DOI: 10.5203/pmuser.201841604

Research Article

Analysis and Prediction of Patterns in Futures Trading Datasets Using LSTM

Beom-Jin Park[1], Christopher Chmelyk[1], Daniel Heslop[1], Gu Zhengyu[1]

[1]Dept. of Computer Science, University of Manitoba, Winnipeg, MB, R3T 2N2

Corresponding Author: B. Park (parkb346@myumanitoba.ca)

Abstract

One of the most promising tools in recent years for the analysis and prediction of time series data, which includes financial market data has been the use of neural networks. While the complexities inherent in market prediction have confounded machine learning techniques, newer deep learning techniques such as long short-term memory (LSTM) show promise in their ability to predict using time series datasets. This paper will explore the feasibility of using a recurrent neural network (RNN) with LSTM as a predictive tool for use with futures trading data, and determine the time ahead with which this particular tool can maintain its predictive accuracy. Using a dataset comprised of all futures trading occurring on the Bourse de Montréal (TMX) during a nine-month period from January to September 2015, we assessed the predictive effectiveness of an RNN in predicting the price of front-end contracts for the futures symbol BAX. We found that while an RNN provided a degree of short-term predictive capability, this capability did not extend beyond a couple of days. Although it failed as a trading instrument to predict futures prices, the RNN could detect, identify, and reflect underlying trends in the data, indicating the tool may hold promise in the detection of trading patterns.

Keywords: Machine Learning, Deep Learning, Time Series Dataset, Prediction, Futures Trading, Market Prediction

1 Introduction

For the obvious reason of profit, mankind's interest in market prediction has existed for as long as markets themselves. While many predictive tools and methods have been employed over the years, relatively recent advances in computing power have enabled new strategies to be attempted. Prediction is one of the most extensively applied tasks in time series data mining[1]. This results from the fact that prediction is a logical extension of data mining processes. There have been many methods used for the purposes of prediction, such as reinforcement learning with Q-learning, Bayesian networks, and recurrent neural networks.

Since their inception, neural networks have been considered one of the best solutions for prediction in financial markets[2]. Because of the complex and chaotic nature of financial market data, traditional statistical methods of prediction and analysis can be of limited utility. This is where machine learning and neural networks may prove useful[3]. By training recurrent neural networks to map input sequences to output sequences for applications in sequence recognition or time series prediction, we can often get an accurate predictive output. This requires a system to store and analyze information computed from past inputs in order to produce the desired useful resulting output. Since recurrent neural networks (RNNs) have an internal state that well-represents time series, they are uniquely suited for the task of prediction relating to time series datasets. The construction of a recurrent neural network is such that it allows retaining of past inputs for analysis while weighting it according to its timestep or 'distance' from the current index. A recurrent neural network can be used to transform an input sequence into an output sequence, whereas static artificial neural networks (ANN), which do not have that context memory, cannot store information for an indefinite period.

Analysis of market data is difficult to perform, due to its cyclic and seasonal variations, as well as the irregular and seemingly random movements these data exhibit[3]. These factors, among others, have contributed to many machine learning techniques failing to demonstrate success in price prediction[4]. The advent of deep learning techniques, such as LSTM, led some researchers to investigate whether these could provide more success. While some researchers have claimed a degree of success in using RNNs with LSTM to perform market prediction[5], however the degree of this success is limited. By maintaining a small window size with which to train their RNNs, the models in these prior attempts react primarily to only very recent data, creating a predicted data set that while appearing to be accurate, in fact provides predictions only a very short time ahead. Our experimental goal was to use an RNN with LSTM to accurately predict the price of futures contracts, as well as to determine how far ahead in

time this technique can assist in prediction.

When analyzing time series datasets, there are two main goals. The first is in modeling the time series in order to gain an understanding of the underlying mechanisms at work to generate the data. The second is forecasting or predicting of future values based on current or previous values in the time series[6]. The main idea of using a learning algorithm in recurrent neural networks is to compute the gradient descent of a cost function regarding the weights of the network. This is particularly useful for the task of prediction with time series datasets. Time series data by their nature are often high-dimensional. It is useful, therefore, to attempt to reduce the dimensionality when analyzing time series datasets. There are two common approaches to this. The first, and most simple, is "sampling", and consists of simply taking the values of the data at regular intervals. The problem with this method is that it has the potential to distort the overall shape of the data unless the sampling rate is sufficiently high. However, high sample rates are what we are trying to avoid. The other common method of dimensionality reduction is to calculate the mean of each segment or bin[7]. The latter method is what we employed to produce single contiguous time series to feed to the RNN.

