Assignment 2: Neural Networks & Keras Intro

BJ Bae

March 8, 2023

Contents

1 Introduction

The purpose of this assignment was to replicate the textbook's examples on 3 different types of neural networks (binary classification, multiclass classification, and regression) and run further research on each of the models. The binary classification example took the IMDB Movies dataset in Keras to determine whether or not a movie would have positive or negative reviews based on its characteristics. The multiclass classification example took the Reuters dataset in Keras to classify a topic based on a short bit of text. The regression example used the Boston Housing dataset in Keras to predict median home prices in a Boston suburb in the 1970s given the neighborhood characteristics. None of these problems are particularly new or difficult but they provided good practice to develop a strategy to handle different types of problems.

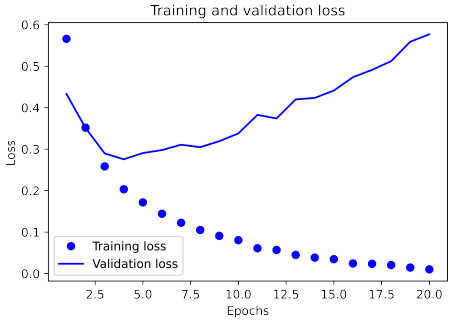
Another part of the assignment was to also pick out a dataset to analyze and create a neural network through the Keras library. The dataset that was used for this part of the assignment was Seoul's bicycle sharing dataset from UCI. The goal for this part of the assignment was to predict how many bicycles would be

used in bicycle sharing system at a certain time given particular characteristics about a day like temperature, humidity, holiday, etc.

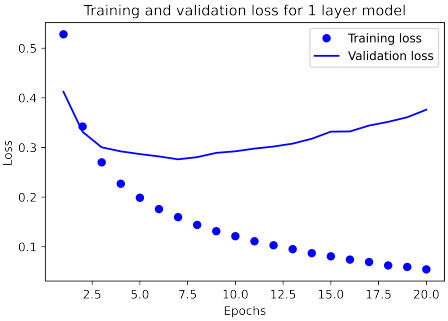
2 Discussion

2.1 Binary Classification

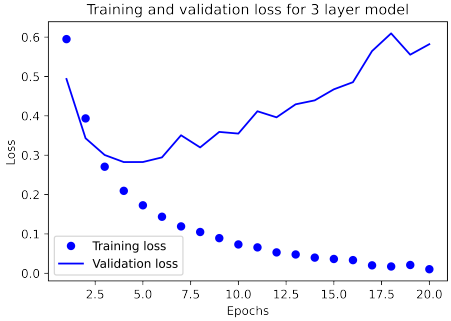
The IMDB movies dataset's further research, shows that removing hidden layers makes the overfitting process slower, while adding layers to the original model, speeds up the overfitting process in less epochs.



(a) original model



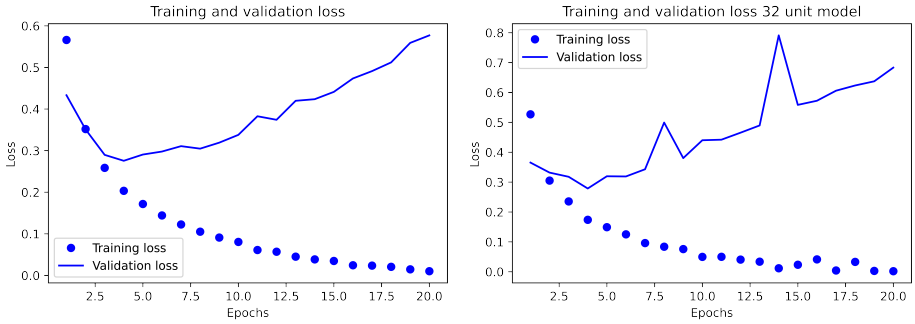
(b) 1 layer model



(c) 3 layer model

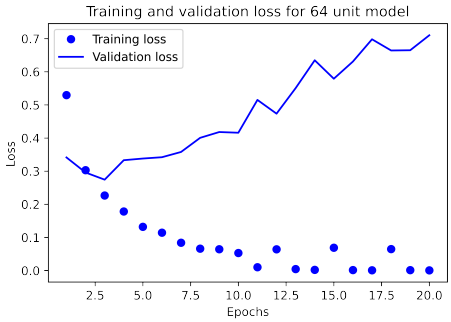
Figure 1: training and validation loss for models with changing depth

Adding more than units per layer did not have any significant difference model improvement. The higher unit per layer seemed to create more spikes in validation data and overfit the model much more in the same amount of epochs than the original model.

