CS230

Stock Market Prediction using CNN and LSTM

Hamdy Hamoudi
SUNet ID: hhamoudi
Stanford University
hhamoudi@stanford.edu

Mohamed A Elseifi
SUNet ID: melseifi
Stanford University
melseifi@stanford.edu

Abstract

Starting with a data set of 130 anonymous intra-day market features and trade returns, the goal of this project is to develop 1-Dimensional CNN and LSTM prediction models for high-frequency automated algorithmic trading. Two novelties are introduced, first, rather than trying to predict the exact value of the return for a given trading opportunity, the problem is framed as a binary classification with the positive class selected as the trades resulting in returns in the top ten percentile of all returns in the training set. Furthermore, the 130 anonymous features are augmented with a logical matrix to reflect the missing data values at each time step, thus preserving any relevant information from the fact that a given feature is missing from a given record. The models are compared using both machine learning accuracy measures and investment risk and return metrics. Two CNN and three LSTM candidate models differing in architecture and number of hidden units are compared using rolling cross-validation. Out-of-sample test results are reported showing high average return per trade and low overall risk.

1 Introduction

Accurate prediction of stock market returns is a challenging task due to the volatile and nonlinear nature of those returns. Investment returns depend on many factors including political conditions, local and global economic conditions, company specific performance and many other, which makes it almost impossible to account for all relevant factors when making trading decisions [1], [2]. Recently, the interest in applying Artificial Intelligence in making trading decisions has been growing rapidly with numerous research papers published each year addressing this topic. A main reason for this growing interest is the success of deep learning in applications ranging from speech recognition to image classification and natural language processing. Considering the complexity of financial time series, combining deep learning with financial market prediction is regarded as one of the most exciting topics of research [3].

The input to our algorithm is a trade opportunity defined by 130 anonymous features representing different market parameters along with the realized profit or loss on the trade in percentage terms. Rather than using regression models to predict the percent return on a given trade opportunity, we decided instead to frame the problem as a binary classification one. First a target column is added to the training data with the trades in the top 10 percentile of all trades in terms of percent return marked as the positive class, while the remaining trades are marked as negative (either losers or small winners). Rather than trading every opportunity identified as a probable winning trade, the models will mostly stay in cash and only trade the few opportunities where the return is predicted to be in the top percentile. This approach is consistent with studies of historical returns on the S&P500 and other market indices showing that the best 10 days in any given year are responsible for generating on average 50% of the total market return for that year. Furthermore, the best 50 days in any given

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

year are responsible for about 93% of the total return for the whole year [4] thus the idea of focusing on identifying the most profitable trading opportunity and avoiding taking unnecessary risk by acting on every possible trade signal. The threshold for identifying positive trades is a hyperparameter that greatly impacts the number of trades executed during the test period (which in turns affects the trading costs), the total return and the maximum draw-down. Due to resource and time limits, this hyperparameter will not be changed in this study and is left constant at top 10 percentile.

2 Related work

Stock market prediction is usually considered as one of the most challenging issues among time series predictions [5] due to the noise and high volatility associated with the data. During the past decades, machine learning models, such as Artificial Neural Networks (ANNs) [6] and Support Vector Machines (SVR) [7], have been widely used to predict financial time series with remarkable accuracy. More recently, deep learning models have been applied to this problem due to their ability to model complex nonlinear topology. An improvement over traditional machine learning models, deep learning can successfully model complex real-world data by extracting robust features that capture the relevant information [8] and as a result achieve better performance [9].

