Predicting stock indices using neural networks

Marcus Enström
menst@kth.se
Axel Alness Borg
axelborg@kth.se
Hafsa Aoutir
hafsaa@kth.se

Abstract

The competitive nature of the financial markets and more specifically considered in this paper, the stock market makes it an interesting candidate for machine learning. The efficient market hypothesis (EMH) states that the current price of a stock reflects all available information, and to consistently "beat the market" and generate an edge is impossible. This hypothesis has however been proven to incorrect in modern financial markets. The task of this paper is to predict whether following day's close price of an index will be higher or lower than the current day given 60 days history of data consisting of both of stock index data but also currency and commodity prices using convolutional neural network.

1 Introduction

The goal with this project is to study whether or not it is possible to gain an edge in the stock market using neural networks to predict future returns using historical data. This study will attempt to predict whether or not the a stock index will move up or down the next day given 60 days of various historical financial data. The rest of this article is organized as follows: In section 2 related work and research are presented. Then in section 3, we presented data used for the study. In section 4, the methods are being presented in details: CNNPred-2D, Stacked CNNPred-2D, LSTM, CNN-LSTM. In section 5, Experiments and relative results are described and finally a conclusion in section 6.

2 Related Work

There are three approaches for stock market predictions: technical analysis, traditional time series forecasting and machine learning methods such as linear regression, exponential smoothing. Currently, support vector machines (SVM) and Artificial Neural Networks (ANN) are more commonly used for the prediction of stock market.

In [3], authors used SVM at time t to predict whether a given stock's price is higher or lower on day t +m. The results showed that the model is able to achieve significant prediction accuracies with some parameters in the long-term, however to achieve prediction accuracies in the short-term we must look at more granular intra-day trading data.

ANN is considered the main machine learning technology in the field, it has been applied to model various challenging problems in engineering and science. However, a difficult task in ANN design is the selection of the suitable number of units that are large enough to fit the purpose but not too large that the ANN over-fit.

However, in [1] Authors discussed stock price index movement using two models based on Artificial Neural Network(ANN) and Support vector machine(SVM) and it has been proven that the performance of ANN model is better than the SVM model.

There are several type of networks that can be used in the design of ANNs: (feed-forward, recurrent, back-propagation): An example of Feedforward neural network is [5] where authors conducted a study where the ability of artificial neural network in forecasting the daily NASDAQ stock exchange rate was investigated, they used several feed-forward ANNs trained by the back propagation algorithm.

Recurrent neural networks are a class of neural network where previous outputs can be used as inputs while having hidden states. LSTM is one of the most common kinds of RNNs. In [4] a combination of Convolution Neural network (CNN) and Long–Short-Term Memory Neural Network (LSTM) is used to acheive a more accurate prediction of the stock price. Results showed that when compared to previous methods it leads to better resutls.

In [2], two methods using convolutions neural networks were proposed to binary classify the following day's closing price based on the two classes, higher or lower than the current day. One of their proposed method uses a 2D convolution while the other method uses a 3D convolution to extract features from the data and use those for prediction.

The methods we discussed so far are all using algorithms that improve the prediction model. However the performance of prediction can also be enhanced by improving the features used to predict the model, for instance authors of in [7], before applying ANN for prediction, they used PCA and two variations to extract better features. The results showed that there is an improvement of prediction when using the features generated by PCA compared to the two other variations.

3 Data

For simplicity we use the same data used in [2] which had been made easily available and processed. This contains data from 2010-2018. For us to be able to fairly compare our results to [2] we chose the same data split that they did, that is, data after 2016-04-21 is used for testing. The rest of the data up until 2016-04-21 is split into training and validation with the rations 75% respective 25%. The data consists of the 5 major stock indices in the US: Dow Jones Industrial (DJI), NASDAQ 100, New York Stock Exchange (NYSE), Standard & Poor's 500 (SP) and Russel. The data also contains price changes of commodities, foreign exchange rates and various bond yields. For specific details, please refer to to the Appendix I in [2]. The relative daily change of the indices and the datasplits are shown in Figure 3.

