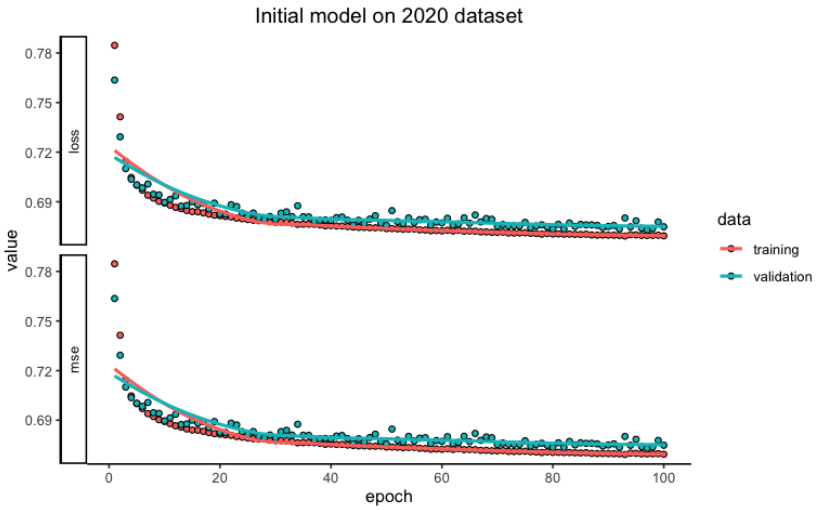

Figure 1 Loss and MSE from fitting 2020 data with initial model

Then I started to add more hidden layers to the model architecture to increase flexibility of the model which may help the model to learn better, in other words improve loss. With the same number of epochs and batch size, adding layers from 3 to 4 decreases more on the training loss indicating better learning. 5 layer model has the lowest training loss as well as the validation loss, while 6 layer did not improve more. All the layer added are hidden layer which has the same number of units as the input layer, 26, and was activated by RELU. According to the training and validation loss plots (Figure 2), no obvious overfitting trend is identified from the initial model as both validation and training loss drops in similar rate. In 4 layer model the validation loss seems deviate around 30 epochs and the decrease rate becomes slower. Similar trend is observed in the 5 layer and 6 layer models but in a greater extent. The overfitting starts to be an issue beyond 5 layers, but may be regulated when the model is below 5 layers. According to the above observations, 4 to 5 layers may be better model architecture for the 2020 subset.