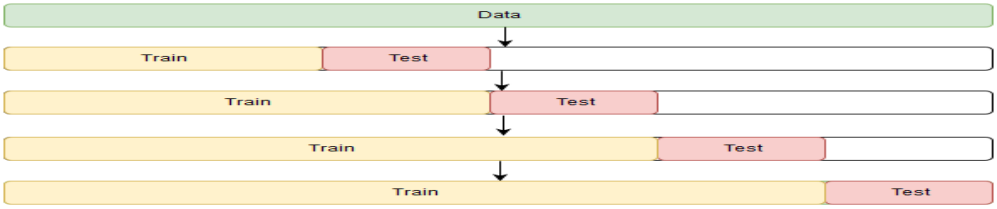
Many examples for the successful use of deep learning methods in developing algorithmic trading models are available and can generally be split into two categories: Deep learning based methods and reinforcement learning based methods. For instance, Arevalo et al. [10] introduced a high frequency trading strategy based on a Deep NN that achieved a 66% directional prediction and 81% successful trades over the test period. Bao et al. [11] used wavelet transforms to remove the noise from stock price series before feeding them to a stack of autoencoders and a long short-term memory (LSTM) NN layer to make one-day price predictions. Furthermore, M et al. [12] compared CNN to RNN for the prediction of stock prices of companies in the IT and pharmaceutical sectors. In their test, the Convolutional Neural Network showed better results than the Recurrent Neural Network and Long-Short Term Memory. The difference in performance was attributed to the fact that CNN does not rely on historical data as is the case with time sequence based models. On the other hand, Sutskever et al. [13] argues for the use of LSTM and sequence-to-sequence models for their ability to retain information from earlier examples in the training set while adapting to newly arriving data. Alternatively, many researchers focused on using Reinforcement Learning techniques for addressing the algorithmic trading problem. For instance, Moody and Saell [14] introduced a recurrent reinforcement learning algorithm for identifying profitable investment policies without the need to build forecasting models, and Dempster and Leemans [15] used adaptive Reinforcement Learning to trade in foreign exchange markets. Reinforcement Learning models present two advantages over Deep Learning predictive models. First, RL does not need a large labeled training data set, This is a significant advantage as more and more data becomes available it becomes very time consuming to label the data set. Furthermore, RL models use a reward function to maximize future rewards (reward functions can be formulated according to any optimization objective of interest such as maximum return or minimum risk), in contrast to DL regression and classification models which focus on predicting the probability of future outcomes. We believe that a combination of both methods in a Deep Reinforcement Learning approach presents the best of both worlds as it allows the agents to learn deep features from the training data while avoiding the need for a labeled data set and allowing for the customization of specific reward functions.

3 Dataset and Features

This study is based on a financial dataset extracted from the Jane Street Market Prediction competition on Kaggle [16]. The available dataset is composed of 2,390,491 record each defined using 130 anonymous features measured sequentially spanning 500 days at different time steps during each day. The number of transactions varies from day to day with the minimum being 29 transactions on day 294 and the maximum of 18884 transactions on day 44. The data does not specify an explicit target but provides five columns that represent the realized percent return on each trade and the returns over 4 different time horizons. The objective is to populate an action column with one of two decisions: to trade or not to trade. Note that the exact nature of the trade is unknown (long or short) as well as the specific instrument or market traded, in other words, only the return values are provided for the output. For this study, return values in the top ten percentile of all returns will be marked with

a positive trade signal while every other trade will be marked with a negative signal. Furthermore, by analyzing the missing values from each feature, it is clear that they follow a fixed time pattern regardless of the number of transactions on any given day which could be valuable information to the network. As a result, we will augment the features matrix with a logical matrix of size [m,130] where m is the number of training examples. Each element of the logical matrix at [i,j] will be set to true if the features matrix has a missing value at the corresponding [i,j] location. Following the creation of the logical matrix, the last 50,000 records of the available data are set aside for testing.

Due to the sequential nature of the dataset, random validation and testing sets are not appropriate and instead we will use a rolling cross-validation approach. We start training with the first 1,000,000 transactions and validate on the next 250,000 records. Next, the first validation set is included in the second training set resulting in a second training set of 1,250,000 records and we use the following 250,000 records for the second validation.set and so on until we reach a training set that includes the first 2,000,000 records and is validated on the following 250,000 records. The rolling cross-validation process is show in schematically in Figure (1) below [20].



Prepossessing of the training and development data is performed over two steps. First, a SimpleImputer from the SKLearn library [17] is used to replace the missing values with the median of each feature over the training set. Next, a RobustScaler from the SKLearn library [18] is used to normalize the data. This scalar removes the median and scales the data according to the inter-quantile range of each feature. The two pre-processors are saved in separate files for use with the test subset.