1.1 Definitions & Background

1.1.1 Recurrent Neural Network (RNN)

An RNN is a class of artificial neural network in which connections between units form a directed cycle. This allows a RNN to exhibit temporal behaviour. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks where each piece is dependent in some fashion on the one that preceded it, such as unsegmented, connected handwriting or speech recognition[8].

RNNs can be configured to accept one or more vectors of input to produce one or more vectors of output. Possible configurations of RNN input and output are one-to-one (one input following through to one output), one-to-many (one input resulting in many outputs), and many-to-many (many inputs resulting in many outputs with each input either resulting in its own output or in different or different numbers of outputs). For our experiments, we employed a many-to-one configuration, providing the RNN multiple vectors of input to produce a single output stream of data. Since we limited our investigations to examining the RNN's predictive ability on a single futures security this justified, it made sense to only consider that as a single output. We chose the many-to-one configuration to minimize the the risk of ignoring factors which affected the output vector of interest, but in a manner we could not foresee.

1.1.2 Long Short-Term Memory (LSTM)

Long short-term memory networks are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber in 1997, and were refined and popularized by numerous people in subsequent works. LSTM algorithms work very well on a wide variety of problems, and are now widely used in time series predictions. LSTM's are explicitly designed to avoid the long-term dependency problem, whereby a current value is in some way dependent on another that long preceded it. By default, they are designed to retain information for long periods of time. Because of this, they do not struggle in dealing with extended time periods[9].

All recurrent neural networks are in the form of a chain of repeating modules. In standard RNN's, this repeating module has a very simple structure, such as a single tanh layer.

In the detailed workings of a RNN with LSTM[9], X_t is the input at time t, and h_t is the output at time t. The repeating modules of the RNN are denoted by A. The sigma layers determine values to pass on, and their weighting. The tanh layer generates a vector of new candidate values that could be added to the state. The 'x' and '+' are gates that allow certain values through.

1.2 Additional Definitions

1.2.1 Machine Learning

The concept of machine learning has been around for well over half a century. However, it has been only within the past few decades that computing power and understanding has allowed for great advances in this area. The neural network is a part of the topic of machine learning. The design and implementation of neural networks was inspired by the study of naturally occurring biological systems, such as the human brain. The idea that there are nerve cells which process stimuli and communicate with neighbouring cells to produce a response was the foundation of neural network design[10]. Neural networks have shown to be extremely accurate at prediction of future values based on previous information, and have been successfully implemented in various fields from facial recognition to intrusion detection in computer security. Because of the wide applicability of neural networks, they continue to be a field of active research, and have been receiving much attention.

1.2.2 Deep Learning

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but

non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. The key aspect of deep learning is that these layers of features are not designed by human engineers — they are learned from data using a general-purpose learning procedure.

1.2.3 Time Series Dataset

A time series dataset is a dataset consisting of a series of values or readings taken at certain intervals over a period of time. Time series datasets are most often quite long, and are considered to be smooth, i.e. neighbouring values are within predictable ranges of each other, as opposed to being completely random[11]. Time series data, due to its nature, has some inherent difficulties that have to be overcome to be properly analyzed. First, they often contain quite a bit of signal noise and high degree of dimensionality. This needs to be accounted for when processing the datasets. Second, time series datasets may not contain enough information to properly understand the processes involved. In other words, there may be variables outside of the scope of the data that affect the values recorded. Especially when working with a complex system such as financial data, there can be nearly an unlimited number of outside factors that could affect the values.

Another difficulty with time series data is the time dependence. For example, a prediction using a specific value at a certain time will not necessarily provide the same result as a prediction using the same value at a different time. To combat this issue, it may be necessary to include more past data, or keep memory of inputs. This can lead to rapidly growing memory requirements, especially when the length of dependencies may not be known with any certainty. A related difficulty is that time series datasets are commonly non-stationary. This means that over time, the characteristics of the data may change or shift. This is often handled by treating the data in the frequency domain, rather than the time-domain, by treating them with wavelet or Fourier transforms.