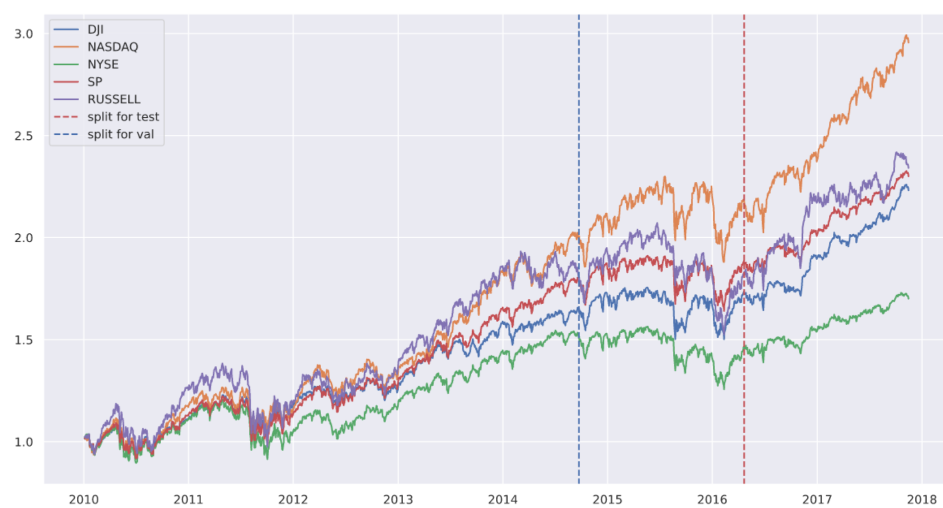

Figure 1: Relative price change of indices in data set

One interesting dynamic of this data set is that the different splits contain quite a bit of different trends. The validation set overall moves more 'sideways' than the training set and testing set.

4 Methods

4.1 CNNPred-2D

The models implemented in [2] was implemented to ensure that we were able to achieve at least similar results. This was relatively easy, considering that they had made the implementation publicly available on GitHub. We did however notice some flaws with their implementations, so our implementation of the CNNPred-2D differs slightly. An exhaustive list of changes is presented below.

- When standardizing the datasets [2] did not scale the validation data and test data with the same means and standard deviations as the training data, but rather scaled each set separately (impossible in real world application). This is considered bad practice and is thus corrected in our work.
- We changed the loss function from mean absolute error to binary cross-entropy as this resulted in better stability for loss as well as F1 & accuracy metrics.
- Their model struggled with heavy over fitting, and we thus added both Gaussian noise as well as l2 regularization for each layer.

4.2 Stacked CNNPred-2D

Due to the stochastic nature of neural networks it is often the case that the trained CNNPred models will differ quite a bit in their predictions. We propose a method which uses several trained CNNPred models in an ensemble to reduce the variance in prediction. The method is often referred as 'stacking' several networks, and comes in different forms. We initially train 10 CNNPred models using training data. We then use these trained networks to predict targets using training data. These training predictions are then used as a new training set for a logistic regression model in combination with the actual training targets.

Once the logistic regression model has been trained using training data only we can proceed to evaluate the validation and test data by generating predictions from each of the 10 CNNPred networks and feeding them to the regression model for the final predictions. An overview of the proposed architecture is shown in Figure 3.

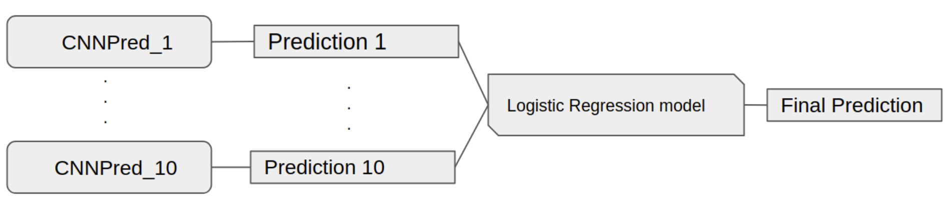

Figure 2: Overview of the proposed stacked CNNPred framework

4.3 LSTM

We also wanted to investigate whether using a LSTM could better capture the time aspect of the data and therfore produce better predictions. To test this we first proposed a simple LSTM with 10 hidden units and then a fully connected layer for the output.

4.4 CNN-LSTM

We also propose a combination of CNNs and a LSTM as done by [4]. The proposed model combines the CNN structer of CNNPred-2D with the addition of batch normalisation in between layers to avoid exploding and diminishing gradients during learning. Then instead of having a fully connected layer as done in CNNPred-2D we proposed adding a LSTM with 5 hidden layers after the convolutions. Then we attached a fully connected layer for the output.

