Author: Ilari Pajula – *NOTE: Report written using price as only feature. GitHub code includes an example of more training features using technical indicators.*

Using Linear-, Dense-, and Convolutional Neural Networks for Forecasting Stock Prices

1. Introduction

Recently, there has been a surge in the amount of individual investing of financial securities like stocks and cryptocurrency. A well-known use of machine learning has always been time series forecasting, and one of the most classic applications of time series forecasting has been predicting fluctuations in a stock's price value. This project report will investigate how neural network (NN) models can be used to make predictions for a stock's future price using multiple-output prediction [1] in section 2, section 3 looks at the three candidate NN models that will be used. Results from training, validation and testing are shown in section 4, and a brief explanation and conclusion is provided in section 5. While this project idea is not original in a sense, I wanted to take it as an opportunity to learn more about neural networks on a subject that interests me. The findings from this project could be used to support real investing activity.

2. Problem Formulation

Predicting future stock prices given historical price data is an example of regression predictive modelling. One quite complex way of using NNs to do this is by using single-shot models [2] for prediction. Single-shot models in time series forecasting get trained on an initial window of time, called the training window. Then they make predictions for multiple datapoints in a window of time subsequent to the training window, called the validation window.

Our application can be modelled using datapoints that represent trading days of a single stock ticker (eg. AAPL). The features will be composed of the stock's closing price on days in the training window. The labels, or quantities of interest, in this problem formulation will be the price of the stock on trading days in the validation window.

The stock ticker chosen will be AAPL (Apple Inc.) and datapoints used for training and validation will be from the trading days in the timeframe 01.01.2015 – 27.03.2021. This data is freely available from Yahoo Finance [3].

3. Method

The raw data consists of 1569 datapoints and will use a training/validation data spilt of around 60/40 (%), but will also reserve one window as a test dataset. The data preprocessing implements a sliding window cross validation method [4] that creates subsets of the data by shifting the training and validation window by a constant amount for each data subset. This is a good way of reducing the impact of limited training and validation data, while also preserving the order of datapoints. Choosing a window length of 170 days for the training window and 60 days for the validation window, we end up with a full window length of 230 days. Setting the offset of the sliding window as 60 days we can increase the number of windows we perform training and validation with.

Below is an example of what the features and the labels look like in one window (data has been normalized with the mean and standard deviation of the training dataset). Here it is also easy to see what that training and validation windows represent. Note that the labels are invisible to the model until the loss is calculated.

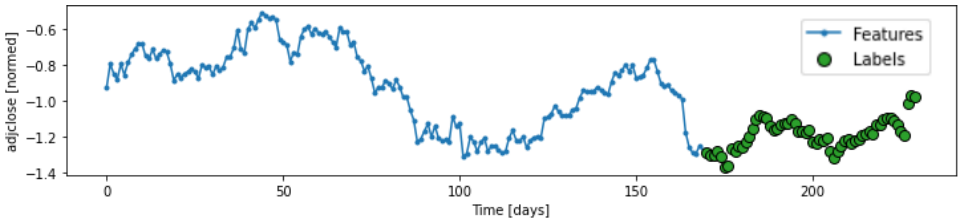


Figure 1: Example of a window with training and validation spilt.

Author: Ilari Pajula – *NOTE: Report written using price as only feature. GitHub code includes an example of more training features using technical indicators.*

Mean absolute error (MAE) [5] is used as the loss function in this implementation because of it ease to implement. It is calculated by taking the sum of the absolute value of the differences in predictions and labels over the number of labeled datapoints. The number of labeled datapoints in one window is 60.

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Models

The three models that will be used for this data in order of increasing complexity are a linear neural network, a dense neural network, and a convolutional neural network. All models were implemented and trained with the Tensorflow and Keras Python modules. Training was performed for a maximum of 100 epochs, but early stopping of training occurred when least 5 successive epochs had progressively worse validation error.

A linear neural network is one without an activation function in its layers. It uses one densely-connected neural network layer between the input and output layers. This dense layer has no activation function, which reduces its prediction capability to that of a linear model. This model makes a prediction for 60 days using a linear projection from a single input timestep.

