Deep Neural Networks in High Frequency Trading

Prakhar Ganesh, Senior, Dept. of CSE, IIT Delhi, and Puneet Rakheja, Founder and CEO, WealthNet Advisors

Abstract—The ability to give precise and fast prediction for the price movement of stocks is the key to profitability in High Frequency Trading. The main objective of this paper is to propose a novel way of modeling the high frequency trading problem using Deep Neural Networks at its heart and to argue why Deep Learning methods can have a lot of potential in the field of High Frequency Trading. The paper goes on to analyze the model's performance based on it's prediction accuracy as well as prediction speed across full-day trading simulations.

Index Terms—Deep Learning, Neural Networks, Multi Layer Perceptrons, Finance, High Frequency Trading

I. Introduction

DEEP Learning methods are prophesied to revolutionize the field of AI and represents a step towards building autonomous systems. We live in an era where we are creating unbelievable amount of data everyday. Neural networks hold the ability to scale problems that were previously unsolvable, causing a huge wave of interest in this field. For example, currently, deep reinforcement learning is used in problems such as learning to play games directly from the pixels of an image, which would not have been considered a scalable problem up until recent years [10], [1].

Deep learning has become increasingly popular [20] since the introduction of an effective new way of learning deep neural networks [21], [22]. It has proved very effective for large-scale tasks, and this success has been based largely on the use of the back-propagation algorithm with rather standard, feed-forward multi-layer neural networks. In addition to improved learning procedures, the main factors that have contributed to the recent successes of deep neural networks have been the availability of more computing power, the availability of more training data, and better software engineering.

The inference of predictive models from historical data is not new in quantitative finance; a number of examples include coefficient estimation for the CAPM, Fama and French factors [5], and related approaches. However the special challenges for machine learning presented by HFT can be considered two fold:

- Microsecond sensitive live trading As the complexity
 of the model increases, it gets more and more computationally expensive to keep up with the speed of live
 trading and actually use the information provided by the
 model.
- Tremendous amount and fine granularity of data The past data available in HFT is huge in size and extremely precise. However there is a lack of understanding of how

Copyright 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

such low-level data, like the recent trading behavior, relates to actionable circumstances (such as profitable buying or selling of shares) [6]. There has been a lot of research on what "features" to use for prediction or modeling [6], [7], [8], [9], yet it is difficult to model the problem without a proper understanding of the underlying relations.

II. RELATED WORK

One of the primary goals of the field of artificial intelligence (AI) is to produce fully autonomous agents that interact with their environments to learn optimal behaviours, possibly improving over time through trial and error.

Currently, deep learning is enabling many other machine learning algorithms, for example reinforcement learning as mentioned earlier, to scale to problems that were previously intractable, such as learning to play video games directly from pixels. Deep Learning has penetrated a lot of fields, including finance. However its reach in high frequency trading is limited [19], specifically due to the computational constraints and primitive problem modeling methods.

There has been a lot of other machine learning algorithm tried and tested in the field of high frequency trading. There has been a lot of work done specifically in terms of feature engineering in HFT, focused on simpler models like linear regression, multiple Kernel learning, maximum margin, traditional model-based reinforcement learning etc. [6], [13].

However, due to the computational complexity of Deep Learning models, lesser work has been done in terms of incorporating such recent and more complex models and instead more focus is made towards extracting useful features from the current trading book state and recent trading behavior. The common feature values like bid-ask spread, percentage change in price, weighted price etc. [6] and some specialized features like order imbalance [18] were among many others that we used in our model.

III. PROBLEM STATEMENT

We aim to create a pipeline which uses information about the past trading behavior and current snapshot of the order book to predict price movement in the near future. We then aim to use this information for making decision in the market for maximum profitability.

A. Tick Data

Tick data refers to most granular level market information available from electronically traded markets. Every order request and trade information is provided as a "tick" event. Current state of the order book refers to the top few (five, in our case) bid orders and ask orders placed along with their proposed prices and volumes. [17]

Tick data is raw, uncompressed data of the trading behavior of the market and requires a lot of storage space along with a minimum standard hardware requirements for capturing the data. Tick data is essential for a lot of different types of analysis of the trends present in the market.