Finally, most features used with time series data must be invariant with respect to translation in time: given the same set of inputs, we should not expect a different output, simply because the inputs occurred at a different time[12]. In the case of financial data such as with the futures market, this is not the case, as transactions occur at seemingly random intervals. This issue can be resolved during preprocessing by grouping or "binning" the data.

1.2.4 Prediction

In times series datasets, prediction is the use of the knowledge that data points in time series datasets are generally within predictable ranges of each other in order to determine the next few values of a series[1].

1.2.5 Futures

Futures contracts are legal agreements to either buy or to sell an item (most often a commodity of some sort) for a specified price at a future point in time. Whereas the stock markets trade stocks in individual companies, futures trading involves the trading of contracts. One thing to note is that futures contracts eventually expire (mature), at which time the holder of the contract could be obliged to fulfil the demands of the contract. This is an interesting quirk of futures trading that must be considered when making predictions or analyzing patterns in trading. As the expiration approaches, contracts are rolled over to the next time period, and trading generally decreases up until the contract is expired[13]. The contract then has a new expiry date. This leads to some preprocessing challenges in producing a single contiguous time series for use in an RNN.

1.2.6 TMX

The Bourse de Montréal, or Montreal Exchange (TMX), was founded in 1832 in Montréal, Québec, and is Canada's oldest exchange. It is the only derivatives exchange in Canada, with expertise in Financial Derivatives Markets, Clearing Services, Data Analytics, and Information Technology Solutions[14].

1.2.7 BAX

The BAX futures contracts are three-month Canadian Bankers' Acceptance Futures traded on the Montreal Exchange. Bankers' acceptances are short-term debt obligations that are backed by a major bank. Because they are backed, the payment of principal and interest on the debt is guaranteed. Investors can purchase these contracts at a discount based on yield, and collect face value at maturity. The maturity period of bankers' acceptance contracts generally ranges from 30 days to one year. Bankers' acceptances were first introduced in Canada in 1962, and BAX futures were the first interest rate contracts to be traded on the Montreal Exchange (BAX is a trademark of TMX[15]). The BAX futures contracts were the futures contracts analyzed for the purposes of this report.

2 Methods

2.1 Description of the Dataset

The dataset consisted of nine months' worth of event history from the Montreal Futures Exchange (TMX), from January to September 2015. This dataset comprised numerous different futures products, and all events related to these products. Examples of related events included bids, asks, trades,

Table 1: *Summary of trading records on TMX, Jan.–Sept. 2015*

Symbol	Records
BAX	1,114,781
CGB	4,922,494
CGF	6,880
CGZ	43
SCF	45
SXA	3
SXF	2,434,429
SXK	5
SXM	13,235
SXY	4
Total	8,491,919

and other specialized events, related to such activities as options and strategy trading. A bid event is defined as an offer to purchase one or more contracts of a specific future for a certain price. An ask event is defined as an offer to sell one or more contracts of a future at a specified price. A trade occurs whenever the prices of an ask and a bid match, and a transaction occurs.

The dataset was provided in 187 comma-delimited flat files, one containing all events for each of the 187 trading days that occurred between January 2 and September 30, 2015. These files, comprising some 80 million records were imported into a Microsoft SQL Server database table for easier filtering and processing. Since our analysis was concerned with the actual trading price of the futures contracts, we dealt only with actual trade events, filtering out all of the other events that did not result in futures contracts changing hands. Including only records that consisted of actual trades resulted in a dataset of about 8.5 million records, consisting of the trades that occurred over all futures products and contracts. A summary of these records can be seen in Table 1.

Many of these products were traded infrequently, with irregular intervals between each series of trades. Of the 10 futures traded on the TMX, only BAX contracts were traded in high volumes on each of the 187 trading days for which we had data. As a result, we decided to focus our study on this particular futures symbol. There were 1,114,781 BAX-symbol trades that occurred over the nine-month period.

Trades occur randomly and at irregular intervals, however RNNs require continuous time series data which is spaced at regular intervals. This was handled by placing the data from all trades into regularly-spaced bins, each of which contained the total volume of trades for a contract, as well as the average price at which it traded during the time interval of the bin. Trading occurs on the TMX from 6:00 a.m. to 4:00 p.m., Mondays to Fridays, excluding holidays. Since an RNN requires a high number of records in its training data set, we decided to maximize the number of bins that our data set could support. This goal was limited by the requirement to ensure that the bin size was large enough to ensure that all bins were populated with trading activity. By running SQL queries on the data set, we determined that while a half-hour bin size did not ensure all bins were populated, maintaining the bin size at 1 hour ensured that trades occurred for each front-end BAX contract in that bin interval.