A dense neural network here is very similar to the linear neural network implementation, except the densely-connected neural network layer uses a rectified linear unit activation function (ReLU) between the input and output layers. This gives the model ability to learn much more complex, non-linear, information about the data. But the model is still limited in that it can only take in one timestep as an input.

The convolutional neural network (CNN) implementation here uses a single one-dimensional convolutional neural network layer which uses ReLU activation functions between its input and output layers. It is also different from the linear and dense networks because as an input it takes in multiple inputs, instead of one input. The CNN implementation in this project uses a fixed-width of 10 days for the input.

Below are example graphical representations [2] of the models inputs and outputs. The example figures show a case where there are 24 inputs, and 23 outputs (predictions). The linear NN and dense NN are represented by figure 2. Both these models only use the most recent input timestep (t=23) to create the predictions. The CNN model works by taking a fixed number of inputs, 3 in this example, to make predictions. This means that the CNN model is the only model with an input that covers a change over time.

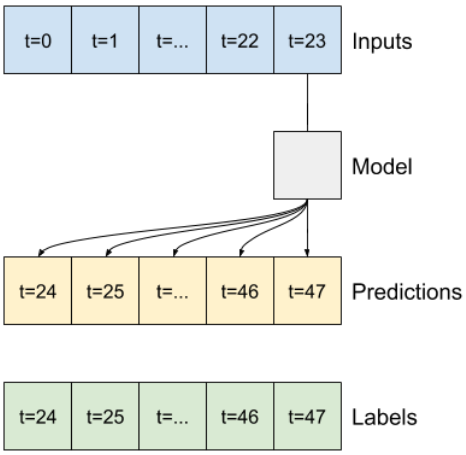


Figure 2: Linear- and Dense NN Graphical Representation

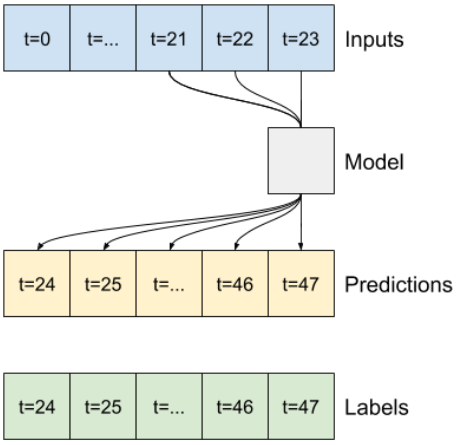


Figure 3: CNN Graphical Representation

Author: Ilari Pajula – *NOTE: Report written using price as only feature. GitHub code includes an example of more training features using technical indicators.*

4. Results

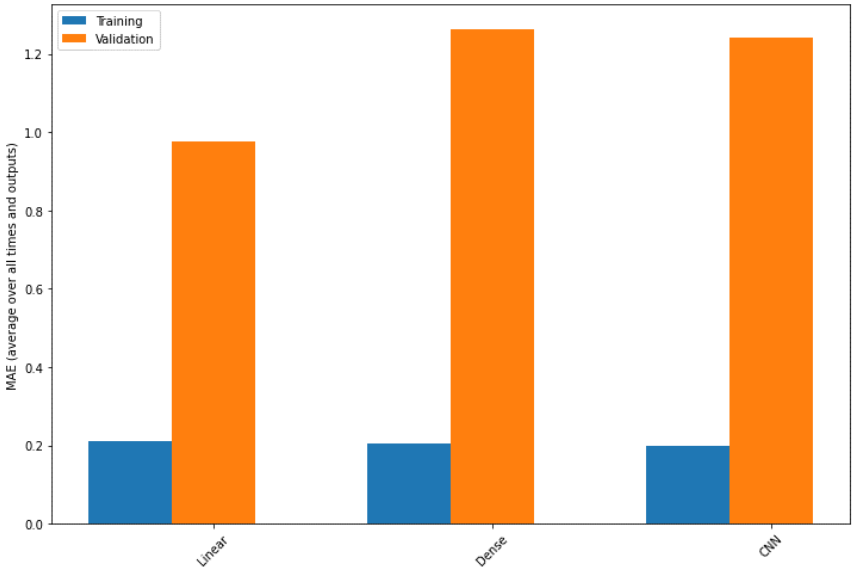
During training and validation stages, we use the sliding window cross validation method. The total loss reported in the table below is calculated by taking the sum of the MAE in each window of a dataset.