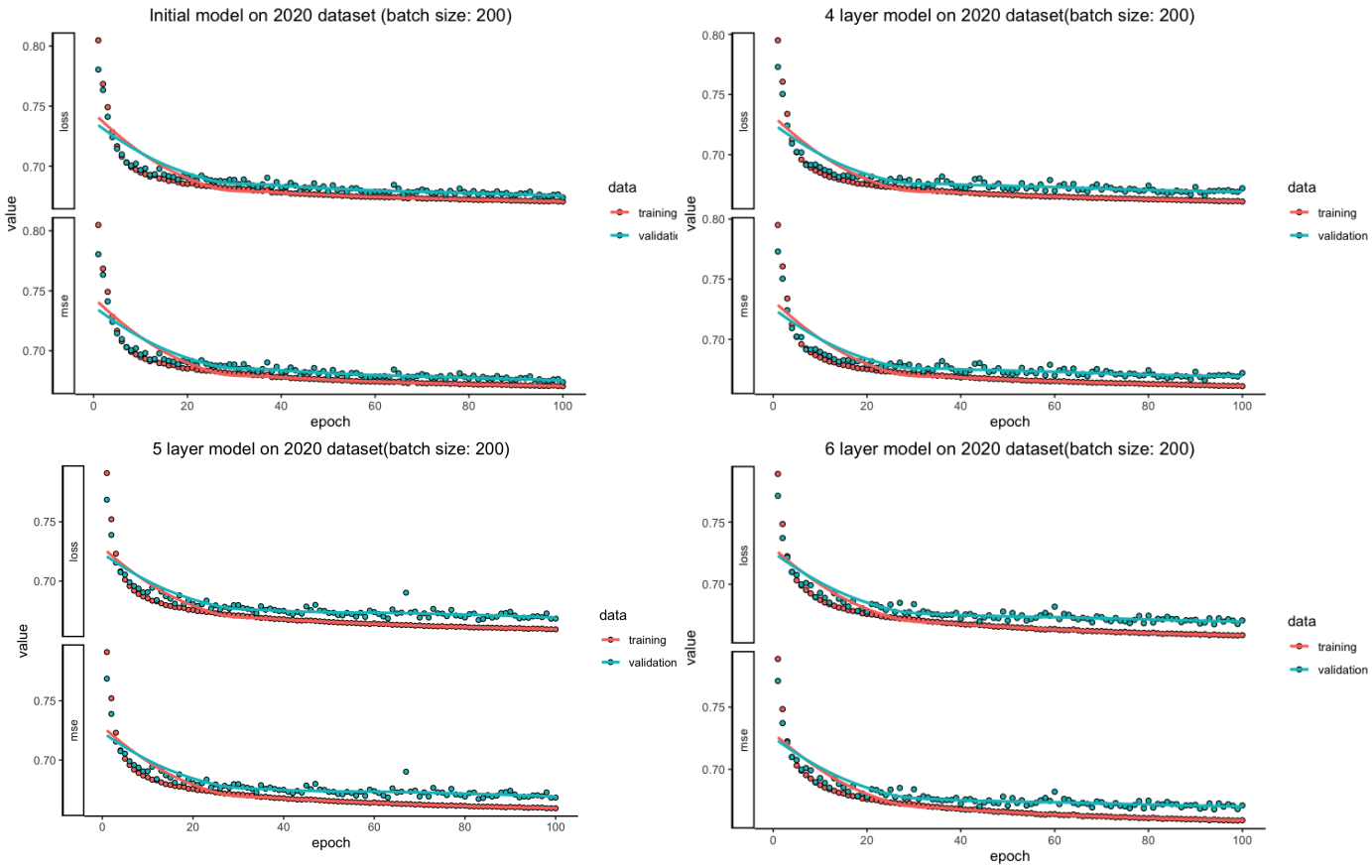

Figure 2 Loss and MSE from fitting 2020 data with 4 model architectures

As I selected two better performed deep learning architectures, the regularization need to be applied to alleviate the overfitting in the model. The regularization I tried are early stop, dropout layer and L1 regularizer. There are a lot more regularizations in the deep learning model, but it is difficult to try multiple combinations, so I only used three in optimizing my model. For the 5 layer model, because I saw divergence at around 30 epochs (Figure 2), I applied early stop with patience of 4 to evaluate how the validation loss differ from the training loss. As I expected, the epoch stops around 32 epochs. However, the difference between validation loss and training loss is similar to the difference before the application of early stop, and the validation loss is not as low as the unregularized model. Then I tried dropout layers at rate of 0.5. Either 2 dropouts or 1 dropouts perform better, and increases validation loss. The validation loss fluctuate more dramatically than the model in Figure 2. Therefore, those punishment may be too much on my 5 layer model. Then I tried one dropout layer after input layer with a rate of 0.2 and

Layers (total)	Regularization	Epochs	Final Training loss	Final val loss	Test MSE
5	Early stop with patience of 4	100 (stop at 32)	0.671	0.679	0.679
5 + 2 dropout	Dropout rate: 0.5	100	0.749	0.741	0.742
5 + 1 dropout	Dropout rate: 0.5 Early stop with patience of 4	100 (stop at 11)	0.743	0.751	0.752
5 + 1 dropout	Dropout rate: 0.2	100	0.687	0.688	0.688
4	Early stop with patience of 4	100 (stop at 40)	0.669	0.677	0.677
4 + 1 dropout	Dropout rate: 0.5 Early stop with patience of 4	100 (stop at 16)	0.733	0.744	0.745
4	L1 regularizer: 0.01 (Input layer)	100	0.663	0.673	0.674
4	L1 regularizer: 0.01 (All layers)	100	0.661	0.670	0.670
4	L1 regularizer: 0.01 (All layers) Early stop with patience of 4	100 (stop at 37)	0.672	0.679	0.680

Table3: changes in validation loss and training loss with different regularizations

without early stopping, which lead to relatively low validation loss. Additionally, there is a convergence between the training loss and validation loss in this model (Figure 3). At around 50 epochs, the validation loss is synchronized with the training loss and according to the values, they difference between training loss and validation loss is minimal. The early stop function does not do well with this dropout as it stops too early and lead to higher validation loss. Therefore, a 0.2 dropout layer after input layer may be the best regularizer on the 5 layer network from all the tries. It may be helpful to add some small additional regulation on this model as the last couple of epochs may overfit the model. However, I have not find the perfect setting to prevent that yet.