Analyzing futures trading with an RNN is further complicated by the fact that futures contracts are time-limited, with one contract expiring every three months. For example, the BAXH15 contract matured in March 2015, while the BAXM16 contract expired in June 2016. A contract which is not delivered at maturity is automatically "rolled over" into the next-expiring contract. For example, anyone left holding the BAXH15 contract when it expired in March 2015 had their holdings automatically rolled over to the BAXM15 contract. As such, if we consider only the front-end contract (the one maturing next), it is possible to analyze the data as one perpetual time series, rather than a set of several shorter-duration time series, if we have a means to handle the rollover.

Several different techniques that can be employed to handle this problem[13], each with their own strengths and weaknesses. We chose to employ the perpetual time series model, to ensure as smooth as possible a transition between the front-end contracts during the rollover. This technique avoids the uncertainty associated with guessing at which point the activity of the new contract supersedes that of the expiring contract by treating the trading data as a weighted average of the two contracts[13]. This method suffers from the fact that, since it averages the price between two contracts, it does not necessarily provide the "real" price of any specific contract. That said, of all of the techniques used to splice futures trading data, this one provides the smoothest transition[13] and is thus best suited to provide input data to an RNN.

Rather than choosing an arbitrary smoothing period over which to average the two contracts (without any *a priori* knowledge of the optimal period to choose), we determined to average the two contracts with the nearest expiry dates on a sliding scale to produce a perpetual time series using the following algorithm:

1. On the expiry date (last trading bin), t_0, of the expiring contract, C_0, count the number of bins, N, to the expiry of the next expiring contract, C_1
2. For t_i, $0 \leq i \leq N$, using data from C_1 and C_2, calculate:
    a. $\text{volume}(t_i) = [\text{i}\times 7\text{volume}_{C2}(t_i) + (N - i)\times\text{volume}_{C1}(t_i)]/N$

b. $\text{price}(t_i) = [\text{i} \times \text{price}_{C2}(t_i) \times \text{volume}_{C2}(t_i) + (N - i) \times \text{price}_{C1}(t_i) \times \text{volume}_{C1}(t_i)] / [\text{i} \times \text{volume}_{C2}(t_i) + (N - i) \times \text{volume}_{C1}(t_i)]$

This algorithm was implemented in an SQL query. The resulting data set consisted of 1870 records (one for each hour of 187 trading days). Graphs of these data can be seen in Fig. 1 and 2.

While the volume data appear noisy and chaotic, the data for price appear to follow an orderly enough pattern for an RNN to predict. While the volume data may have been far too noisy to predict with any degree of accuracy, we observed that the large spikes in trading volume corresponded with accompanying increases and decreases in price. As a result, we determined to include the volume data as an input to the RNN (along with the price), to generate a predicted price.

2.2 Execution

When we try to decide the parameters we use for our training process, we need to choose them considering their trade-offs with respect to time and accuracy. For example, if the accuracy of our prediction did not improve significantly beyond a certain number of iterations, increasing them further serves to greatly increase processing time with diminishing returns. Our goal in experimentation is to find an optimum balance between these factors. In an RNN, the epoch refers to the number of iterations used to train our neural network. The RNN processes the data in batches, a certain number of records at a time. Once the RNN has processed the entire training data set (one epoch), it attempts to minimize the loss/error on each batch of records by updating the weights on each batch. The neural network tries to find the optimal value where the error of a batch of data can be minimal for all epochs.

The loss (error) remaining at the completion of each epoch of training is shown in Fig. 3. Following a steep reduction between 1 and 20 epochs, further reduction is greatly limited. Since almost no further reduction in loss was observed for several epochs leading up to epoch 100, this was the number of epochs we chose for our experiments.

Ideally, with unlimited processing power, we would analyze each line of the training dataset and find the perfect value to reduce the loss value to zero. However, this scenario, corresponding to a batch size of 1, would be incredibly inefficient. Further, such a scenario would also be exceptionally susceptible to noise in the dataset. Rather, to reduce the processing we require, while minimizing the impact caused by outlier data, the RNN processes the data in larger batches at once, updating the mean weights of the data points as it proceeds.