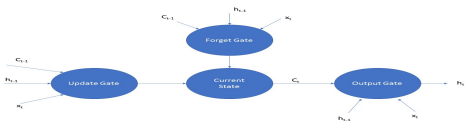
4 Methods

Two types of models are tested for this project. Three LSTM and two CNN models differing in architecture and/or number of hidden layers are considered. Using the rolling validation procedure described previously the best model from each family is identified and used for final out-of-sample testing.

1 - CNN Models: A convolutional neural network is a type of deep neural networks that is effective in forecasting in time series applications. In our case we use a 1-dimensional CNN to extract features from the input tensor. A Max Pool 1D with a pool size of 2 is applied to each CNN layer. The output from the last convolutional layer is flattened and passed to one or more dense layers before applying a sigmoid activation to classify the trade. During training we apply label smoothing of 0.2 to the Binary Crossentropy loss function to effectively lower the loss target from 1 to 0.8 to lessen the penalty for incorrect predictions, we believe this is necessary given the volatile and unpredictable nature of future stock market predictions using the model. Two architectures are considered as shown in Figure (2) in the appendix, the main difference is the size of the network by adding additional 1D CNN layers with increasing filter sizes as well as adjusting the number of dense layers.

2 - LSTM Models: LSTM is a deep neural network architecture that falls under the family of recurrent neural networks (RNN). RNNs are deep networks that have feedback loops. Traditional RNNs suffer from what is known as the problem of vanishing and exploding gradient in which the network either stops learning (vanishing gradient) or never converges to the point of minimum cost (exploding gradient). LSTM are designed to eliminate both problems and hence have become popular in modelling complex sequential data. LSTM layers consist of cells that store historical state information as well as gates that control the flow of information through these cells. LSTM cells have three types of gates: forget gate, update gate, and output gate. The forget gate outputs a number between 0 and 1, where during the learning process a "1" means "completely keep this information" while a "0" is translated to "completely ignore this information". The update gate chooses which new data will be stored in the cell. First, a sigmoid layer chooses which values will be changed and then a tanh layer creates a vector of new candidate values that could be added to the state. Finally the output

gate decides what will be the output of the LSTM cell which will be a combination of the cell state and the newly arriving data. The LSTM cell structure is shown in the figure.



In the figure, h_{t-1} represents the output from the previous neuron, x_t is the input to the current neuron, $C_t - 1$ is the neuron state at the previous time step. The LSTM model architecture used is shown in Figure (3) in the appendix. The first LSTM layer has hidden units varying from 64 to 128 to 256. The LSTM layer is followed by a dropout layer with a keep probability of 75%. Followed by a second LSTM layer with hidden units varying from 32 to 64 to 128. Followed by a second dropout layer with a keep probability of 75%. Finally a softmax layer is used to output the trade decision between 0 (no trade) or 1 (trade).

5 Experiments/Results/Discussion

For this study the objective is to train the networks to minimize the mean squared error over the training set. Adam optimization is used for both LSTM and CNN models. The Adam optimization algorithm is an extension of stochastic gradient descent and has shown significant advantages in minimizing non-convex functions. A learning rate of 0.001 was selected after some initial experimentation with reduced training sets as well as a batch size of 32. For the LSTM, five different sequence lengths were tested (15, 20, 25, 30, 35, 40) each representing a trade-off between using longer lags to determine the trade decision with the risk of including too much irrelevant information in a highly dynamic environment. Based on initial tests, it was determined that a sequence length of 10 provided the best results over the validation set.

To compare the candidate models we will use precision, recall and F1 scores for each model as well as the Sharpe Ratio, Total Return and Maximum Draw-down over the test period. Typically, classification accuracy is defined as the total number of correct predictions divided by the total number of predictions made for a dataset. However in this case, accuracy is an inappropriate measure because the problem is highly imbalanced by design. Recall that only the top 10 percentile of all training records are marked with "1" thus the overwhelming majority of the training set is from the negative class meaning that even a poor model can achieve high accuracy scores by simply choosing to not trade at all.

For the 1D CNN model, we tested the model with and without Batch Normalization and found that it improved results particularly when training with a lower number of epochs. The Dropout layer after each convolution was tested with a rate range between 0.1 and 0.5, we found that the additional regularization gained from the higher dropout rate produced the best result. Both Average Pool and Max Pool were tested, the difference in performance between the two was negligible. Decreasing the batch size from 256 in earlier models to 32 was particularly effective.