Table 1: Training and Validation Losses

	Linear NN	Dense NN	Convolutional NN
Training	0.2100	0.2047	0.2000
Validation	0.9773	1.2619	1.2413

This is easy to visualize as a bar plot. The CNN model achieves the lowest training loss, but the Linear NN model achieves smaller validation loss.

Figure 4: Training and Validation Losses



It is also possible to visualize and compare some of the predictions that that three models made on a test set. In the figures below a test window is used to compare the predictions made by each model to the labels. In the test window the dense model performs the best according to the MAE value, which is unlike the validation results that were obtained where the dense model performed the worst.

Linear NN

MAE = 0.03463

Dense NN

MAE = 0.02210

CNN

MAE = 0.02987

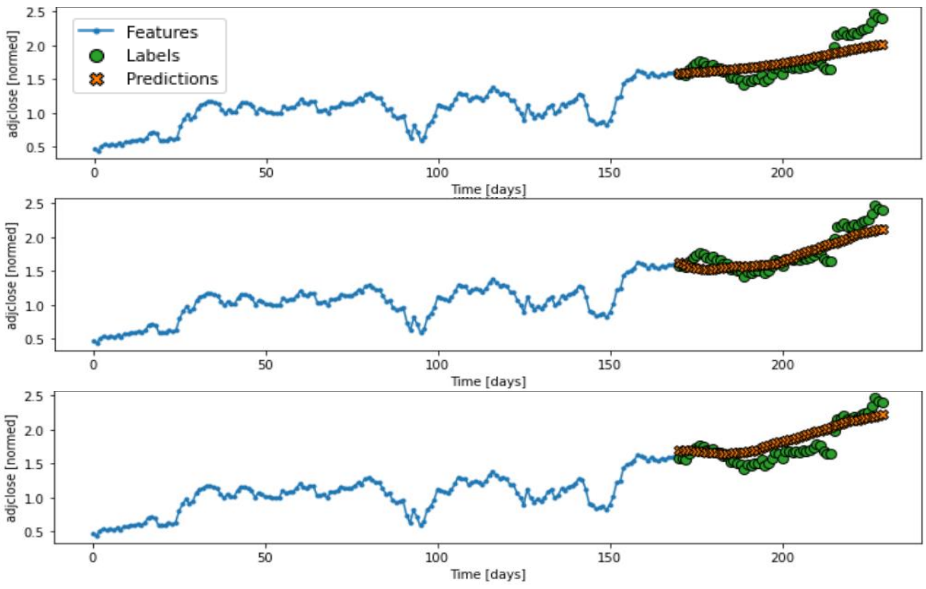


Figure 4: Performance on a test window.

Author: Ilari Pajula – *NOTE: Report written using price as only feature. GitHub code includes an example of more training features using technical indicators.*

For fun, we can also test the models on the current market by making predictions for the future price of AAPL by inputting the latest 170 days of price data into our model. Now I just need to wait 60 days months to collect the MAE values on these predictions. In this figure, the real price is on the y-axis and it has been reverted from its normalized version.