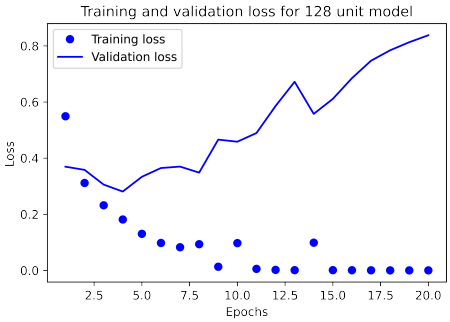


(a) original model

(b) 32 unit model



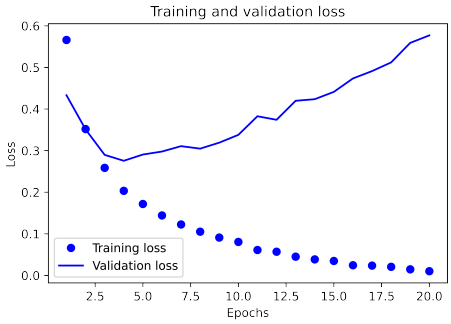
(c) 64 unit model



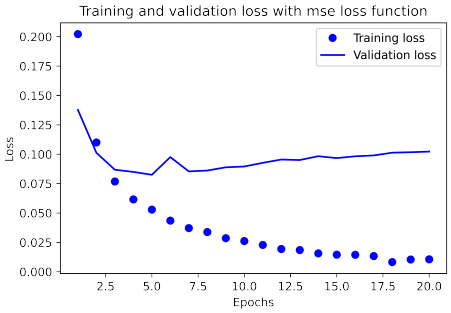
(d) 128 unit model

Figure 2: training and validation loss for models with changing depth

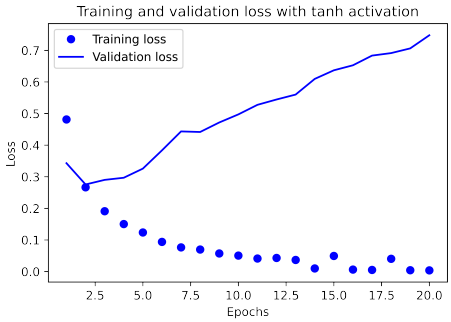
Changing the loss function to mse resulted in the validation loss being relatively constant between epochs. The tanh activation function created a faster overfitting model and created a worse fitting model than ReLU.



(a) original model



(b) model using mae loss function

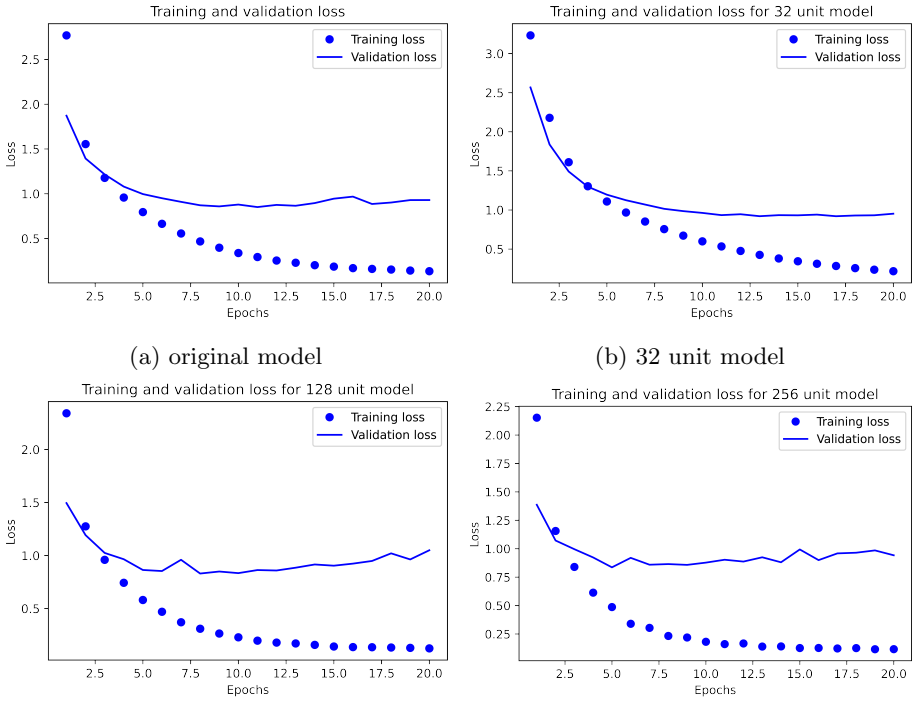


(c) model using tanh activation

Figure 3: training and validation loss for models with changing depth

2.2 Multiclass Classification

The model with 128 and 256 unit per layer performed the best with the lowest validation loss. The 32 unit model did worse than the original model which was using 64 units.



(a) original model

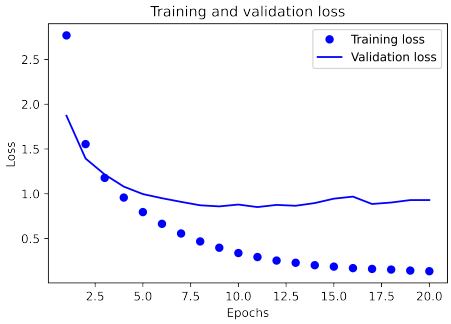
(b) 32 unit model

(c) 128 unit model

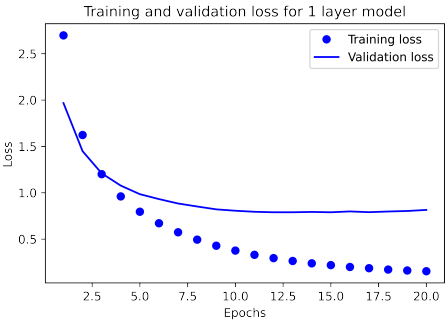
(d) 256 unit model

Figure 4: training and validation loss for models with changing width

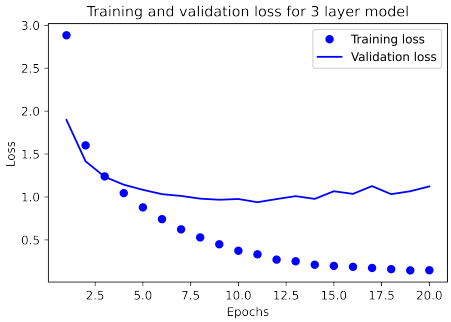
The 1 layer model seemed to do slightly better than both the original and 3 layer model. It was also more stable with a smooth curve compared to the rugged curves of the validation loss of the original and 3 layer model.



