Novel Deep Learning Trading Bot Strategies

Austin Apt   
*Department of Computer Science*  
*University of Georgia*Athens, USA  
[ama66843@uga.edu](mailto:ama66843@uga.edu)

*Abstract*—.

Automated trading systems have seen increasingly wide use since the 1980’s. These are agents capable of automatically executing buy and sell orders on a commodity exchange ordering to some set of rules. Recent developments of commission-free trading and the 24/7 cryptocurrency market present two new incentives for the development of always-online trading bots. This paper presents state-of-the-art automated trading systems built using supervised and reinforcement learning-based algorithms to measure their performance on the market.

Keywords-Automated Trading Systems, Deep Q Networks, Convolutional Time Series, Long-Short Term Memory, Cryptocurrency

The code for this paper here: https://github.com/ama66843/ ConvolutionalCloudWorkloadAnalysis

# Introduction

There are more opportunities than ever for individuals to invest with the advent of commission-free trading providers such as Robinhood and Alpaca. These companies typically offer relevant application programming interfaces (API) which allow individuals to access the market and automatically execute buy and sell orders using a scripting language like Python. This has driven the continued development of trading bots, which are examples of automated trading systems (ATS). An ATS can be setup to check the market for any given time interval and execute an arbitrarily large number of orders faster than any human, while also saving the account holder from having to constantly check the market by hand.

Classical ATS systems typically consist of expert-curated features and indicators such as a stock’s relative strength index (RSI), moving average convergence divergence (MACD) and Ichimoku Cloud structure. Rules such as the 70/30 are used with RSI in order to create a buy/sell signal (acc. to [1], 66.6 and 33.3 are truer indicators of bull and bear markets than 70/30 or 80/20 rules). More specifically, when the RSI reaches a value of 70, this indicates that the stock is overbought by the market at large and should be sold by reasoning that a downward correction will shortly follow. Conversely, when the RSI reaches 30, this designates that the stock is oversold and may shortly have a positive correction.

The RSI is only one of many man-made indicators which can be used as a basis for an agent’s buy and sell signals. Another is the Ichimoku Cloud which uses four different moving averages, and a lagged “closing price” feature line. When the Leading Span A (average of 9-day and 26-day MA’s) is above Leading Span B derived from a 52-day MA, it signifies an uptrend and that recent movements suggest a higher price support and bull market conditions. The Ichimoku Cloud is often combined with RSI in order to maximize risk-adjusted returns.

More recently, ATS’s have been created using machine learning (ML) principles, as seen across most fields with appropriate time series data. This includes the application of long-short term memory (LSTM) [2], which are examples of recurrent neural networks (RNN) with the addition of multiple non-linear activation gates to mimic a human’s ability to remember and forget. In 2016, Google Deepmind presented WaveNet, a causal dilated 1-d CNN which was used to generate and predict audio waveforms. Instead of being applied to images and videos, this model presented a CNN which applies 1-d convolutions on sequenced or windowed time series input data. Causal denotes that only past observed information is considered for future predictions and dilated refers to the way in which the receptive field grows exponentially for each convolutional layer as shown below. Similarly to the LSTM, this architecture is an attempt to capture clear present information while also maintaining some effective representation of data that may be far in the past.

Another approach to creating an ATS system is to format the stock market as a playing field for a reinforcement-learning (RL) based agent. In RL models, an agent is able to teach itself through its own repeated interactions within its observed environment. Instead of using an error function to compare results to actual values (this is more desired for regression or classification problems), the agent has some notion of a reward function which it will continue to improve on through being in the environment. At a given time step, usually known as a state *s*, the agent can take some action *a*, and then receive a reward *R* and be in a new state *s’*. In the case of this paper, the state or environment is current incoming cryptocurrency or stock price data, and the agent’s current portfolio split between liquid dollars and commodities. The available action space consists of, e.g. making (1) buy and (2) sell orders, or holding (3) a commodity at each given time step or state given.

TensorTrade is a specialized implementation of the general deep RL framework Tensorforce, which is itself branched from TensorFlow. The package consists of a variety of modular components which allows the user to quickly test a variety of different trading bots and schemes. The implementation discussed here uses a Deep-Q Network (DQN) as the basis for agent actions. DQN’s are an extension of Q-Learning (QL) where actions are chosen using a cost function that incorporates current and subsequent future rewards due to that action. At each step or state s, QL creates a Q-table (expressed Q[S,A]) where all possible actions, and the subsequent states s’ they bring the agent to are considered. Q’ represents the ideal policy to take at that step.