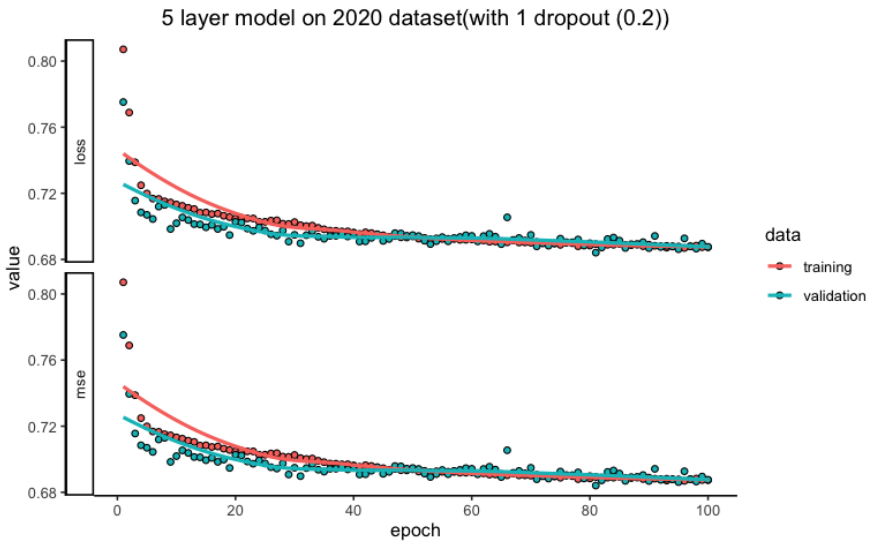

Figure 3 Loss and MSE in 5 layer model with one 0.2 dropout layer fitting on 2020 subset

For the 4 layer model, although the training loss is not as low as the 5 layer model before regularization, it is worth to try different regularizations on that architecture as it is better to have a simpler model than a model with a lot of layers and parameters. The early stop works pretty good on this model as it retains most learning on the training set and the validation loss is relatively low. The drop out layer of 0.5 provides too much punishment on the model. Then I tried different settings of the L1 regularizer on 4 layer model. I used 0.01 as the parameter for L1. I first applied it on the input layer which leads few changes in the loss and the trend comparing with the one without. Therefore, I decide to enhance the regularization on the model and applied the same L1 to all 4 layers, which leads to low validation loss (Table 3). However, I still inspect some overfitting in the model (Figure 4). Therefore, I applied early stop (stopped around 37 epochs) on the same model which increases both training and validation loss but the distance between the loss are smaller. Among all I would pick the 4 layer model with L1 = 0.01 on all layers with early stop as the best 4 layer model fitting the 2020 subset because it has relatively small validation loss and stoped early to prevent the overfitting. The difference between training and validation loss is relatively small, so the model stops to fit before the loss diverge too much (Table 3 and Figure 5). I believe that there may be a better parameter for this 4 layer model, but it may take a long time to try out different possibilities.