(a) original model



(b) 1 layer model



(c) 3 layer model

Figure 5: training and validation loss for models with changing depth

2.3 Regression

There were no futher experiments with this regression example.

2.4 Bicycle Sharing Regression Model

The bicycle sharing dataset has information about date, temperature (°C), humidity (%), wind speed (m/s), visibility (10m), dew point temperature (°C), solar radiation (MJ/m^2), rainfall (mm), snowfall (cm), season, hour of day, functioning day , holiday, and rented bike count. The functioning day field is a boolean value that represented if the bike sharing system was functioning or under repair. The holiday field is also a boolean value that represented if that day was a holiday or not. For data cleaning before feeding the data into the models, the date field was removed to not have any identifiers on specific dates that the model could have access to for training, although seasonality and time of day were both important fields that the model could train on. The data points where the non-functioning day field was true and the entire field of functioning day were also removed. The model could use this field to predict a drop in bicycles rented with 100% correlation with this variable, which was not the

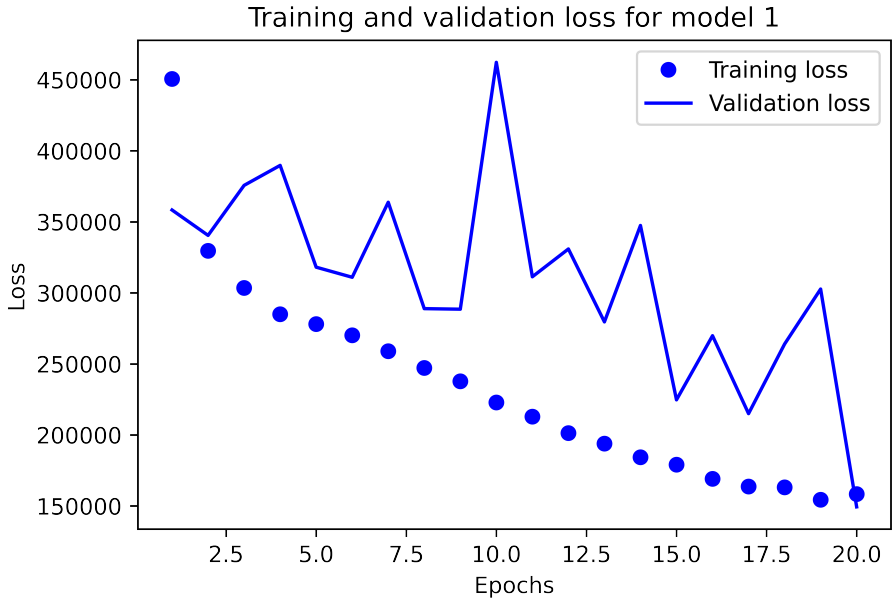
intended purpose of this exercise. With all the non-functioning days removed, the column would only contain functioning days which would be useless for the model so the column was also dropped. However, the data points on holidays were not dropped.While a drop in bicycle rentals might be inferred, it is a more natural occurence and it is not immediately clear what type of effect hoidays might have on bike rentals. This variable might also aid the model in describing the anomalies that happen during holidays.

The season, hour of day, and holiday fields were all one hot encoded, while the rest of the continous predictors were standardized.

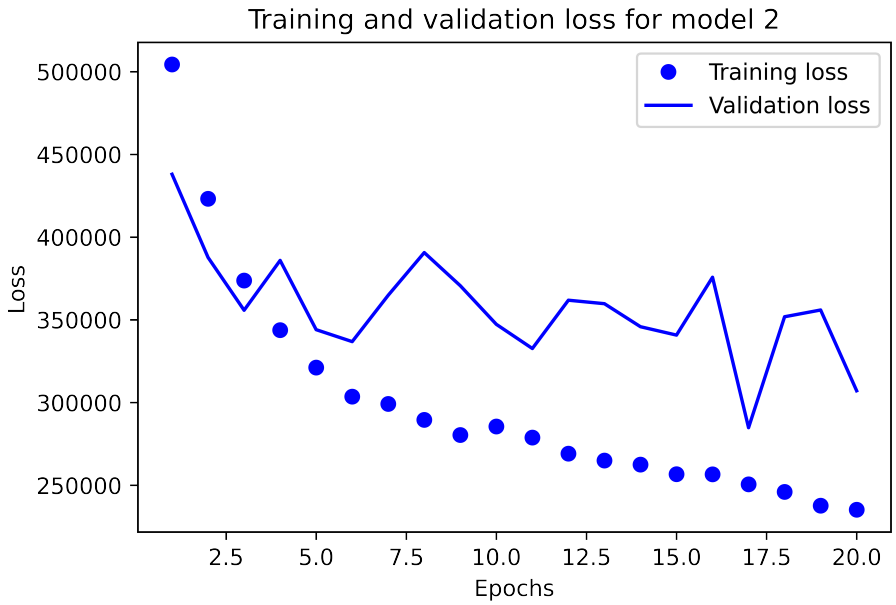
3 Methods

There were a couple different models that were created in the process of finding the best model in predicting the most accurate bicycle sharing rental rate estimate.