Depending on the state and action space, this can become prohibitively expensive to update at each iteration of the agent’s exploration e.g. 10,000 states and 1,000 possible actions creates a table of size 10,000,000. DQN’s attempt to solve this issue by using a neural network to approximate the ideal Q-value function at each state. Instead of a Q table being calculated for every state and action, only the current state is used as input to a DQN and Q-values are output for all possible actions.

# Related Work

There has been extensive research in using time series models to capture and predict commodity price movement. Classical methods like linear regression and ARMA models have been used, as well as classifier-based methods using SVM or Random Forests which use additional features to predict when stock movements occur. The highest performing models use some form of LSTM in order to learn and predict future price values.

This paper presents a further evolution of an LSTM network applied to stock price data, as well as a reinforcement-learning based model implementation.

These existing models have used data survey intervals ranging from 5-30 minutes, with longer intervals generally Chart

Description automatically generated  
  
Fig. 1. Google Cluster (5-minute intervals)

allowing for smoother data and fewer demand spikes. While the Google cluster data is available at a resolution of one-minute, most machine apportioning occurs every five minutes. Note that   
by averaging data to increase machine learning model performance, fine granularity is lost. VM allocation is a task which requires small resolution in order to proactively create machines to serve users, while long term predictions allow for IT professionals to estimate hardware costs and look for general usage patterns throughout the day.

# Approach

## Problem Definition

A typical trader would like to know commodity price at some point in the future, for example tomorrow or next month in order to make a buy or sell decision today. This can be predicted given some past sequence of daily stock open, close, high, low and volume information in the most basic sense.

The data used here is hourly to take advantage of the fine granularity available for cryptocurrency data. The objective is to find a time series model, which when given some past sequence of data, can accurately predict overall commodity price at least one timestep ahead.

## Design

Text

Description automatically generated  
  
 (b) Alibaba Cluster (5-minute intervals)

The remaining models were trained by sequencing the five-minute interval data into sliding windows of width eight, representing forty-minutes worth of data. This is represented by the equation (1) below with n=8, with the model predicting CPU use one timestep ahead.

P(t) *f*(P(t-1), P(t-2),...,P(t-n+1), P(t-n)) 

This value must be weighed and balanced against both hyperparameter and practical considerations. While there is some optimal model value for this sequence length, the model becomes computationally more expensive the longer it is; it then also requires that much more data to create predictions. Window widths were tested between eight and twenty, representing 40-100 minute intervals and following general guideline values denoted in the LoadDynamics [12] paper.

## Long Short Term Memory Model

The LSTM [14] cell was created in order to capture both long and short-term dependencies when trained on time series data. It is a variant of a recurrent neural network (RNN) with extra gated functions meant to mimic the human brain’s decision to store or forget information up to that point. Generally speaking, these additional non-linear activation gates avoid the vanishing gradient and missing long-term dependency problems of traditional RNN’s.

Neural networks built with LSTM cells have been used extensively in recent history, commonly outperforming the most complex existing linear models e.g. SARIMA. This has occurred across a wide variety of fields where time series data is involved: natural language processing, stock price and traffic flow prediction, etc.

Diagram

Description automatically generated

Fig. 2. LSTM Cell with input data at bottom left

Hyperparameter optimization is required when building a large neural network if the model is expected to perform best on that one specific dataset. This paper uses a small grid search based off values for the Google trace dataset in similar research. Due to the relative insensitivity to gap length when training as compared to traditional RNN, most LSTM models still perform well if built with a sufficiently complex network architecture. Because these models will purposefully be exposed to unseen datasets to measure their robustness, it is not vital that they be completely optimized on the Google trace.