Plots of the processing time and final loss value determined for batch sizes of 16, 32, and 64 for runs of 30 epochs and 60 time step values per prediction point are shown in Fig. 4. We observe that processing time required declined consistently as the batch size was increased, as we would expect. Moreover, an excessively-large batch size resulted in an increased loss. The most interesting observation from Fig. 4 is that, beyond a certain point, a reduction in batch size does not result in an improvement in the final loss value.

Based on these results, we determined our optimum batch size to be on the order of 32. At this level, we optimize the accuracy of the prediction, while limiting the length of time required in processing the training data.

The number of time steps (records) used in the prediction is another parameter of the RNN. As with the previous parameters, the desired accuracy of the prediction must be weighed against the processing time required to achieve it.

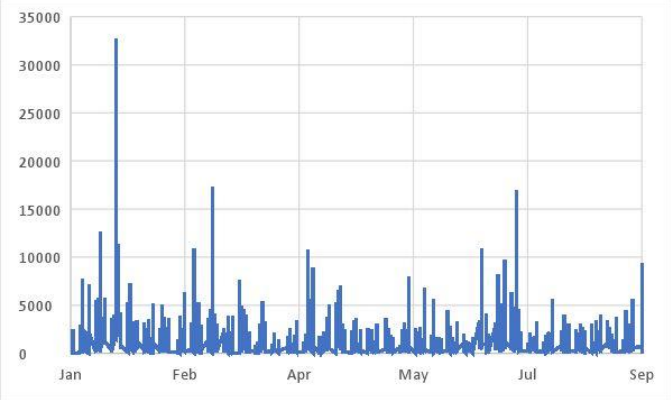

Figure 1: *Spliced BAX front-end contract trading volume data, grouped into 1-hour bins.*

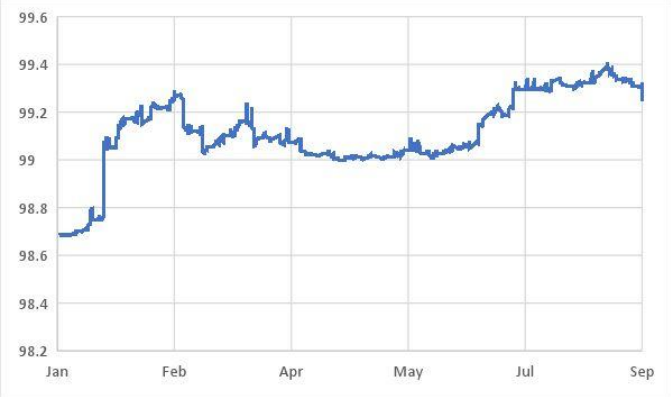

Figure 2: *Spliced BAX front-end contract trading price data, grouped into 1-hour bins.*

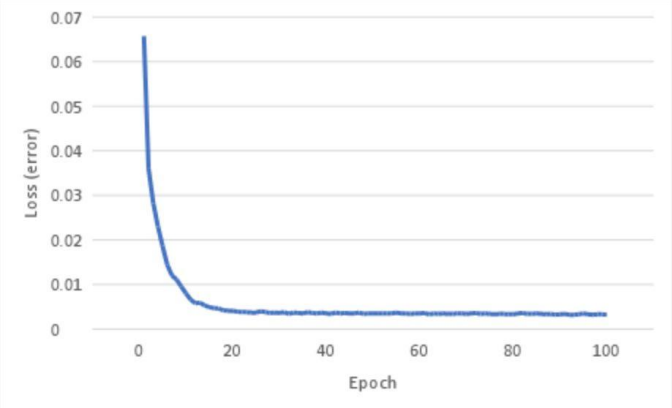

Figure 3: *The loss remaining after the completion of each epoch (batch size = 32).*

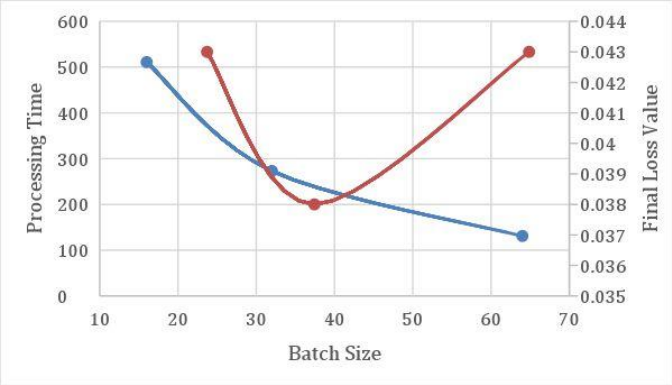

Figure 4: *Processing time required (blue) and final loss value (orange), by batch size.*