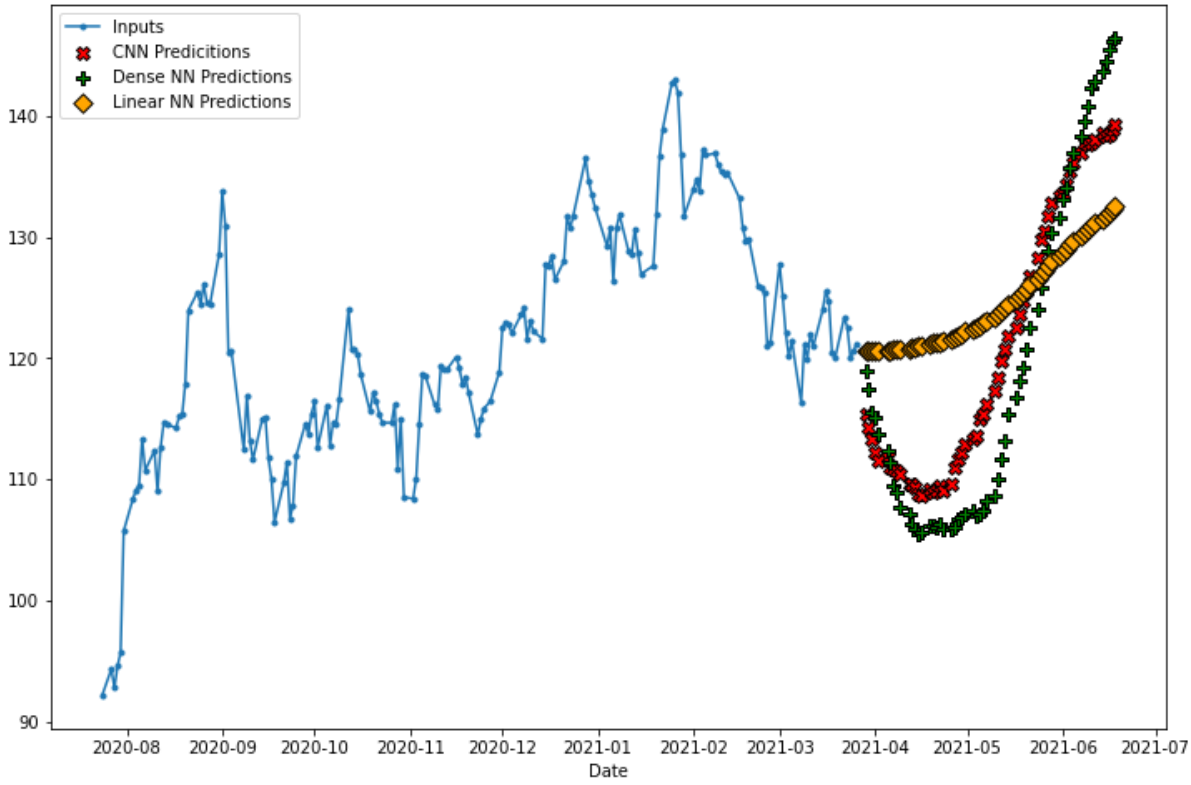


Figure 5: Example of predictions for the current market

5. Conclusion

This project report studied three different models for learning a hypothesis that would predict the price of a stock for 60 days given 170 days of historical price data for that stock. All three models were different implementations of neural network models with one hidden layer only.

All three models were able to get very similar values for training error. A reason for this could be that the models were all given a very large number of maximum epochs (100) to fit the data. The linear NN is able to produce the model with the smallest validation error, despite having the smallest complexity of the three models. This would indicate that the Dense NN and CNN models tend to overfit the data and make assumptions that the loss function penalizes them for. Visually, the linear model makes predictions that resemble the moving average of the labels with very small variance between predictions, while the Dense NN and CNN have larger variance. This is probably why the linear NN has a smaller validation error on the larger validation dataset, but by chance managed to get slightly worse results on the testing window.

For the sake of this report, data was limited to only the AAPL ticker and using a small amount of datapoints for which my computer could still manage the training for. Further exploration of this subject would include more data from other tickers and longer timeframes from which to sample training windows.

In conclusion, choosing the Linear NN to model the hypothesis is justified, because it achieves better results with the validation dataset.

Author: Ilari Pajula – *NOTE: Report written using price as only feature. GitHub code includes an example of more training features using technical indicators.*

6. References

[1] - Tensorflow, Time Series Forecasting, multiple-output models, Last accessed 27.03.2021, https://www.tensorflow.org/tutorials/structured_data/time_series#multi-output_models

[2] – Tensorflow, Time Series Forecasting, single-shot models, Last accessed 27.03.2021, https://www.tensorflow.org/tutorials/structured_data/time_series#single-shot_models

[3] – Yahoo Finance, Apple Inc. (AAPL) Historical Data, Last accessed 27.03.2021, https://finance.yahoo.com/quote/AAPL/history?p=AAPL

[4] – Pradip Samuel, Medium, CROSS-VALIDATION IN TIME SERIES MODEL, Last accessed 27.03.2021, https://medium.com/@pradip.samuel/cross-validation-in-time-series-model-b07fbba65db7

[5] – Wikipedia, Mean Absolute Error, Last accessed 27.03.2021, https://en.wikipedia.org/wiki/Mean_absolute_error