## Causal Dilated Convolutional Net Model

Convolutional neural nets work by applying kernels, defined by some width and height, to input data. The variety of kernels aim to extract a feature map which is representative of the input data. Similarly to LSTM models, CNN are inspired by biological processes. It is based off the human eye, which functions as one organ but contains millions of individual cortical neurons which individually only respond to a restricted region known as the “receptive field” [15] which is analogous to the kernel passing over small portions of input data. Historically they have been used to great success on image and video classification tasks. Since Google Deepmind’s WaveNet paper 1-d CNN have been applied to more traditional time series tasks such as stock price and traffic prediction with mixed results. There is even a successful implementation of a Graph WaveNet [16]. Due to the noise seen in cluster data, especially the Google trace, this paper implements and applies a WaveNet-style CNN to model the time-series prepared datasets.

WaveNet utilizes a causal dilated convolution net. Causal denotes that the filters are only applied to past sequences of data relative to the current timestep and dilated refers to the way in which the model grows its receptive field. The convolutional filter grows in width exponentially as it covers increasingly larger causal past sequences, while avoiding being computationally prohibitive. (2) shows how a list containing dilations with a base of 2 are created in pseudocode.

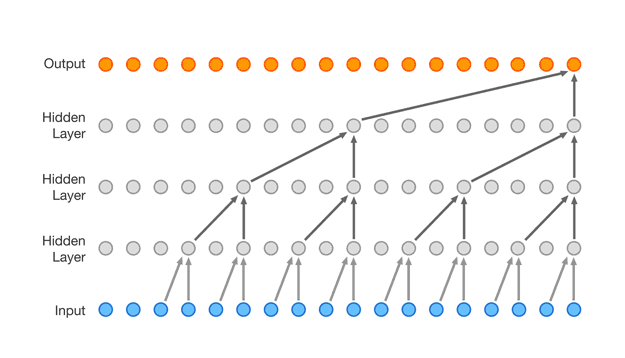


Fig 3. Wavenet CNN with dilation rate of 2

*dilation\_rates* = [2\*\*i for i=0 to 7] (2)

This paper implements an architecture similar to that pictured in Fig. 2. but with the receptive field eventually expanding to 64.

Hyperparameter tuning is more important for CNN than most LSTM when building models which approach LSTM baselines. Such values are the aforementioned dilation rate, the number of layers in the network, learning rate, number and size of the kernels in each layer, batch size, activation function, the addition of dense and/or dropout layers, etc. Due to time constraints, it is likely that a WaveNet built with a different set of hyperparameters would perform better than the basic model presented in this paper.

# Experiments

This section presents the experimental portion of the project. After being trained on the larger and older Google trace dataset, each model was evaluated on the unseen Alibaba dataset in order to measure their relative generalizability.

## Experimental Setup

**Workloads:** Two workloads were used to train and evaluate the LSTM and WaveNet-style models. They are the 2011 Google and 2017 Alibaba cluster traces. The CPU and memory utilization of each can be seen in Fig. 1 after the datasets were summarized into five-minute time intervals. Although the raw Google dataset is available at one-minute resolution, the number of job statistics is magnitudes greater for start time statistics every five-minutes.

**Metrics:** Models were trained using mean squared error (MSE) as a loss function, which is formulated as the sum of squared (prediction) errors (SSE) divided by the number of samples. Loss values were collected along with R2 and mean absolute error (MAE) statistics for each model.

. **Chart, scatter chart

Description automatically generated**

Fig. 4. Google CPU Usage Lag-1 Plot

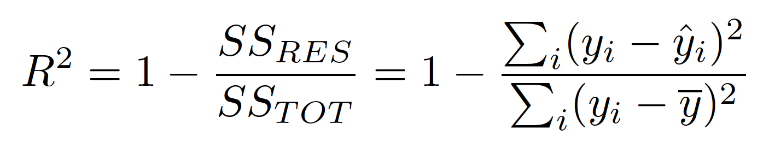


Fig. 5: R2 value expanded

**Baselines:** A lag-1 model was used as a baseline where future value is set equal to the current value. Fig. 4 shows a high correlation between adjacent data points for the Google trace. R2 values are given by the equation shown in Fig. 5. The form of the R2 equation shows that its value is a measure of how well a model’s predictions perform compared to using a naïve mean as output prediction model.

**Implementation:** The models were built using sklearn, pandas, tensorflow and keras. Training and tests were completed on a 8-core local machine with Intel i7-8550U CPU and 16GB RAM. Models were built using both MSE and MAE as loss functions during initial testing before deciding to use MSE for the models displayed in this paper from better performance. MAE and R2 values are collected in order to compare model performance on the datasets.