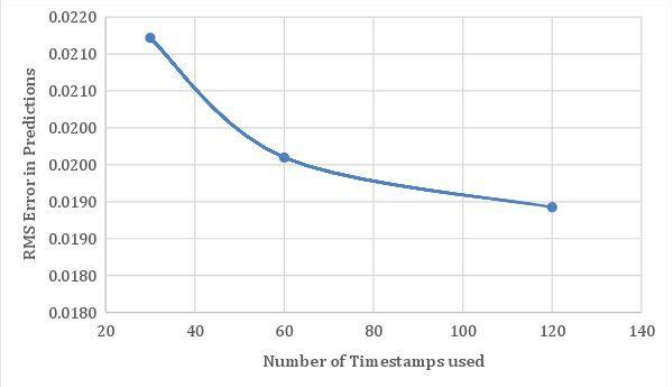

Figure 5: *Root-Mean-Square Error over all predicted values, 1 bin ahead, for 100 epochs and batch size of 32.*

Figure 5 shows the the RMSE over all predicted values for varying numbers of time steps used. As the figure shows, while increasing the number of time steps decreases the overall error, it does so with diminishing returns. Since we found using more than 120 time steps led to excessively long processing times, we used a time step value of 120 records for our experiments to minimize the error, while keeping the processing time manageable

An RNN with LSTM works by back-propagating the weightings between layers using an error-minimization strategy. The choice of which strategy to employ depends on the data we are trying to predict. One of the simplest, and most commonly-used error measurements is the root-mean-square (RMS) measurement. Since our aim for this experiment was to minimize the error in all predicted price measurements, we employed an RMS propagation LSTM, which keeps a moving average of the squared gradients for each weight and divides it by its mean squared to fix the learning rate.

Using these determined parameters for our RNN, our team decided to test the RNN's capability to predict the price of the front-end BAX contract, for each of 1, 3, 5, 10, 20, and 30 bin intervals in the future. For each of our experiments, our team fed the first 1700 records of the processed data set into the RNN as training data, using the remaining 187 records to perform the predictions and compare the results.

3 Results

The results of the experiments are shown in Fig. 6–11. As is demonstrated by these charts, in none of the experiments was the RNN able to pinpoint the price. Moreover, the accuracy of its predictions appear to decline considerably the further in the future its predictions are projected. Figures 10 and 11 in particular, demonstrate very little relationship between the predicted and actual price for predictions 20 and 30 bins in the future, respectively. Notably, since we were working with bin sizes of one hour for 10-hour trading days, this means the RNN showed little predictive capability only two and three days in the future.

4 Discussion and Conclusions

After conducting six experiments with varying numbers of time steps it was noticed that an RNN is not useful for predicting the price with any real accuracy, particularly for predictions of more than 10 bins (one day) ahead. While the RNN seems able to sense price changes as (or after) they occur, even the short-range predictions appear unable to reflect sudden or sharp price movements, tracing these as fluid curves rather than as they occur, as sharp spikes in price. This indicates that the RNN is merely reflecting the movement it is detecting in its most current data, rather than predicting future movements, based on its prior learning.

The accuracy of the predictions (predictably) decrease as the RNN is tasked to range its predictions even a couple days (20 bins) into the future. This, coupled with the lag observed between when price movements begin occurring and when

they are observed in the RNN's predictions suggest an RNN to be, at least as we have implemented it, of limited utility as a price-predictive tool. That said, the RNN's ability to sense and reflect trends and movement in the data suggest that it may show some promise as a tool to detecting patterns, but with the caveat that the time range it is analyzing not be extended too far beyond the data it is provided.

Given the RNN's sensitivity to fluctuations in the provided data, it would be an interesting extension to these experiments to see how the results would be affected by changing the bin size. Some sizes that may produce interesting results are two hours, up to a whole day. Grouping the data into larger bins would result in reducing the noise levels of the input data, potentially improving the RNN's performance.

Another interesting experiment that could be done is to run these experiments using an unsupervised learning neural network. The RNN used for the purposes of this paper was a supervised network, meaning that we needed to train the network on preselected data before feeding it the data used to make its predictions. It would be interesting to see the results of an unsupervised neural network could provide, particularly when provided more input vectors to analyze. Given the ability of RNNs to detect patterns, such an experiment could potentially reveal unforeseen relationships in trading between multiple contracts.