B. Bid - Ask Spread

Bid and Ask are the prices that buyers and sellers respectively are willing to transact at, the bid for the buying side, and the ask for the selling side. A transaction takes place when either a potential buyer is willing to pay the asking price, or a potential seller is willing to accept the bid price. The spread between the top bid and the ask price at any moment is called the bid-ask spread. The mid-price is the mean price of the top bid and the top ask price. [14]

Most of the standard trading algorithms work on the principle of mid-price prediction or mid-price movement prediction. However a big drawback of this technique can be seen from Fig 1. Clearly in case 1, there is an increase in the mid-price of the product, but since we need to enter the market at the ask price and exit at the bid price, we are actually incurring a loss in this transaction. In case 2, the movement of the price is large enough to cover the bid-ask spread and thus is a profitable transaction.

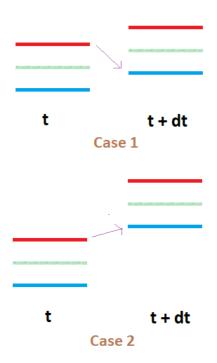


Fig. 1. Two possible cases of increase in mid-price analyzed. Red, blue and green represents the ask, bid and mid price of the product respectively.

We have however diverted from this traditional approach and instead focused on predicting only those price movements which are substantial enough to cross the bid-ask spread. It is important to understand that since we only consider change in price that cross the bid-ask spread as a "movement", which would certainly generate profit, predicting "no movement" does not mean that there was absolutely no movement in the price of the product. It just means that the movement was not substantial enough to cross the spread and thus was as useless for our model as no actual movement in the price.

C. Mean Reversion

Mean reversion is a financial theory which suggests that the price of a stock tends to return towards its long running mean/average price over time [16]. Trading on this strategy is done by noticing companies whose stock values have significantly moved away in some direction from its long running mean and thus is now expected to move in the opposite direction. Ignoring the erratic trading periods at the start and end of the day, this behavior in stock prices is evident in most of the stock markets across the world [15]. The oscillating behavior of the price around some changing mean across the day as seen in Fig 2 is an example of Mean Reversion.

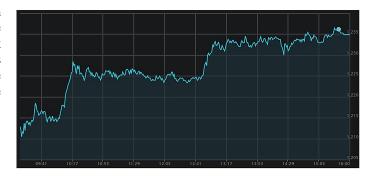


Fig. 2. A snapshot of a full-day behavior of an anonymous product. Captured from http://terminal.moneycontrol.com/

Using mean reversion in stock price analysis involves both identifying the trading range for a stock and computing the mean around which the prices will be oscillating. There exists a few good statistical algorithms for the mean prediction, however the movement of the mean value in itself is something that comes into evidence a little too late for such statistical methods which can cause wrong predictions. Allowing a deep learning model to understand this oscillating nature of price movement and letting it learn the mean values and the trading range can substantially boost the prediction accuracy.

IV. PROPOSED MODEL

The proposed pipeline contains a deep learning model predicting the stock price movement followed by a financial model which places orders in the market based on the predicted movement. The training of the neural network can be broadly divided into two parts, the first part is thorough training using the data from the previous day and the second part is online training in mini batches, while receiving the live data. Now let us discuss the prediction model in detail.

A. Deep Neural Model

The problem statement is modeled as a classification problem between three classes,

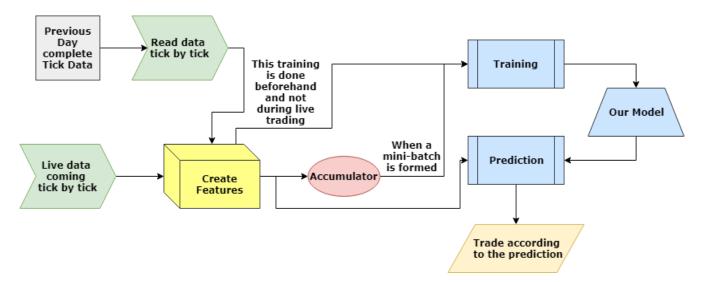


Fig. 3. Complete pipeline

- 1) Increase in price. Will cross bid-ask spread.
- The change in price, if any, won't cross the bid-ask spread.
- 3) Decrease in price. Will cross the bid-ask spread.

The model used is a standard MLP model. However, instead of using a single MLP model, we switch to an ensemble of 3 models. We use one-vs-one classification techniques using these 3 models to get final confidence scored for each class.

Activation functions used are ReLU at every hidden layer and the softmax function after the last layer while the error function used is Categorical Cross entropy. Predictions in which the confidence score of the winning class is less than some predefined confidence threshold are considered not good enough to act upon. These are tagged as "no confidence" predictions and the behavior of our trading model in these case is same as its behavior when the predicted class is "no movement".