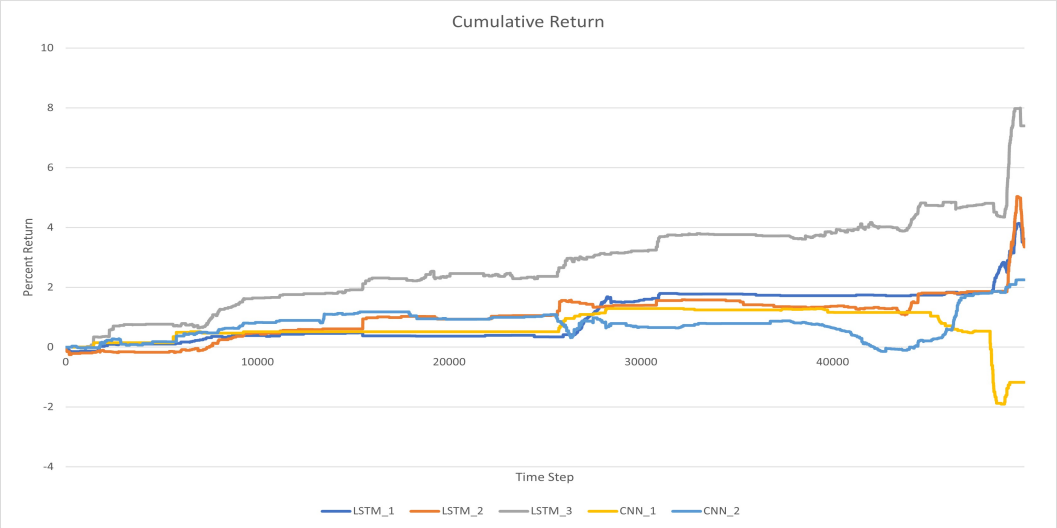
The results for the best model after four rolling-validation runs are given in Table (1) below.

Model	Dev. Precision	Dev. Recall	Dev. F1	Test Precision	Test Recall	Test F1
LSTM64x32	0.40	0.01	0.02	0.33	0.02	0.04
LSTM128x64	0.37	0.02	0.03	0.36	0.02	0.04
LSTM256x128	0.36	0.03	0.06	0.39	0.04	0.07
CNN1	0.41	0.01	0.01	0.46	0.01	0.02
CNN2	0.28	0.03	0.05	0.41	0.04	0.08

Table 1: ML Metrics for Last Validation and Test runs

The precision metrics (percentage of positive identification that was actually positive) does not vary significantly for the three LSTM models but has a significant drop from the first to the second CNN model. Given that the positive class is defined as trades in the top 10 percentile, many of the misclassified positives will still be winning trades even if not among the best trades originally targeted. This will be clear from the risk return metrics which will show that even with a low precision, the models are still profitable. The recall (percentage of true positive actually classified as positive)

metric shows that all 5 models are only able to capture a very small percentage of the best trades which leaves a lot of room for improvement. It is however noticeable that the biggest LSTM model as well as the second CNN model achieve the highest precision, which indicates that the models suffer from high bias specially that the recall over the validation and test sets are very close. The F1-score is a combination of precision and recall and shows see that the two largest models (most trainable parameters) achieve the highest scores indicating that future work should try even deeper models.



Figure(3): Cumulative Return over Test Period

The cumulative return over the test period are shown in Figure (4) below. The LSTM256x128 model generates the highest cumulative return 0f 7.4%. The financial performance metrics for the LSTM models are reported in Table (2).