## Evaluation of Model Accuracy

Datasets were split into training and test datasets using a 80/20 split, with MAE and R2 values collected for each portion. After a small grid search, five models were chosen for evaluation with three LSTM networks and two CNN’s. All excepting one LSTM network built with input sequences of length 20 were built using length 8 due to better performance on the Google cluster.

Basic descriptions of the three LSTM models are that two used two layers of 64 LSTM nodes each, with one model using two layers of 128 LSTM cells. One of the two 64 node models was trained using a sequence length of 20. Both convolutional models were 1-d CNN’s consisting of seven causal layers with a dilation rate of 2 which increased its receptive field to 64 sequences. The only difference between the two models was that one was trained with batch size 32, and the other 64. Epoch size of 350 was used for CNN models according to early stopping rule, and 500 for the LSTM networks.

Chart, waterfall chart

Description automatically generated

Fig. 6. Models Evaluated on Google Cluster Trace

Fig. 6. shows R2 and MAE statistics for each of the six models (including the Lag-1 baseline). While all models performed better than naïve-mean on the training data, both CNN returned negative R2 values on the test data. All three LSTM networks performed better than the Lag-1 baseline which returned R2 values roughly equal to .5 for both training and test sets.

Chart, waterfall chart

Description automatically generated

Fig. 7. Percent Difference Compared to Lag-1 Baseline

Fig. 7. shows the percent difference in R2 and MAE values as compared to the Baseline values displayed in Table 1. Positive difference of R2 values is good, while MAE is desired to be decreased. Both CNN’s improved on MAE for the training data set, but returned slightly higher values for testing. Having shown that these models perform better than baseline in at least one measure (all for the LSTM networks), they were set to predict future CPU usage given time series data from the Alibaba cluster trace. These performance metrics are shown in Fig. 8.

Chart, scatter chart

Description automatically generated

Fig. 8. Alibaba CPU Usage Lag-1 Plot

Chart, waterfall chart

Description automatically generated

Fig. 9. Models Trained on Google Used to Predict Alibaba Cluster

Interestingly, the Lag-1 Baseline Model had a much larger spread in value for the Alibaba dataset compared to Google for the test and training datasets. It performed roughly five times better for test data than training. By exposing the trained models to unseen data, the paper measures the relative robusticity and generalizability of the models to different workload types.

Fig. 10 shows how the LSTM models were able to achieve better R2 values on all training portions of the Alibaba trace. Most impressive is the test R2 of the LSTM model trained on sequences of length 20. While this model performed nearly three times better than the baseline given the training dataset, it also achieved a higher R2 score of .57823 on the test set. This suggests that LSTM models are able to learn general workload patterns from the Google trace that exist in the Alibaba data as well.

Chart, waterfall chart

Description automatically generated

Fig. 10. Percent Difference Compared to Alibaba Lag-1 Baseline

## Figures and Tables

1. Statistics for google trace prediction

| Model | [Portion]-[Measure] | | | |
| --- | --- | --- | --- | --- |
| Training-R2 | Test-R2 | Training-MAE | Test-MAE |
| ConvB32 | .11023 | -.85359 | .04364 | .06914 |
| ConvB64 | .09548 | -.33965 | .0498 | .06869 |
| LSTM64x2 | .82483 | .62465 | .02874 | .0464 |
| LSTM128x2 | .91604 | .63882 | .02128 | .04884 |
| LSTM64Win20 | .68534 | .66666 | .03592 | .04303 |
| Lag1Baseline | .49342 | .48904 | .05223 | .0629 |

1. Statistics for alibaba trace prediction

| Model | [Portion]-[Measure] | | | |
| --- | --- | --- | --- | --- |
| Training-R2 | Test-R2 | Training-MAE | Test-MAE |
| ConvB32 | -2.1243 | -7.5178 | .15337 | .31141 |
| ConvB64 | -1,3846 | -1.7519 | .15502 | .27909 |
| LSTM64x2 | .15129 | .20538 | .13296 | .15887 |
| LSTM128x2 | .16738 | .23879 | .12834 | .16964 |
| LSTM64Win20 | .29471 | .57823 | .11548 | .13676 |
| Lag1Baseline | .11125 | .55208 | .15131 | .16557 |

