Assignment Three RNN Stock Price Predictions (AMP Dataset)

DTSC13-301

Muiz Murad

13599863

Table of Contents

Executive Summary

The task was to build a model that had a low mean squared error but did not overfit or underfit. The desired outcome was to have a model with a small number of parameters, low mean squared error but a balance between its training and validation loss. The process in which to find the best model was based off a logical but sequential method in which multiple baselines models were created and refined, building a combined model and then fine tuning to build the best model. The problems faced within the task are based around infrastructure and resource availability, easier and more processing intensive methods are not able to be used as the machine being used is not able to process the code quickly and efficiently but rather take multiple days and become a more exhaustive and time inducive method. This will overcome by using smaller parameters in which the user will reduce the size of the model based on logical and educated thinking and methods. The final and best model to be used will be a combined CNN and LSTM model with 7265 parameters and Test MSE of approximately 0.005.

Problem Definition and Data Assembly

The input data for the model was the AMP.AX stock data using the low/close/adj. close/volume inputs. It is organized in 4 different columns being the Low, Close, Adj. Close and Volume. The aim is to predict the future stock prices of the model to provide an indication of the stock's trends and future values. The problem at hand is a supervised learning problem as we are detecting inputs and building a generative model. The dataset is independent and hence one output will not affect the other.

Evaluation Protocol Decision

For this case, we need to create a validation dataset to be used. As we will be guiding ourselves to build the best model we will need to evaluate the Mean Absolute Error, as we are using no metrics in this modelling process we need to ensure we have the best possible available metrics being used. We will use the Validation set to calculate the MAE on as if we calculated it on the Training set we wouldn't allowing ourselves to determine if the model was generalizing well on the validation or otherwise unseen data. We train the model on the training data set which has been normalized. The model evaluation technique we are using is the mean_absolute_erorr metric in the keras package. We can also use the loss function in the keras package to help us guide our optimization of the model and to reduce any over or underfitting. The loss function is a method of evaluation that tells us how well the algorithm can build a model from the given data. However, if the loss function is too large then it would indicate that the data presented to the model is not being predicted properly. For this dataset we will use the mean absolute loss function as we are trying to predict the future stock price, hence we will need a metric to tell us the difference between our predicted and true values.

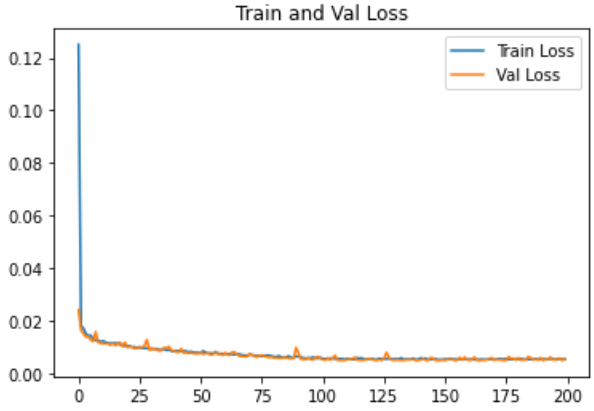
Preparation of Data

We load the data and then determine the shape of one sample, we do this to ensure keras is able to make sense of what type of data it will be using within the input layers of the neural network. After this step we complete two crucial steps to ensure the data can be read quickly and is normalized. Firstly, we scale the data which changes it to a value between 0 and 1 to allow the MAE metric to function correctly and then we reshape the data as the LSTM model is a three-dimensional model requiring 3 features to be used.

Development of Model To Beat Baseline

This is a crucial part of the process as we determine the best last-layer activation, loss function and optimization configuration for which the model will run. As we are dealing with a non-classification problem, we will not need to use an activation layer but will need to use mean absolute error as our loss function. Our optimizer will be adam as it is the easiest to use and combines the most valuable properties of the exisiting RMSProp and AdaGrad optimizers allowing for it to use an optimization fitting process even with noisy data.