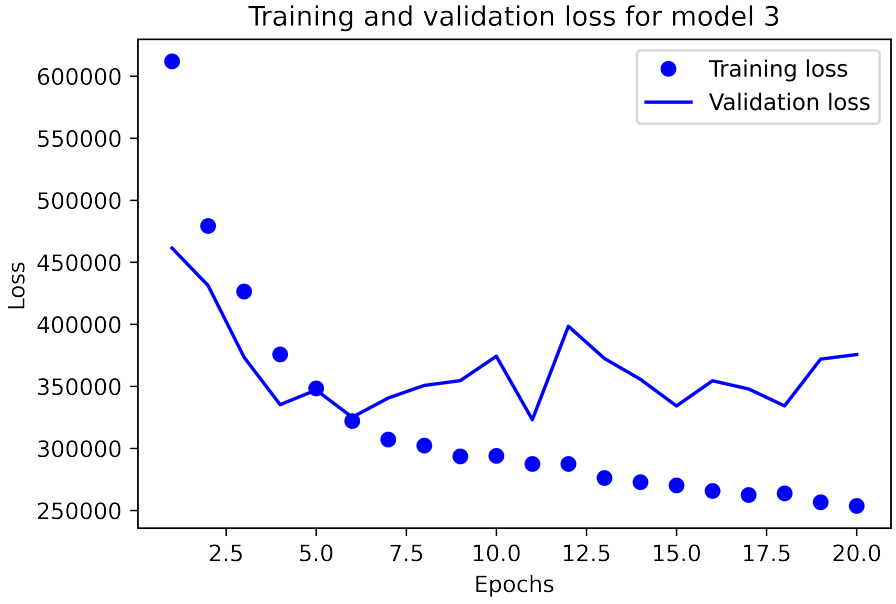
All the models used a 20% dropout layer between each hidden layer. All the loss functions were using mean squared error with the adam optimizer and also displaying the mean absolute error metric.



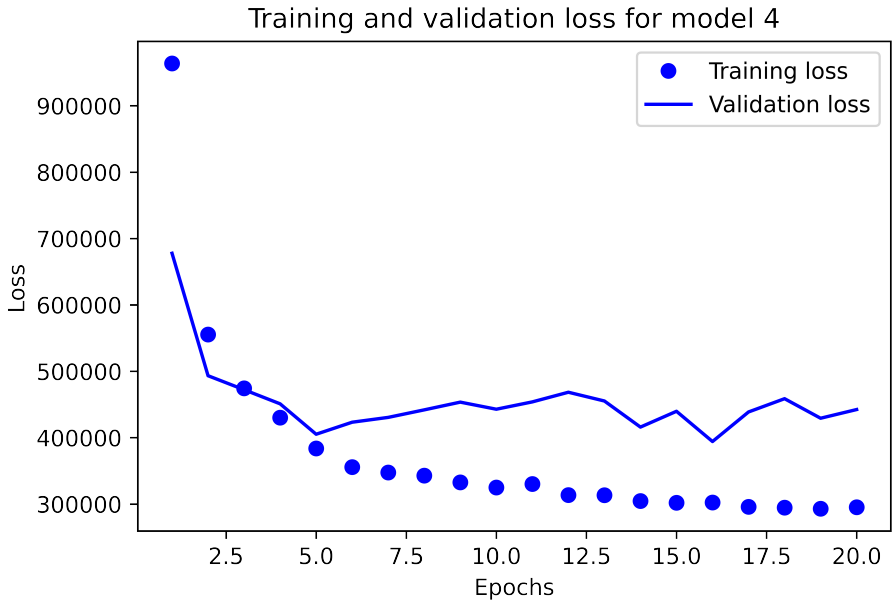
The first model used a 8-8 hidden layer structure with a batch size of 1. It resulted in very high loss and big spikes in the validation loss, but there was a noticeable negative slope in both training and validation losses, meaning that the model was underfitting.



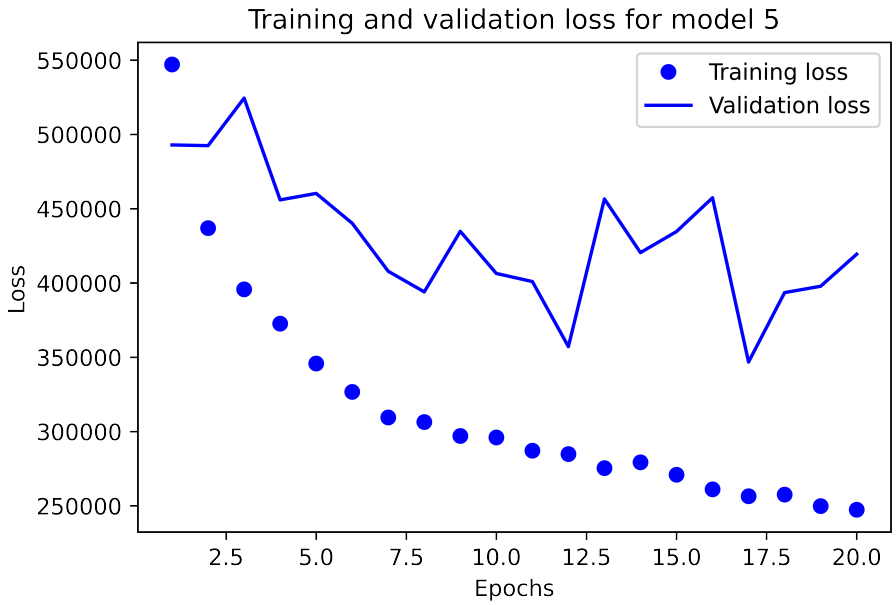
Model 2 used the same structure as model 1, but with a higher batch size of 10 to mitigate the large spikes in the validation loss. While losses went up it solved the instability problem of the validation loss.