Since every security in the market has a different frequency of activity, both history for feature creation and range of future prediction is discussed in terms of number of ticks instead of absolute time. It can be interpreted as ignoring the concept of actual time and considering every tick in the life of a security as one second.

B. Online Training

The market dynamics keep on changing with time and using just the knowledge of the market behavior on previous days is not a smart choice, no matter how good our model is. So we need to also keep updating the model online while autonomously trading side by side. This is where we incorporate online training and use back-propogation at regular intervals to keep updating the model weights. As mentioned earlier, processing speed of the model is a very important factor in the trading domain and thus updating the model after every tick is not a feasible solution, so instead we create minibatches of data for training.

We accumulate ticks to create these batches and after some specified number of ticks, we update our model using this mini batch. Since the model weights are already initialized by the weights learned on the previous day data, we hypothesize that the model understands the concept of mean inversion but just needs to fine tune itself according to the current time of the day and market conditions and thus lesser number of iterations during the online training are required.

C. Complete Pipeline

The complete pipeline can be broadly divided into three sections:

- Pre-Training The model needs to be trained beforehand using the data available from the market activity of previous days. The trained weights are saved and used to initialize the model before taking it live.
- 2) Prediction Features are created on the run and the model predicts price movement of the stocks using these features at every tick. Bid and Ask orders are placed in or withdrawn from the market using this information.
- 3) Online Training Running parallel to the predictions, we need to accumulate the feature values and corresponding ground truth values which we will get in the near future. Once a mini-batch is formed, the weights are updated by tuning the model using this batch and the same process of accumulating data starts again.

Refer to Fig 3 for a a better understanding of the flow of the pipeline.

V. DATASET

The dataset consists of tick by tick (tbt) data of over 1200 securities for a whole week. Every tick contains the price and volume information of the last transaction and the top 5 levels of the order book of both the bid and ask side.

The market behavior is not mean reverting in the first few minutes after the market opens and in the last few minutes before it closes. So in order to capture the mean reverting trend in the market, we removed the first 30 and the last 30 minutes of data from every day to create a cleaner dataset. Also, we

categorized securities based on their frequency of activity in the market everyday and only worked with securities which had frequency between 50,000 to 100,000 ticks on an average everyday.

The final dataset that we used for training and simulation contains around 200 filtered securities, with an average activity of around 70,000 ticks everyday. Thus we had approximately 70,000 * 200 = 14 million data points for every single day and a whole week's worth of similar data to work with.

VI. MODEL ANALYSIS

The traditional comparison metrics like accuracy, F-score etc. across all possible output classes are not appropriate in this case due to the factors like risk involved, opportunities to earn profit and the speed of prediction.

A possible example of two different extremes can be noticed by varying the confidence bound of the model. For example, a model with low confidence bound will have negligible "no confidence" predictions and thus will provide a lot of opportunities for the user to gain profit but will also fail an awful lot number of times. On the other hand, a model with high confidence bound will have almost all of its predictions as "no confidence" prediction and thus will give prediction of price movements only in the most obvious of cases and thus will provide with close to zero opportunities for the user to enter or exit the market, even though the model will be extremely accurate.

Thus a combination of few different measures and terms are defined and used in correlation with each other to evaluate model performance and for parameter tuning. Every analysis done from this point forward is done on an unseen full day trading simulations for known securities. The terms defined below will be directly used in the analysis that follows:

- Actionable Predictions Predicting increase or decrease in the price (which will cross the bid-ask spread) is an actionable prediction since it initiates an action of either entering or exiting the market. Predicting no movement in price or a no confidence prediction are not considered actionable predictions.
- 2) Accuracy Percentage of correct actionable predictions out of all actionable predictions.
- 3) Confidence Bound The threshold bound of the confidence score above which a prediction is considered valid. If the confidence score of the winning output class is below the confidence bound, the final prediction is considered as a no confidence prediction. The trading model behavior in this case is same as the behavior in case of no movement prediction.
- Participation Percentage Percentage of actionable predictions out of all the predictions.

A. Online Training

To understand the impact and importance of online learning in our model, we ran two different models, one with regular weight updates and another one with constant weights throughout the simulation. The two models were compared across a variety of different parameter values and every time the online learning model seemed to do exceptionally better than a simple neural model as can be seen in Fig 4.