To make a model to beat the baseline we will use a stacked LTSM Style model that will have a fixed set of parameters. As we can build a model that can beat the baseline, we know it is likely that the inputs are useful in the modelling process. We have created a basic model by putting together a simple two-layer LSTM Model which demonstrates an approximately a Validation MAE of 0.0056. Below we have a graph for the training and validation loss of this model:

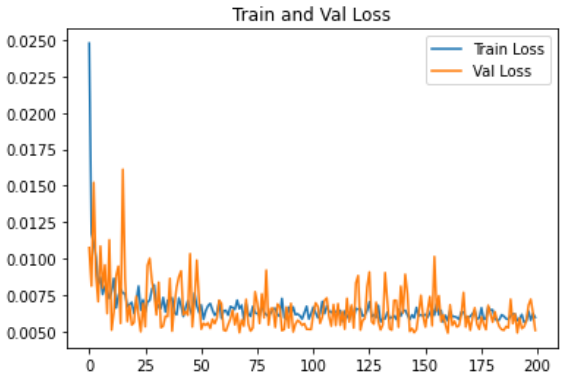


As we can build a model that can beat the baseline, we know it is likely that the inputs are useful in predicting the outputs. We have created a basic model by putting together a simple one-layer CNN which demonstrates an approximately 50% accuracy, this is shown in our graph below:

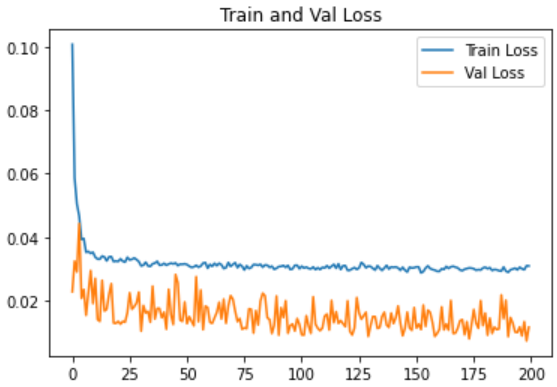
We will now further onto developing a model(s) that will demonstrate lower MAE's and clear signs of over fitting to indicate us how we need to create a balance between our inputs to achieve the best model to ensure it isn't under or overfitting extremely.

Scaling Up: Developing A Model That Overfits

Each of the models will use the same gradient function (Adam) as they are recommended as best practice for building a model with this type of data. We will start by developing a model with a learning rate to see the affect of that on the model, when adding a learning rate we are using a large learning rate we will increase the noise on the stochastic gradient. Learning rates have also been seen to make the model converge more quicker and hence cause a problem over fitting. When using the learning rate we can see that the model loss has become very nosiy as expected however the model is converging quicker which can be seen by the output below:



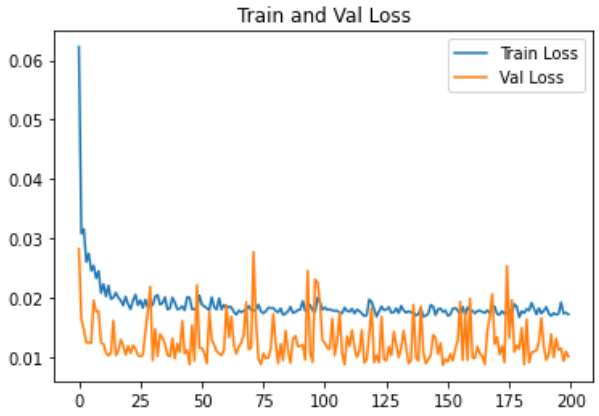
We will now investigate the increase of network complexity to see if the model is overfitting, we made a stacked LSTM model and also included dropout to reduce the noisy output we were provided with in the second model.



After increasing the network complexity and using a dropout rate which will randomly add and subtract neurons within in the model to generalize it better, we have been able to develop a model that is now underfitting significantly. As the aim was to reduce the noisy Validation and Loss graph, we have been able to do this with the change in optimizer to RMSProp which has made the validation and training loss much less noisy than previously. We will choose the first overfit model as our overfit guideline model and last one as a underfit model to provide us a guideline to as how we should tune the model further to provide optimal MAE whilst having a low network size.