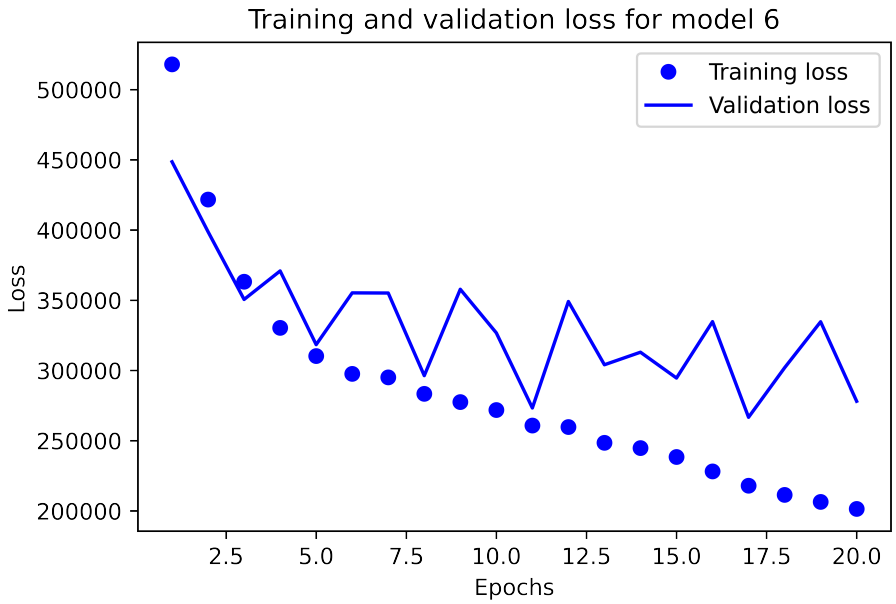
Model 3 modified model 2 by increasing the batch size to 20, but the change made the stable validation loss worse and the relative training loss the same.



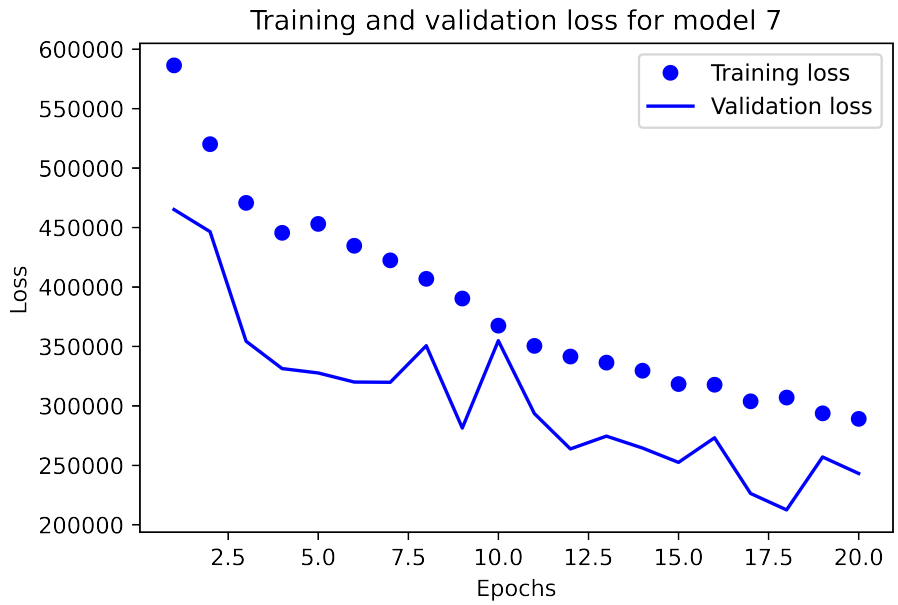
Model 4 kept the batch size of 20 but added one more layer making it an 8-8-8 structure to increase model complexity. While this looked like the validation loss was overfitting, it seemed that the loss was unchanging through the epochs. This was confusing, so instead of increasing the depth of the neural network, the next iteration of the model added more width instead.



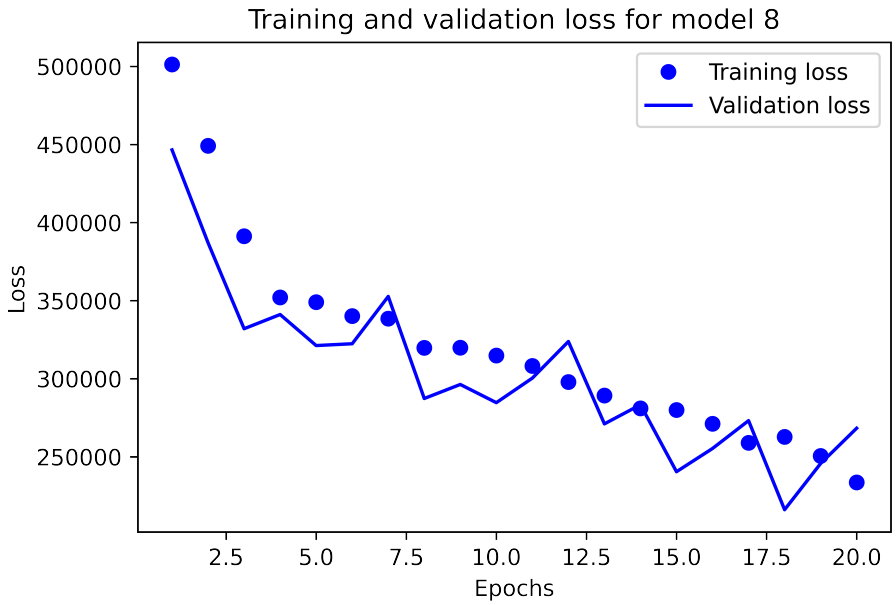
Model 5 built off of the 8-8-8-1 structure to a 16-16-16. This model completely overfit and was not the right direction. One reason that was hypothesized was because the input layer of 35 features were being bottlenecked by the first 16 unit hidden layer.