5 Experiments

5.1 CNNPred-2D

The results of the CNNPred-2D model are shown in Table 1. The training progress is discussed more in Appendix A and a graph of the evolution of metrics during training is shown in Figure 4. The achieved results are very similar to those presented in [2]. For example for the S&P data set they report a mean accuracy of 0.4799 while we achieved 0.4754.

Dataset	model	datasplit	F1 mean	F1 std	F1 max	F1 min
DJI	CNNPred 2-D	test	0.4667	0.0375	0.5432	0.3791
		val	0.5303	0.0362	0.6156	0.4549
NASDAQ	CNNPred 2-D	test	0.4812	0.0246	0.5269	0.4024
		val	0.5339	0.0420	0.6234	0.4610
NYSE	CNNPred 2-D	test	0.4613	0.0337	0.5491	0.3998
		val	0.5438	0.0444	0.6725	0.4622
RUSSELL	CNNPred 2-D	test	0.4776	0.0302	0.5310	0.4005
		val	0.5547	0.0367	0.6234	0.4603
SP	CNNPred 2-D	test	0.4754	0.0257	0.5169	0.3998
		val	0.5356	0.0408	0.6277	0.4643

Table 1: Results of CNNPred-2D

5.2 Stacked CNNPred-2D

The results of the stacked version of CNNPred-2D are shown in Table 2. These results show a huge increase in performance, all models achieve an F1 mean of > 0.53.

Dataset	model	datasplit	F1 mean	F1 std	F1 max	F1 min
DJI	CNNPred Stacked	test	0.5415	0.0165	0.5691	0.5234
		val	0.5440	0.0450	0.5986	0.4690
NASDAQ	CNNPred Stacked	test	0.5814	0.0213	0.6015	0.5435
		val	0.6107	0.0314	0.6667	0.5767
NYSE	CNNPred Stacked	test	0.5408	0.0285	0.5775	0.4955
		val	0.5469	0.0708	0.6536	0.4394
RUSSELL	CNNPred Stacked	test	0.5342	0.0256	0.5722	0.4956
		val	0.5496	0.0298	0.5816	0.4961
SP	CNNPred Stacked	test	0.5483	0.0254	0.5775	0.5125
		val	0.5282	0.0244	0.5556	0.4930

Table 2: Results of stacked CNNPred-2D

To visualize if the stacked model performs especially well in a certain market environment we choose to plot weather or not a prediction was actually correct along with the relative price change in the index. The stacked model with the highest validation accuracy was chosen and predictions were made on the test set. The green dots for a given day indicate that a correct classification was made while a red dot indicates an incorrect classification. The scatter plot should be read as a 1D plot but has been given a scatter in the y-axis to make it more readable. It seems that the distribution over time of correct classifications is independent of time which is a desired property when predicting indexes as a rapidly decaying pattern makes it hard to predict the future.

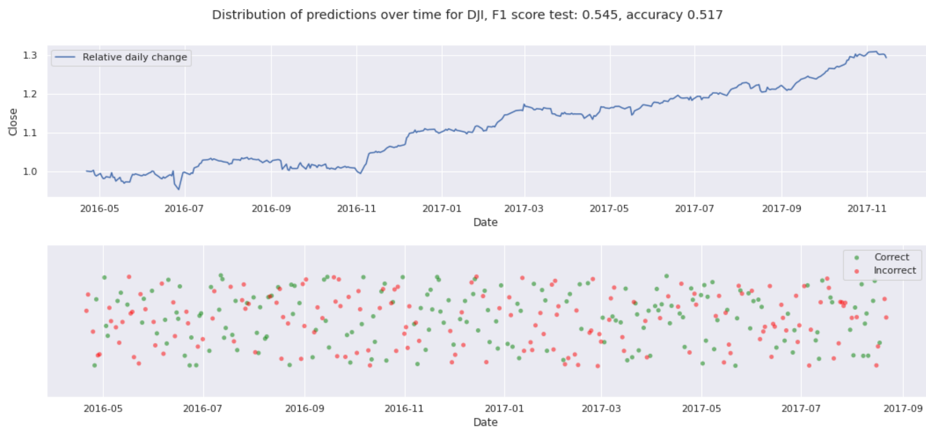

Figure 3: Distribution of predictions over time

5.3 LSTM

The results of of the LSTM model can be seen in Table 3. Theses results varies a lot and generally achieves a worse accuracy than CNNpred-2D.