Model	Total Return	Max Draw-down	Number of Trades	Avg. Ret. per Trade	Sharpe Ratio
LSTM64x32	3.61%	0.63%	351	1.03%	0.022
LSTM128x64	3.34%	1.69%	309	1.08%	0.018
LSTM256x128	7.39%	0.59%	491	1.50%	0.032
CNN-1	-1.18%	N/A	107	-1.1%	-0.007
CNN-2	2.25%	N/A	416	0.54%	0.014

Table 2: Financial Performance of LSTM and CNN Models

As expected, all three models took very few trades from the possible 50,000 opportunities available. However, the performance in terms of average return per trade taken is excellent as well as the very low draw-down of this strategy. The best model is the LSTM256x128 across the board with almost double the total return as any other model and with the lowest risk. It is also noticeable how the average return per trade is about 50% higher with the best model despite taking 100 more trades, which reflects the improvement in both precision and recall of the model as the number of parameters is increased. Finally, one possible explanation for the good performance of the models despite the very low recall values is that the models are learning to identify the best trades, but when they fail, they do not fall from them, still identifying good trading opportunities even if not the best.

6 Conclusion/Future Work

A novel approach for training deep neural network for automated training was presented. Rather than attempt to predict the exact return at every future time step, the problem is formulated as a binary classification one with the goal of identifying the most promising trading opportunities. Furthermore, the feature matrix was augmented by adding a logical array to preserve the information about missing features at each time step. Result show positive returns with very low risk as a result of only targeting

the safest trading opportunities. If more time and resources are available, deeper networks would be tested as well as different threshold for the positive class (this study considered only one such threshold at top 10 percentile). Combining Reinforcement Learning with the LSTM model could also be investigated with the reward function based on the identification of major opportunities only.

7 Contributions

Mohamed Elseifi wrote the prepossessing function, the post-processing (testing and results analysis) function, the LSTM models, the final paper and the presentation slides. Hamdy Hamoudi wrote the CNN model code, the two sections in the final paper related to CNN.

References

[1] Stelios D. Bekiros (2010) Fuzzy Adaptive Decision Making for Boundedly Rational Traders in Speculative Stock Markets *European Journal of Operational Research* **202**(1) :285-293.

[2] Zhang, Y., Yang, X. (2016) Online Portfolio Selection Strategy based on Combining Experts Advice *Computational Economics* **50**(5).

[3] Cavalcante, R.C., Brasileiro, R.C., Souza, V.F., Nobrega, J.P. and Oliveira, A. (2016) Computational Intelligence and Financial Markets: A Survey and Future Directions *Expert Systems with Applications* **55**:194-211

[4] Wang, L., Hajric, V. (2020) The Cost of Bad Market Timing Decisions in 2020 was Annahilation *Bloomberg*.

[5] Wang B, Huang H, Wang X. (2012) A novel text mining approach to financial time series forecasting *Neurocomputing* **83**(6): 136-145.

[6] Guo Z, Wang H, Liu Q, Yang J. (2014) A Feature Fusion Based Forecasting Model for Financial Time Series *Plos One* **9**(6): 172-200.

[7] Prasaddas S, Padhy S. (2012) Support Vector Machines for Prediction of Futures Prices in Indian Stock Market *International Journal of Computer Applications* **41**(3): 22-26.

[8] Hinton GE, Salakhutdinov RR (2006) Reducing the Dimensionality of Data with Neural Networks *Science* **313**(5786): 504-507.

[9] Bengio Y, Courville A, Vincent P. (2013) Representation Learning: A Review and New Perspectives *IEEE Transactions on Pattern Analysis Machine Intelligence* **35**(8): 1798-1828.

[10] Arevalo, A., Nino, J., Hernandez, G. and Sandoval., J. (2016) High-Frequency Trading Strategy Based on Deep Neural Networks *ICIC*.

[11] Bao, W.N., Yue, J. and Rao, Y. (2017) A Deep Learning Framework for Financial Time Series using Stacked Autoencoders and Long-Short Term Memory *Plos one* **12**.

[12] M, H., Gopalakrishnan, E.A., Menon, V. and Kp, S. (2018) NSE Stock MArket Prediction Using Deep-Learning Models *Procedia Computer Science* **132**(10): 1351-1362.

[13] Sutskever, I., Vinyals, O. and Le, Q. V., (2014) Sequence to Sequence Learning with Neural Networks *Advances in neural information processing systems*: 3104-3112.

[14] Moody, J.E. and Saffell, M. (2001) Learning to Trade via Direct Reinforcement *IEEE Transactions on Neural Networks* **12**(4): 875-889.