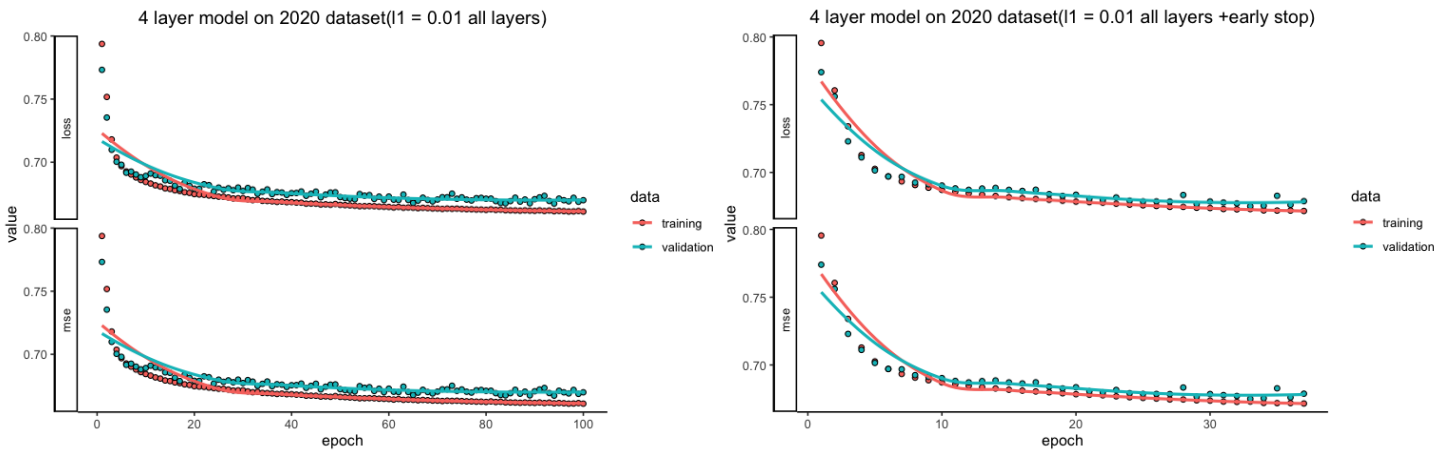

Figure 4 Loss and MSE on 4 layer model with L1 fitting on 2020 subset

Since the test MSE shown in Table 3 are scaled MSE, I converted the predicted y values back to their original format and calculated the RMSE. The RMSE from the 4-layer with L1 and early stop is 51.62, and the RMSE from the 5-layer with 0.2 dropout layer is 51.88 where the test RMSE from random forest is 51.65. The 5-layer without any regularization model has the lowest test RMSE at 51.19 but still within the difference of 0.5 and may potentially overfit the model. All those model performs similarly and the deep learning model does not seem to perform better than the random forest models, and I am not able to see which feature weight more in the deep learning model.

I fit the initial 3 layer model on the 2016-2019 dataset with 100 epochs as well and with a batch size of 200 to speed up the fitting. As expected, the fitting on the old subset looks different from the 2020 subset. The validation set performs better than the training set in terms of MSE (Figure 5), and I did not see the validation loss increase above training loss at any point within 100 epochs. The final training loss is 0.682 and the validation loss is 0.673 in this model. Similar to what I did to the 2020 subset models, I add another hidden layer with the same number of units to the model (4 layer). The 4 layer model seems to fit better and the two loss function starts to converge at around 25 epochs (Figure 6). The final training loss is 0.667 and the validation loss is 0.664, both lower than the 3 layer model indicating better performance. As I am worried about some overfitting in this model, I applied an early stop with patience of 4 to the 4 layer model which stops the model at 29

epochs, where train loss is 0.680 and the validation loss is 0.673. Then I added another hidden layer to have the 5 layer model and fit with 100 epochs, the train loss is 0.663 and validation loss is 0.661, which is not much improvement on the 4 layers and the early stop performs similar as the early stop on 4 layer model. Therefore, the 4 layer model may be a better fit for the old subset. Without early stop, the original format RMSE is 34.47 and is 34.68 with the early stop. The RMSE from random forest on old subset is 35.1. Both 4 layer deep learning model performs slightly better than random forest. If I would be conserve on the overfitting, I will choose the one with early stop (right Figure 6).