			F1 mean	F1 std	F1 max	F1 min
Dataset	model	datasplit				
DJI	CNNPred Stacked	test	0.4081	0.0653	0.4952	0.3130
		val	0.5454	0.0288	0.5790	0.4992
NASDAQ	CNNPred Stacked	test	0.4227	0.0842	0.5385	0.2839
		val	0.5193	0.0311	0.5710	0.4357
NYSE	CNNPred Stacked	test	0.3862	0.0548	0.4949	0.4955
		val	0.5599	0.0.0229	0.5815	0.5095
RUSSELL	CNNPred Stacked	test	0.4391	0.0612	0.5079	0.3130
		val	0.5145	0.0241	0.5458	0.4683
SP	CNNPred Stacked	test	0.3945	0.0803	0 .5389	0.3116
		val	0.5387	0.0154	0.5584	0.5118

Table 3: Results of LSTM

5.4 CNN-LSTM

The result of the CNN-LSTM can be seen in Table 4. The results varies a lot between the best and worst F1 score. The average accuracy is worse than for CNNpred-2D but for some of the datasets the best F1 score is better.

Dataset	model	datasplit	F1 mean	F1 std	F1 max	F1 min
DJI	CNNPred Stacked	test	0.4335	0.0771	0.5591	0.3157
		val	0.4830	0.0231	0.5381	0.4668
NASDAQ	CNNPred Stacked	test	0.3742	0.0654	0.4939	0.2878
		val	0.4770	0.0320	0.5230	0.4234
NYSE	CNNPred Stacked	test	0.4277	0.0771	0.5592	0.3157
		val	0.5011	0.0.0231	0.5381	0.4668
RUSSELL	CNNPred Stacked	test	0.4408	0.0465	0.5113	0.3596
		val	0.4837	0.0369	0.5309	0.4212
SP	CNNPred Stacked	test	0.4014	0.0717	0 .5207	0.3116
		val	0.4817	0.0202	0.5062	0.4311

Table 4: Results of CNN-LSTM

6 Conclusion

As expected, it is very difficult to train a network to predict movements in the stock market. We estimate that the vast majority and the most successful of research regarding the subject is not published as it is done with the goal of maximizing gains within privately owned "quant" funds. As patterns are discovered and published they will be implemented and seize to exist if there is any value to be extracted.

The results shown by the proposed Stacked CNNPred model are, although far from perfect, impressive considering that the training size is relatively small with quite a high dimensionality. Our results show that using a combination of CNNs and LSTMs does not significantly improve the predictive power in a classification setting compared to only using a CNN network. During training we noticed that the models generally tended to overfit so quite a lot of regularisation was needed. If more training data had been used the need for regularisation could have been lower and possibly benefited the more complex models that most likely overfits more easily.

References

[1] Yodele et al. *Stock price prediction using neural network with hybridized market indicators*. 2012.

[2] Ehsan Hoseinzade and Saman Haratizadeh. *CNNPred: CNN-based stock market prediction using several data sources*. 2018. arXiv: `1810.08923 [cs.LG]`.

[3] Saahil Madge. *Predicting Stock Price Direction using Support Vector Machines*. 2015.

[4] Wu Jimmy Ming-Tai et al. "A graph-based CNN-LSTM stock price prediction algorithm with leading indicators". In: (2021). URL: `https://link.springer.com/article/10.1007/s00530-021-00758-w#Sec2`.

[5] Amin Hedayati Moghaddam, Moein Hedayati Moghaddam, and Morteza Esfandyari. "Stock market index prediction using artificial neural network". In: *Journal of Economics, Finance and Administrative Science* 21.41 (2016), pp. 89–93. ISSN: 2077-1886. DOI: `https://doi.org/10.1016/j.jefas.2016.07.002`. URL: `https://www.sciencedirect.com/science/article/pii/S2077188616300245`.

[6] Samuel L. Smith et al. *Don't Decay the Learning Rate, Increase the Batch Size*. 2018. arXiv: `1711.00489 [cs.LG]`.

[7] X.Zhong and D.Enke. "Forecasting daily stock market return using dimensionality reduction". In: (2017).