Regularizing Models And Tuning Hyperparameters

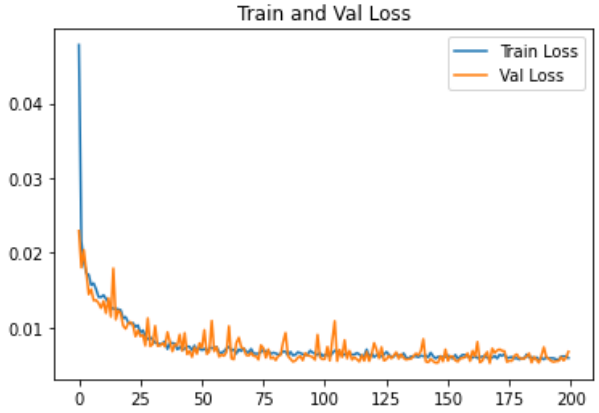
As seen in our literature reviews, we were able to see a clear trend in the best models that consistently were providing accurate results between the True and Predicted stock price. As we had completed this research, we were able to easily identify three models that were of use to us being the: CNN + LSTM, CNN + Attention + LSTM and the basic CNN model. CNN models were seen to be common in stock price predictions due to their ability to recognise features and patterns specifically to the input data allowing the model to generalize and perform better. To ensure we are keeping our process consistent, we will use a Batch Size of 32 and 200 epochs to ensure we able to determine the optimal cut off point.



We will start by developing a basic CNN model and investigate the results:

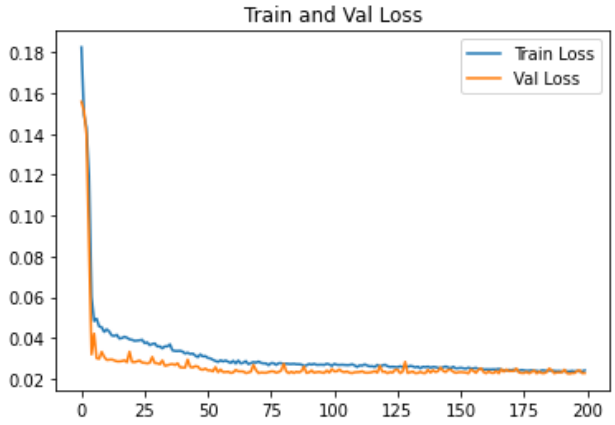
We are able to see that the CNN model is underfitting and the validation loss is noisy indicating fluctuations within the True and Predicted stock prices indicating that the model is providing inaccuracies, this model had an overall Val MAE of 0.01 which is quite high in comparison to our baseline model. We will now move onto using a CNN and LSTM combined model to see the affects.

The CNN and LSTM model results are shown below:



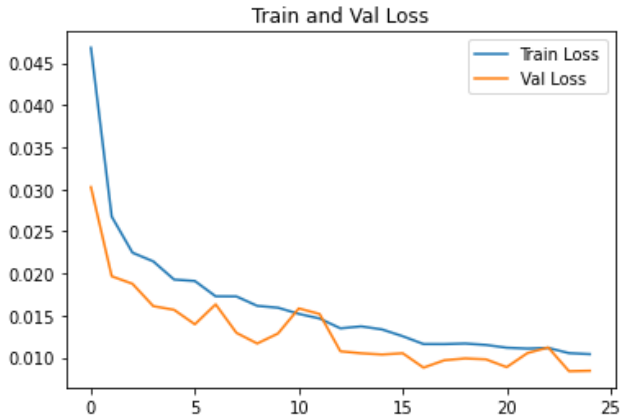
We can see that the model has significantly improved both with its Validation and Training loss and overall Validation MAE. We have been able to reduce the validation MAE to 0.0068 and the validation and training loss have converged meaning that the model is not over or underfitting indicating that this type of model is generalizing well on the stock data and the difference between the True and Predicted prices is not much. Our final model that was determined as a best fit is fusion between a CNN, Attention and LSTM model. We will use this as this model will have a low network complexity but will possess high capabilities.