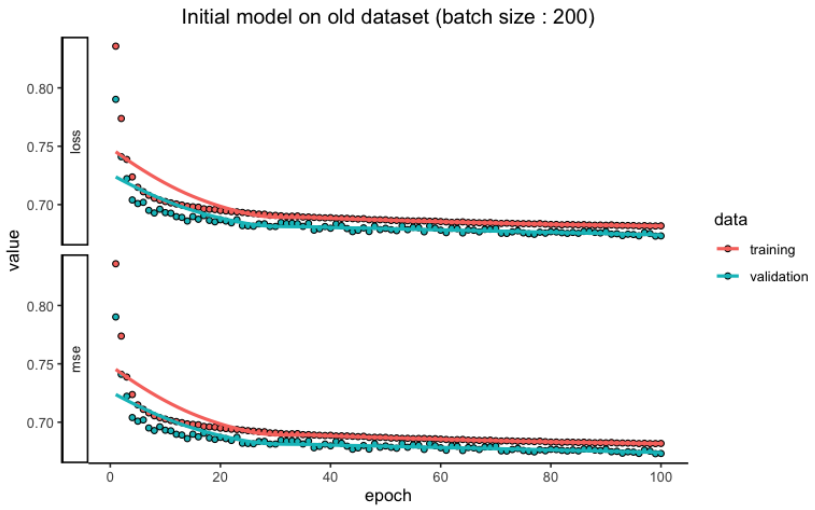

Figure 5 Loss and MSE in the initial model on the old subset

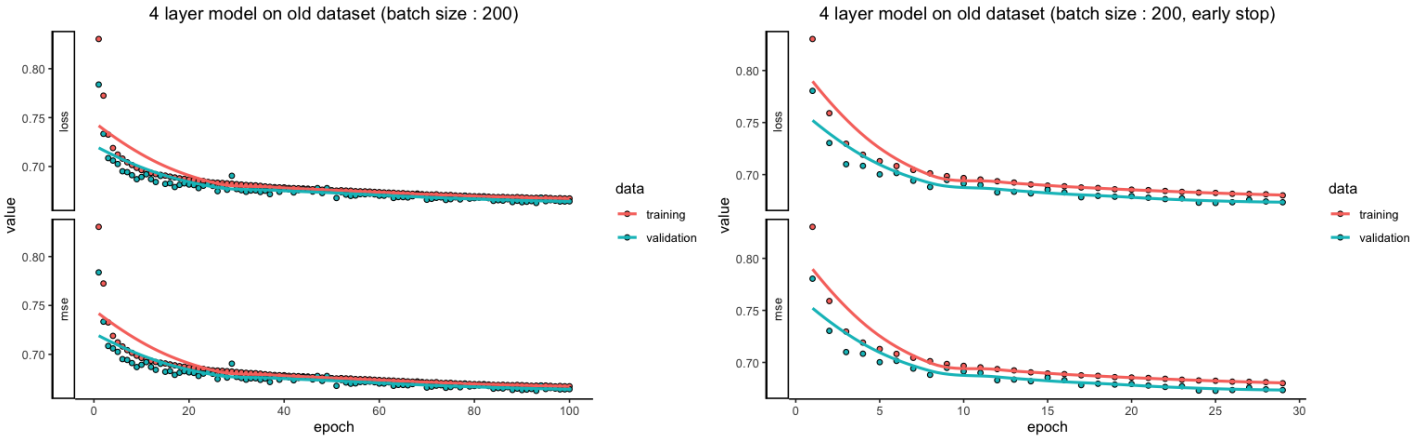

Figure 6 Loss and MSE in the 4 layer model on the old subset

Evaluation and Conclusion

For the best models in each subset, the lowest RMSE I am able to generate is 51.19 for 2020 subset and 34.47 for old subset. However, I am worry about overfitting trend on the training set, the final model I picked for 2020 subset is the 4 layer with L1 on all layers plus early stop which gives test RMSE of 51.62. The random forest model on that subset has the best RMSE at 51.56, so the deep learning model performs similar to random forest on this subset. For the older subset, the 4 layer with early stop gives RMSE at 34.68 and the best from random forest is 35.1. According to the loss function plot, I find the older subset may fit better in the neural network model, since the validation set performs better than training and the overfitting is minimal. However, in the random forest model, the older data set had a stronger overfitting tendency as the tree number increases at certain point comparing to the 2020 data.

It would be better to compare the residuals in deep learning model and in the random forest model, but both models require long time to run and I did not have saved residuals fore tree models, so I am not able to include that in this report.

My assumption on those two subset is that the old subset may have fewer variance than the new dataset, so that both deep learning model and tree model can easily learn from the training set. In the 2020 subset, even though it is hard to learn, increasing model capacity did not improve the training too much and the tuning the regularizers did not help with lowering the test MSE. The 2020 data may have some variances that are due to random noise as that is the period when the pandemic started. The noises may interfere the training of the model. However, it is also possible that I picked wrong parameters for the modify the model, so that I saw minimal improvements.

The issue with deep learning model is that it has too many hyper parameters and I am confused when to add flexibility and when to increase the punishment. Also, for most of the time, when I fine tuning the parameters, it did barely nothing on the model. Even with the results showing in this report, the changes in MSE and loss are small and can be random chances. Moreover, it is hard to interpret. Therefore, I may not choose to use deep learning model for the prediction of this data.