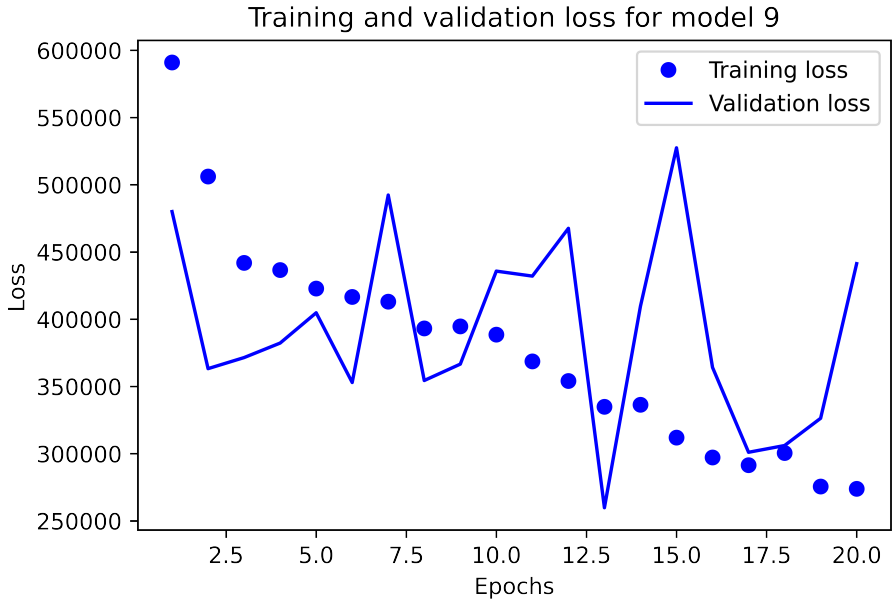
To fix model 5's overfitting, model 6 made the first hidden layer 64 units (64-16-16). This helped stop the overfitting and decreased overall loss to the lowest levels.



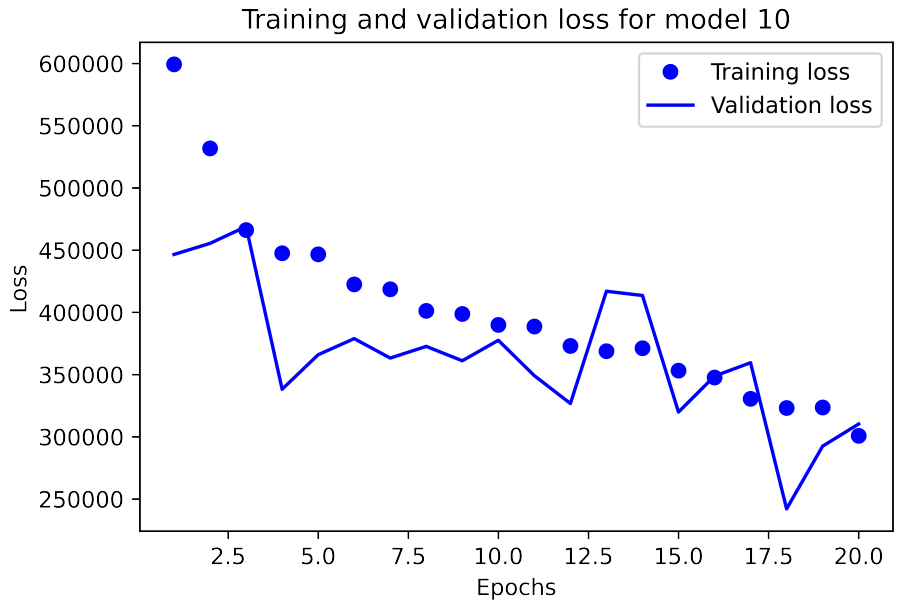
Following up on model 6's more asymetrical width structure, model 7 made the second to last hidden layer more narrow, down from 16 to 8 units (64-16-8). This significantly helped bring the validation loss. The validation loss was below the training loss. One of the explainations for this was that the training data goes through the dropout layers which might contribute to higher loss that the validation data doesn't go through. Since the loss was so low, the next model adds complexity to gain lower loss before it overfits.



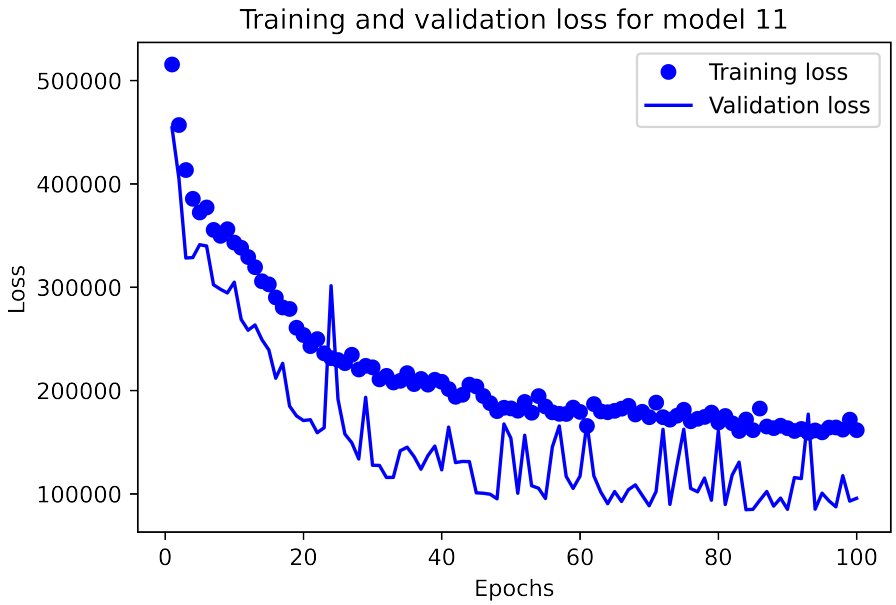
Model 8 adds another hidden layer making the structure 64-32-16-8. For all these past models, unless stated otherwise, has no change to the batch sizes or any other parameter. This model performed even better and continued to not overfit.