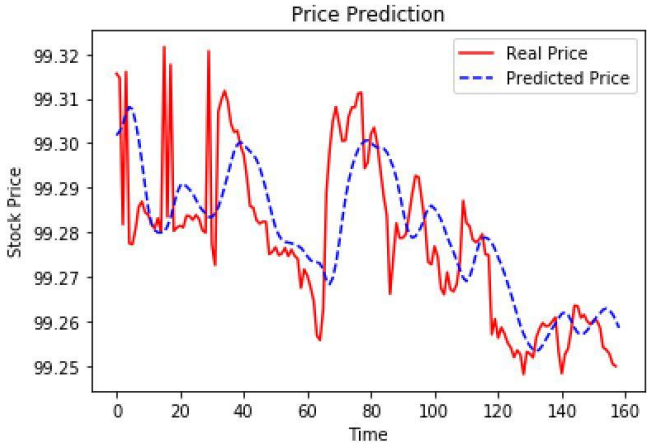

Figure 6: *Predictions 1 bin ahead.*

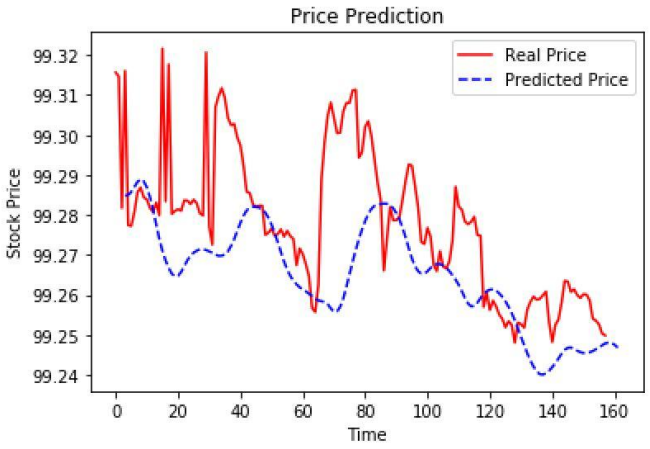

Figure 7: *Predictions 3 bins ahead.*

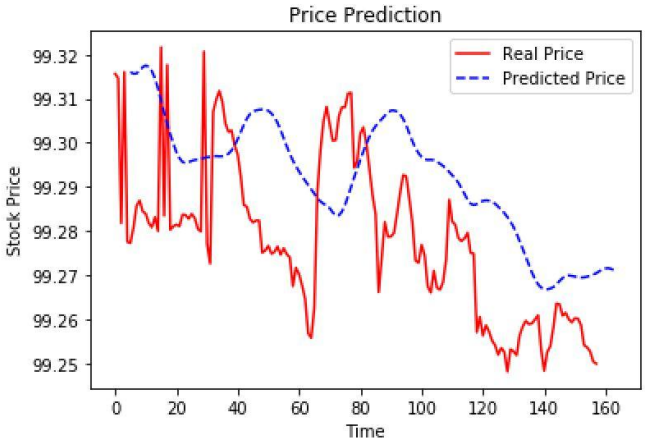

Figure 8: *Predictions 5 bins ahead.*

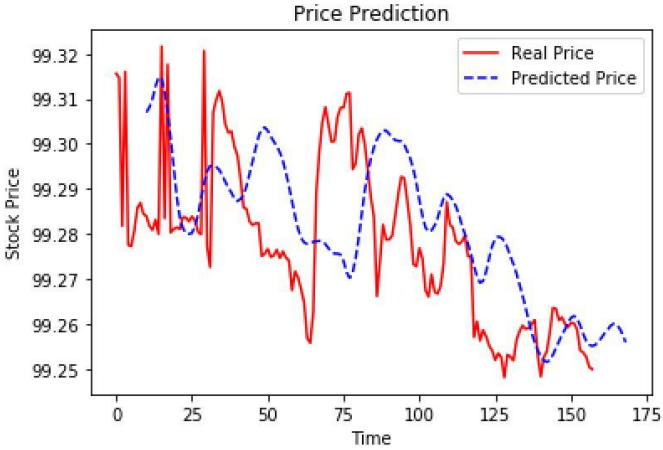

Figure 9: *Predictions 10 bins ahead.*

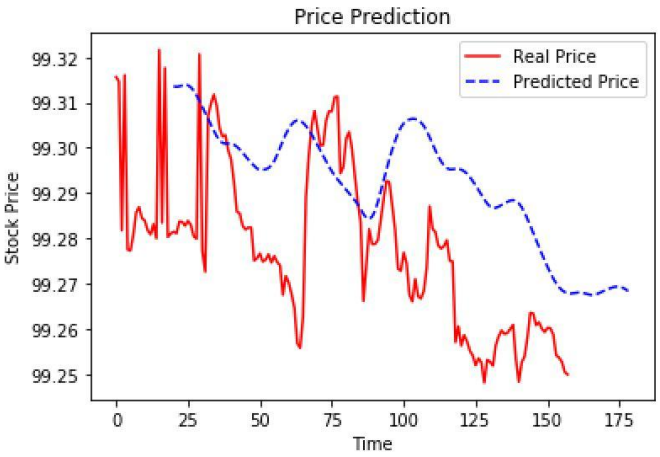

Figure 10: *Predictions 20 bins ahead.*

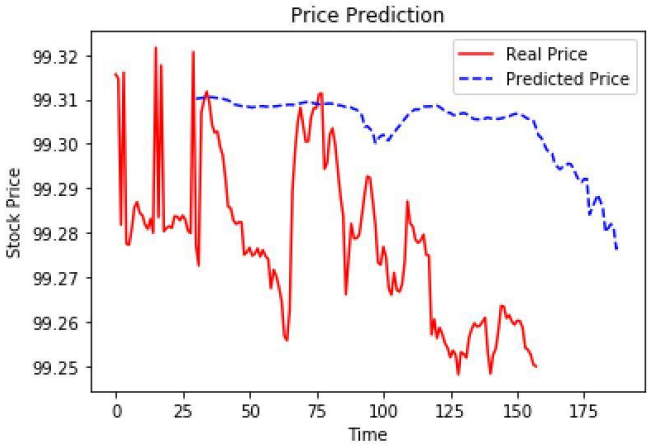

Figure 11: *Predictions 30 bins ahead.*

References

1. Esling, P. & Agon, C. 2012. *ACM Computing Surveys*, 45: 12.
2. Rather, A. M., Agarwal, A., & Sastry, V. N. 2015. *Expert Systems with Applications*, 42: 3234–3241.
3. Rout, A. K., Dash, P., Dash, R., *et al.* 2017. *Journal of King Saud University — Computer and Information Sciences*, 29: 536–552, doi:https://doi.org/10.1016/j.jksuci.2015.06.002.
4. Samarawickrama, A. J. P. & Fernando, T. G. I. 2017. *In: 12th IEEE International Conference on Industrial and Information Systems (ICIIS)*, 1–5.
5. Roondiwalal, M., Patel, H., & Varma, S. 2015. *International Journal of Science and Research*, 6: 1754–1757.