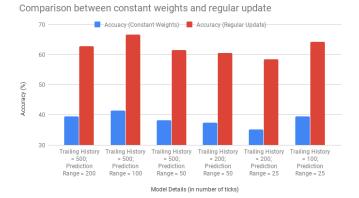


Fig. 4. Comparison between constant weights and regular update

B. Trailing History and Prediction Range

As previously mentioned, the unit of time used in our pipeline will be in terms of the number of ticks instead of absolute time. The two important parameters of our model are the trailing window of time which is used to create features for the current tick and the range of prediction representing how much into the future are we predicting.

75 500

70 100 500

65 500 500

60 888 200

45 45 40 200

30 25 50 100 200

Variation of Accuracy with Trailing history and Prediction Range

Fig. 5. Variation of Accuracy with Trailing history and Prediction Range (Data labels represent Trailing history).

Both of these values need to be represented in terms of fixed number of ticks. We experimented with different combinations of the two and compared their performance. To do a fair assessment of the models, their participation percentage was made constant at approx 10% by varying the confidence bound and then their accuracy was compared as shown in Fig 5.

Using a history of 500 ticks for feature creation and then predicting 100 ticks into the future seems to be outperforming any other combination. Since our hypothesis is based on the model understanding the trend of mean reversion in the market, clearly these optimal parameter values can vary from one market to the other. For our analysis we will use this combination in the all the following comparisons.

C. Mini Batch

The size of mini-batch is also a parameter of concern. In case we take mini-batches of larger size, there is a longer period of time for which the model runs on constant weights and thus performance decreases a little. However with larger batch sizes, we have a benefit of speed since it requires update after a larger interval and lesser computational power can be spent on it. Also with larger batches, there is a smaller chances of the noise present in the data affecting our training adversely. The opposite happens in case of smaller batch sizes and thus we need to create a balance for the best prediction accuracy with good prediction speed.

With the 500 tick history, 100 tick prediction range and a constant participation percentage of around 10%, the performance of the model for different mini-batch sizes was compared in Fig 6. It is clear that decreasing the batch size any further than 100 does not provide us with any substantial increase in accuracy. Another thing we need to keep in mind is that in order to keep up with the live predictions, we would need twice the amount of computation power in case of batch size of 50 than of 100. Thus it might not be worth it to shift to a batch size of 50 for the minimalistic improvement in accuracy and will depend on one's preferences and availability of computation power.



200

100

Mini-Batch Size

50

Fig. 6. Variation of Accuracy with Mini-Batch Size

D. Multi-Layer Perceptron

500

45

Once we have finalized the input-output parameters, we need to tune our MLP to incorporate the fact that we cannot afford a very complex model due to the time restraint and prediction speed requirement and at the same time a simple regression model might not be able to provide us with desirable predictive power. Thus some form of non-linearity needs to be introduced in the model along with making sure that the complexity remains at a minimum.

Different model architectures were experimented with. Every hidden layer had a ReLU layer at the end and the final layer had softmax for all the models compared. The final results can be seen from Fig 7. Clearly the increase in accuracy beyond a (10, 10) model is not significant and as mentioned earlier, the choice of model depends on the availability of computational power.

Variation of Accuracy with different MLP Architectures

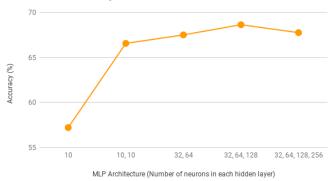


Fig. 7. Variation of Accuracy with different architectures

E. Confidence Bound

Variation of Accuracy with the Participation Percentage

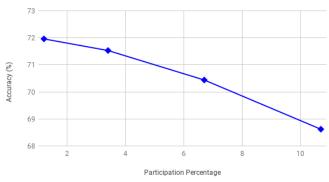


Fig. 8. Variation of Accuracy and Participation percentage

One important aspect of our model is our ability to switch between giving more importance to participation percentage of the model or accuracy of its predictions. The analysis of the same is done in Fig 8 and one can always adjust the bar between the two according to the person's willingness towards the risks involved. However just to put things in perspective, since there are 14 million ticks happening on an average everyday, one percentage change in participation percentage amounts to a change of 140,000 chances of making a trade everyday in absolute terms.

F. Prediction Speed Analysis

The above analysis was solely based on the profitability and accuracy of the model. However an important aspect of the proposed pipeline is availability of enough computational power to keep updating the model weights of every product in parallel while making predictions in real time.

To give a perspective to the reader, we coded our real time prediction model in C++ along with a working trading module and our code for updating the model weight in python and both of these were running in parallel, communicating in real time. The server on which we ran the two programs was Intel 4

core, 3GHz processor with 16 GB RAM. The two programs were running in parallel and in pace with each other when we were trading live across 20 different securities using a 500 tick trailing history, 100 ticks prediction range, with a mini-batch of size 100 and (10, 10) MLP model.