[15] Dempster, M.A. and Leemans, V. (2006) An Automated FX Trading System using Adaptive Reinforcement Learning *Expert Systems Applications* **30**(5): 543-552.

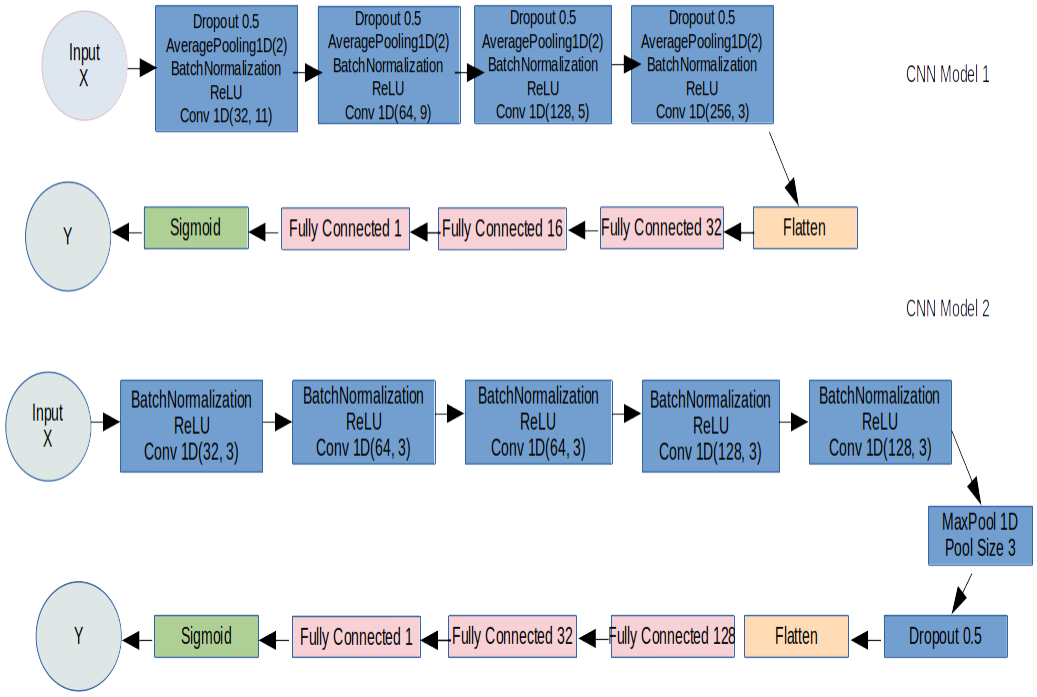
[16] Kaggle, *Jane Street Market Prediction*, "https://www.kaggle.com/c/jane-street-market-prediction"

[17] SciKit Learn, *Imputation of Missing Values*, https://scikit-learn.org/stable/modules/impute.html

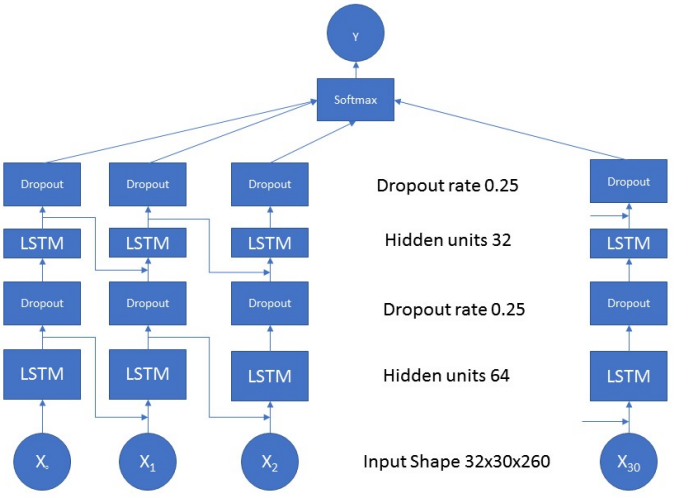
[18] SciKit Learn, *Preprocessing*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

[20] Stack Exchange, *Cross Validated*, "https://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection"

Appendix



Figure(1): CNN Model Architectures



Figure(2): LSTM Network Architecture