6. Han, J., Kamber, M., & Pei, J. 2012. *Data Mining Concepts and Techniques*. 3rd edition, Morgann Kaufmann.
7. Fu, T.-C. 2011. *Engineering Applications of Artificial Intelligence*, 24: 164–181.
8. Pascanu, R., Mikolov, T., & Bengio, Y. 2013. *In: Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28, 1310–1318.
9. Olah, C. Aug 27, 2015, Understanding LSTM Networks. URL `colah.github.io/posts/2015-08-Understanding-LSTMs`.
10. Tan, P.-N., Steinbach, M., & Kumar, V. 2005. *Introduction to Data Mining*. Pearson.
11. Shasha, D. & Yunyue, Z. 2004. *High Performance Discovery in Time Series: Techniques and Case Studies*. Springer Verlag.
12. Längkvist, M., Karlsson, L., & Loutfi, A. 2014. *Pattern Recognition Letters*, 42: 11–24.
13. Masteika, S., Ruthauskas, A. V., & Alexander, J. A. 2012. *International Conference on Economics, Business and Marketing Management, IPEDR 2012*, 29: 265–269.
14. Montreal Exchange. 2017, Montreal Exchange: Canadian Derivates Exchange. URL `m-x.ca/accueil_en.php`.
15. Bourse de Montreal Inc. May 2009. BAX Three-Month Canadian Bankers' Acceptance Futures.