The addition of the attention layer allows the model to retain all the input space and transfer learning and then make an output based on the retained knowledge, in this case we make the attention layer retain the past 60 days of data and then predict the next day's price. The results of this model are shown below:



The model can be seen to take a longer time to converge at around the 200th epoch, indicating that this model will take a large amount of computational power whereas we were able to see a better model be made with a less complex model. The overall Val MAE was o.0228 which is larger than the previous model. After the investigation of these models we can clearly see that the CNN and LSTM model is best fit for the AMP.AX data.

The CNN and LSTM model used will now be refined using manual techniques as we are restricted on the amount of computational power required for Grid Searching to automatically provide the best parameters. The manual tuning will be done by changing the network size, adding and dropping layers and also determining the bets number of epochs. This process is outlined in the code, but the results of the best model are shown below:

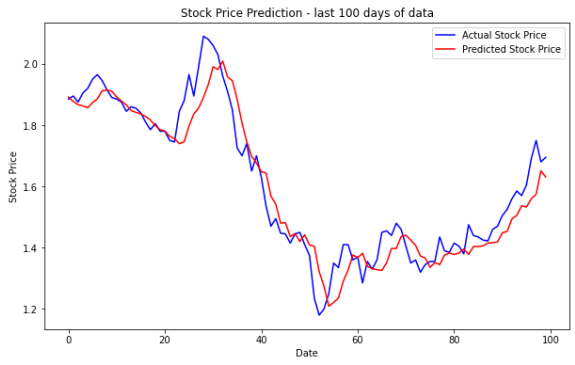


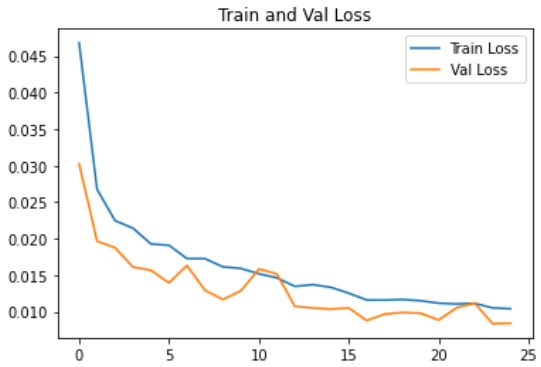
We can see that at the 25th epoch the model is converging indicating that the model is not over or underfitting and is providing best fit, however we should also note that a large dropout rate was also present within this model. The dropout rate helped us better generalize the data and reduce our initial overfitting problem.

Overall, the best model was the most basic model and as mentioned the best way to predict tomorrow is by predicting today in the world of numbers and this is what we saw throughout our process. Complexity will only make the model learn too many specific details about the certain numbers it provided within the data set and not be able to generalize to different Stock Exchange Listed Companies. The CNN layers and LSTM help the model learn the specifics and generalize at the same time whilst minimising network parameters to ensure the best model is made.

Best Model Visualization

We can see from the stock price prediction graph below that, for the past 100 days the model was predicting quite well as seen by the red line in the graph. The blue line represents our True Stock price. As we can now determine that our most non-complex model was the best, the best way to predict stock prices is by focusing on the data we have now. The training and validation loss also indicated the model was generalizing quite well as seen by the prediction graph.





Best Model Metrics

As we can see our best model had a Test MSE of 0.005 which is significantly low and only had a total of 7265 parameters. It had 6 layers whom of which the LSTM was the largest due to its advanced capabilities and memory neurons.

```
Val MAE: 0.007738947402685881
Test MSE: 0.005211040570324315
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_29 (Conv1D)           (None, 58, 16)            64
_________________________________________________________________
max_pooling1d_29 (MaxPooling (None, 29, 16)            0
_________________________________________________________________
lstm_31 (LSTM)               (None, 29, 32)            6272
_________________________________________________________________
dropout_11 (Dropout)         (None, 29, 32)            0
_________________________________________________________________
flatten_8 (Flatten)          (None, 928)               0
_________________________________________________________________
dense_34 (Dense)             (None, 1)                 929
=================================================================
Total params: 7,265
```