A Appendix A - Training CNNPred-2D

Figure 4 shows the evolution of the binary cross entropy loss, the accuracy of predictions and the F1 score for both the training set and the validation set. Due to our relatively small training size the entire training set was used as batch-size. [6] a larger batch size yields a greater stability during training. This aligns with our experiences while constructing our experiments.

One crucial decision is to decide when to stop training, often referred to as the bias-variance trade off. We decided to use early stopping with the F1 validation as measure due to the fact that validation F1 tended to increase even after the validation loss had reached its minimum.

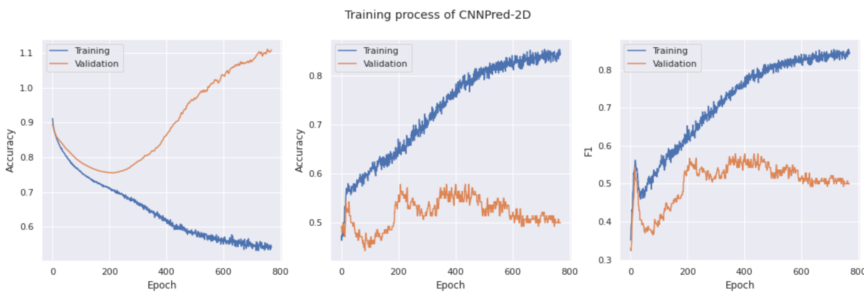

Figure 4: Evolution of loss, validation, and F1 during training of a CNNPred-2D model

B Appendix B - A toy trading strategy

Constructing an algorithmic trading strategy is a difficult task with endless pitfalls and surely a task beyond the scope of this project. However, for those interested we have constructed a toy example. Three different long-only ’portfolios’ are constructed. Each day a choice is made to go all in in the Dow Jones Industrial index or not upon market opening. The position is then closed the same day during market closing. The three different portfolios are as follows

- Stacked CNNPred 2D strategy: Using the best performing model on the validation set generate predictions. If the prediction is up, buy at open sell at close.

- Stocks only go up strategy: Buy at open every single day and sell at close.

- The gambler strategy/Random walk: Flip a coin and buy at open if it lands on head and sell at close.

Note that this example completely ignores transaction fees and the transactions are assumed to be small enough to not affect prices upon entry and exit in the market.

The performance based on the test period of our data set of each strategy is shown in Figure 5. Obviously these three strategies are all beaten by a buy and hold strategy due to an upwards trend in the data set and none of these strategies are able to capture overnight gains.

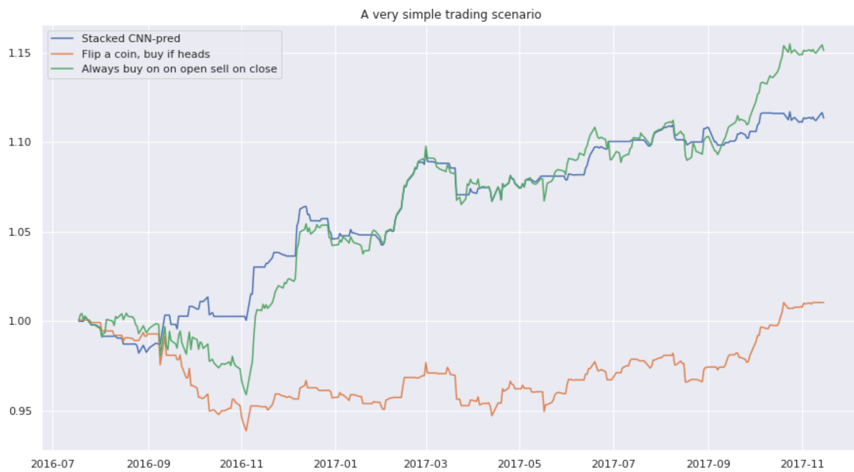

Figure 5: Toy trading strategies

Although the CNNPred strategy is outperformed by a 'buy & sell everyday' strategy in terms of cumulative gains it offers lower volatility (standard deviation of daily returns adjusted for 252 trading days a year). As shown in Table 5 it also offers a higher Sharpe ratio, which is often considered as risk adjusted return.

Portfolio	Volatility	Sharpe
CNNPred 2D	0.0444	1.834
Buy & sell every day	0.0613	1.742
Coin flip strategy	0.0475	0.177

Table 5: Volatility and Sharpe for different strategies

This toy trading example was not created to prove an edge in the market, as this would require more intense testing, but rather to give the reader an intuition of of how the model performs.