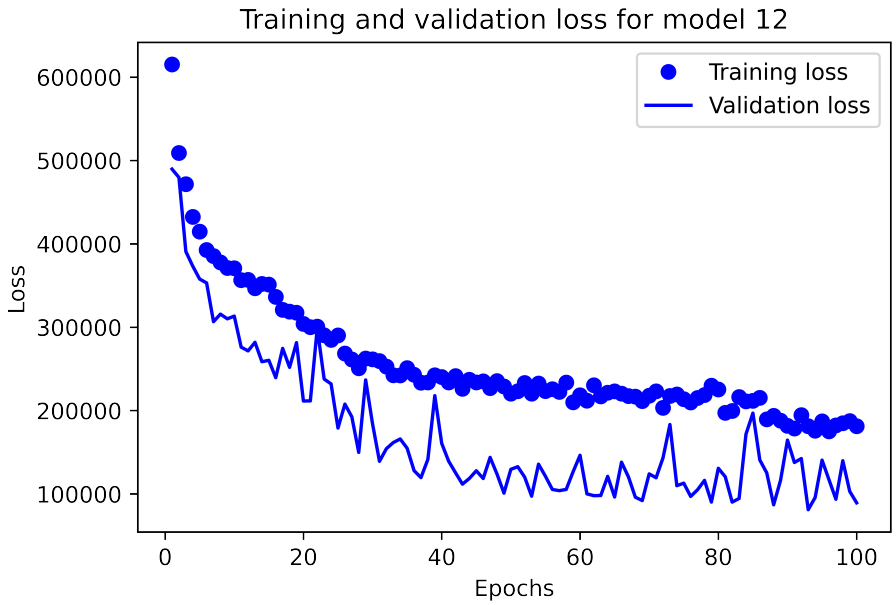
The incremental additions to the model seemed to work, so model 9 added 2 more layers, creating a 128-64-32-16-4 structure. This was more radical that the previous iterations because it added hidden layers in the front and the back at the same time. The performance was not good compared to model 8 and the overall increase in loss and validation loss spikes returned.



Model 10 was used to test whether if model 9's deficiencies could be fixed with a higher batch size, but this was not the case and the loss got worse.



Model 11 went back to the more successful model 8 (with a batch size of 20 and a 64-32-16-8 structure), but the number of epochs the model took to train increased from 20 to 50. The purpose was to see if longer training could improve performance by a big margin or whether the model was stuck with the performance of the current model. 50 epochs was a relatively good point to stop training because the model started to see diminishing returns around that point and the loss went down significantly compared to epoch 20, but the model found it's limit at epoch 50.



Model 12 tried to improve model 11 by increasing the generalizability and increasing the batch size to 30. This model again achieved a very low overall loss with the validation loss being lower than the training loss. This was about the best the model got in terms of achieving the lowest validation loss even though the loss was still pretty big at around 10,000.

Other attempts in modifying model 12 ended in about the same or worse performance so this model seems like the best performer that this analysis found. Other methods like increasing batch size to 100, adding a kernel regularizer to each dense layer, and adding layers all resulted in a lower performing model.

4 Analysis

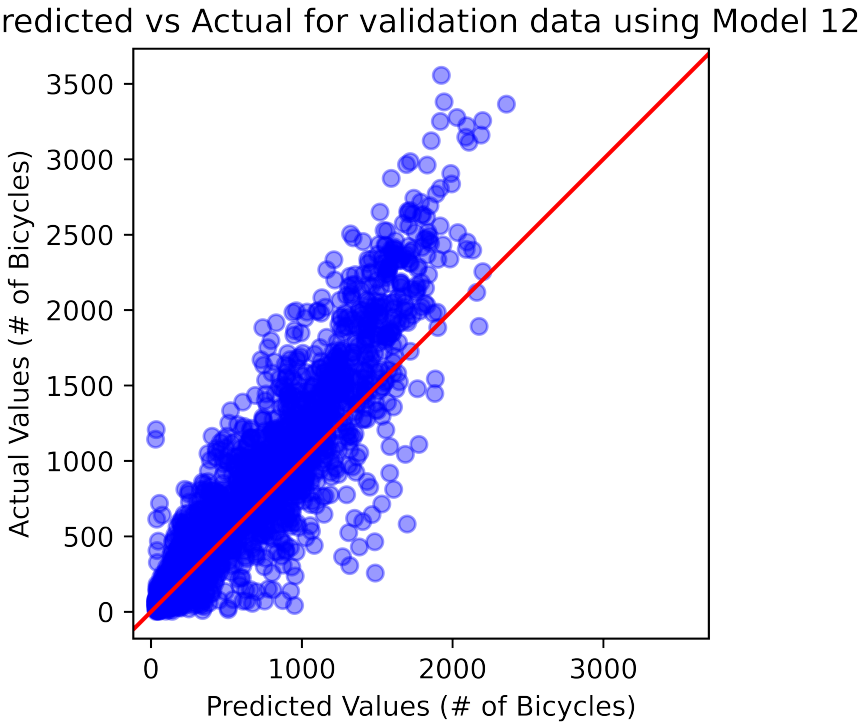


Figure 6: Predicted v.s. Actual

Figure 6 shows the how the predicted value of the number of bicycles rented and the actual number of bicycles rented compare using model 12 to predict the predicted values. The ideal situation is to have all the points on the red line to imply that the model correctly predicted all the actual numbers correctly. This is not the case, but the distribution around the line seems even which is a good sign. Also another observation is that there are significantly more data points near the bottom left then fade out towards the top right. This means that there is some imbalance in the dataset.