VII. CHALLENGES AND FUTURE WORK

Most of the Machine Learning models in this field are based on primitive modeling techniques. Strategies like using ticks as a measurement of time instead of the absolute time itself, considering only those price movements which were able to cross the bid-ask spread, introduction of a confidence bound to ensure the difference between actionable and no confidence predictions etc. were different from these conventional approaches. It is possible that a huge reason of our model's success were these hyperparameters. Thus experimenting with other existing successful models in this field, using the problem modeling techniques we propose, could also significantly boost their performance.

Since it is clear that online learning was an important aspect of the model described in this paper, we can assume that providing this extension to some other existing pipelines can also provide better prediction accuracy.

While our focus was only on the prediction part of the pipeline, there are also a lot of financial models which can use the information provided by our model and create a better trading strategy for more profitability. One can also experiment to see how well can our model integrate with these techniques to create the most profitable autonomous trading system.

ACKNOWLEDGMENT

We would like to thank WealthNet Advisors for their continuous guidance and access to computer hardware and a working trading module for all the experimentation and analysis done in this paper.

REFERENCES

- Kai Arulkumaran and Marc Peter Deisenroth and Miles Brundage and Anil Anthony Bharath, A Brief Survey of Deep Reinforcement Learning, 2017
- [2] Volodymyr Mnih and Koray Kavukcuoglu and David Silver and Alex Graves and Ioannis Antonoglou and Daan Wierstra and Martin Riedmiller, Playing Atari with Deep Reinforcement Learning, 2013
- [3] Deep Reinforcement Learning Doesn't Work Yet, Irpan, Alex
- [4] The Dark Secret at the Heart of AI, https://www.technologyreview.com/s/604087/the-dark-secret-at-theheart-of-ai/
- [5] Common risk factors in the returns on stocks and bonds, Fama, Eugene and French, Kenneth, 1993, Journal of Financial Economics
- [6] Michael Kearns and Yuriy Nevmyvaka, Machine Learning for Market Microstructure and High Frequency Trading
- [7] Dat Thanh Tran and Martin Magris and Juho Kanniainen and Moncef Gabbouj and Alexandros Iosifidis, Tensor Representation in High-Frequency Financial Data for Price Change Prediction, 2017
- [8] LEE, CHARLES M. C. and READY, MARK J., Inferring Trade Direction from Intraday Data, The Journal of Finance
- [9] Rama Cont and Arseniy Kukanov and Sasha Stoikov, The Price Impact of Order Book Events, 2010
- [10] Volodymyr Mnih and Koray Kavukcuoglu and David Silver and Alex Graves and Ioannis Antonoglou and Daan Wierstra and Martin Riedmiller, Playing Atari with Deep Reinforcement Learning, 2013
- [11] Tuomas Haarnoja and Vitchyr Pong and Aurick Zhou and Murtaza Dalal and Pieter Abbeel and Sergey Levine, Composable Deep Reinforcement Learning for Robotic Manipulation, 2018

- [12] Shixiang Gu and Ethan Holly and Timothy Lillicrap and Sergey Levine, Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates, 2016
- [13] Adamantios Ntakaris and Martin Magris and Juho Kanniainen and Moncef Gabbouj and Alexandros Iosifidis, Benchmark Dataset for Mid-Price Prediction of Limit Order Book data, 2017
- [14] Investopedia, Bid and Asked, https://www.investopedia.com/terms/b/bidand-ask.asp
- [15] Market making and mean reversion, Tanmoy Chakraborty and Michael Kearns, 2011
- [16] Investopedia, Mean Reversion, https://www.investopedia.com/terms/m/meanreversion.asp
- [17] Trade Ideas, Tick Data, https://www.trade-ideas.com/glossary/tick-data/
- [18] Darryl Shen, Order Imbalance Based Strategy in High Frequency Trading, 2015
- [19] Arevalo, Andres and Nino, Jaime and Hernandez, German and Sandoval, Javier, High-Frequency Trading Strategy Based on Deep Neural Networks, 2016Intelligent Computing Methodologies
- [20] J. Markoff. Scientists See Promise in Deep-Learning Programs, New York Times, Nov 24, 2012
- [21] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. Neural Computation, Vol. 18, 1527-1554, 2006
- [22] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 - 507, July 2006