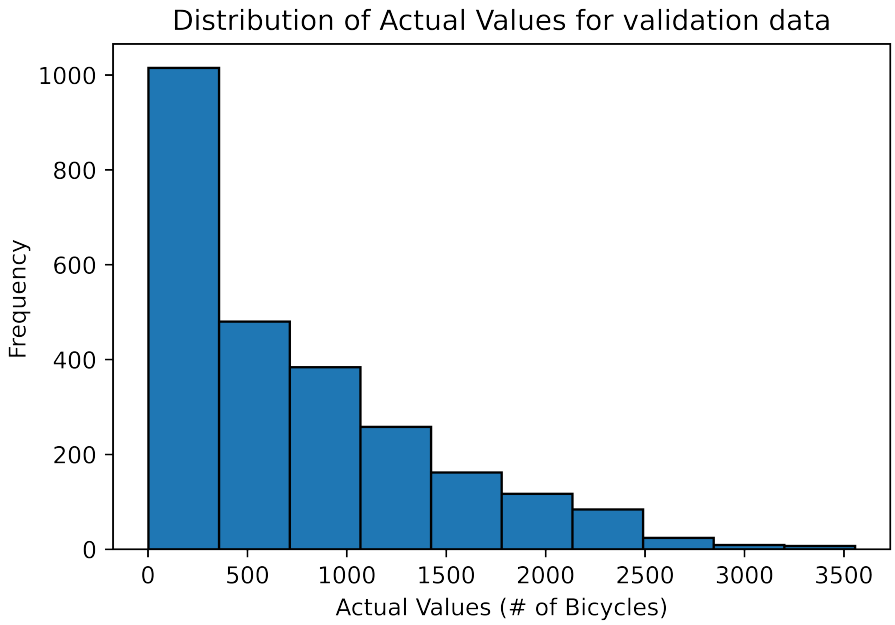


Figure 7: Data Distribution

Figure 7 shows the imbalance in the dataset. There are much more lower number of rented bikes than higher number of rented bikes. This might be due to some external factors like weather condtions but also might represent that there is a strong base of renters that use the service often with some of the users only renting once in a while. This also can explain why the model underperformed in certain cases where there were less data points to train on in certain areas of the data, especially the lower frequency data points. The model might also be skewed to predict the higher frequency points with higher accuracy than lower frequency points.

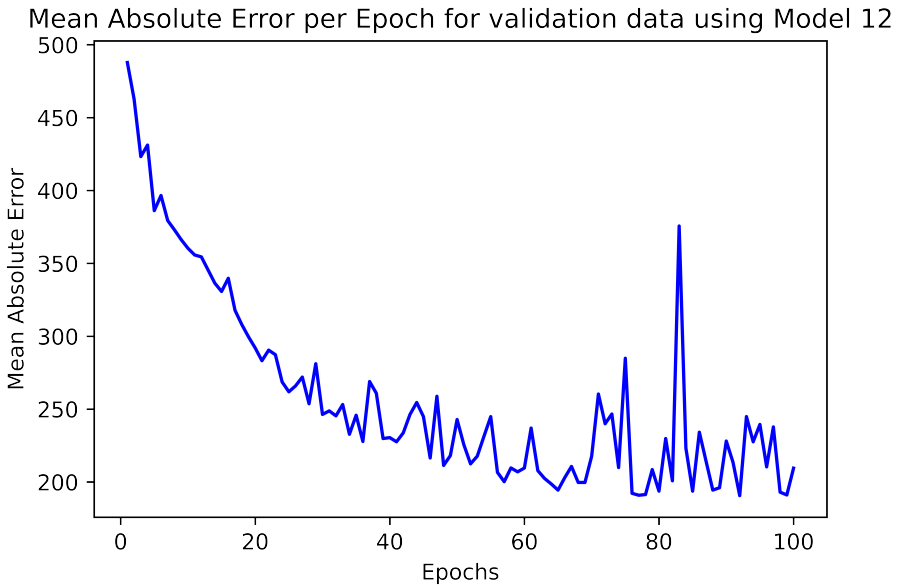


Figure 8: Mean Absolute Error

Figure 8 shows the mean average error for model 12 with the validation data across the epochs. There is a point that the mae meets where the model cannot generalize on new data.

5 Conclusion

The model using the Seoul Bicycle Sharing dataset was moderately able to predict the general number of people who would be renting a bicycle at a certain time. Since this is a seasonal time series dataset, it would also be interesting to compare to other methods where the time dimension can be preserved better.

6 References

The first 3 neural network examples and code were taken from Francois Chollet's Deep Learning with Python (1st Edition)

The Bicycle Sharing dataset was found on University of California, Irvine's Machine Learning Repository under the name "Seoul Bike Sharing Demand Data Set".

- Data Source :http://data.seoul.go.kr/
- SOUTH KOREA PUBLIC HOLIDAYS. URL: publicholidays.go.kr