Chart, histogram

Description automatically generated

Fig. 11. CNN batch=32 Google Predictions

Chart, histogram

Description automatically generated

Fig. 12. CNN batch=64 Google Predictions

Chart, histogram

Description automatically generated

Fig. 13. LSTM-64x2 Google Predictions

A picture containing chart

Description automatically generated

Fig. 14. LSTM-128x2 Google Predictions

Chart, line chart

Description automatically generated

Fig. 15. LSTM-Win20 Alibaba Test Prediction

##### Discussion and future work

**Model Robustness:** The paper shows that some advanced network architectures are capable of performing well on multiple workflows. The LSTM model trained on sequences of length 20 was able to perform better than baseline models specific to each dataset despite only being trained on the Google trace.

**Hyperparameter Optimization:** It is likely that the WaveNet type architecture here can be fit to perform much better on both datasets given its available complexity. Future work could measure the robustness of models more highly optimized to one cluster trace such as the Google dataset. It would be interesting to see whether models more highly optimized on the Google trace performed better or worse on the Alibaba trace than those given semi-arbitrary hyperparameter values.

**Applying Convoluted Features to LSTM:** Future work could involve the implementation of a hybrid Conv-LSTM model where kernel feature maps are created at each time frame and then fed to LSTM cells.

##### Conclusion

This paper presents a study into the advantages of using recently presented deep models as automated trading agents exchanging cryptocurrencies and USD. Two state-of-the-art model architectures were created, one using a ConvLSTM architecture, and the other with a reinforcement learning Deep-Q Network. The LSTM was chosen as most recent successful workload prediction models achieve high prediction accuracy with it. The Deep-Q Network was inspired by Google Deepmind’s recent results of using such a network to play Atari 2600 games at a super-human level using reinforcement learning.

##### References

1. J. H. Hayden, *RSI: the complete guide*. Greenville, SC: Traders Press, 2004.
2. Sepp Hochreiter and J¨urgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 1997.
3. “Feature extraction using convolution,” *Feature extraction using convolution - Ufldl*. [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/Feature\_extraction\_using\_convolution. [Accessed: 01-Dec-2020].
4. “How to Utilize the Ichimoku Cloud Trading Strategy,” *Timothy Sykes*, 22-Oct-2020. [Online]. Available: https://www.timothysykes.com/blog/ichimoku-cloud/. [Accessed: 02-Dec-2020].
5. V. Sim, “Using Deep Learning to Create a Stock Trading Bot,” *Medium*, 27-Aug-2020. [Online]. Available: https://medium.com/analytics-vidhya/using-deep-learning-to-create-a-stock-trading-bot-a96e6351d31c*.* [Accessed: 01-Dec-2020].
6. A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio,” arXiv.org, 19-Sep-2016. [Online]. Available*:* https://arxiv.org/abs/1609.03499v2. [Accessed: 08-Dec-2020].
7. Matiisen, Tambet (December 19, 2015). ["Demystifying Deep Reinforcement Learning"](http://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/). *neuro.cs.ut.ee*. Computational Neuroscience Lab. Retrieved 2018-04-06.
8. Ankit ChoudharyIIT Bombay Graduate with a Masters and Bachelors in Electrical Engineering.I have previously worked as a lead decision scientist for Indian National Congress deploying statistical models (Segmentation, “Deep Q-Learning: An Introduction To Deep Reinforcement Learning,” *Analytics Vidhya*, 27-Apr-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/. [Accessed: 01-Dec-2020].
9. Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph WaveNet for Deep Spatial-Temporal Graph Modeling,” *arXiv.org*, 31-May-2019. [Online]. Available: https://arxiv.org/abs/1906.00121. [Accessed: 09-Dec-2020].
10. Hubel, D. H.; Wiesel, T. N. (1968-03-01). "Receptive fields and functional architecture of monkey striate cortex". The Journal of Physiology. 195 (1): 215–243. doi:10.1113/jphysiol.1968.sp008455. ISSN 0022-3751. PMC 1557912. PMID 4966457.
11. Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph WaveNet for Deep Spatial-Temporal Graph Modeling,” *arXiv.org*, 31-May-2019. [Online]. Available: https://arxiv.org/abs/1906.00121. [Accessed: 09-Dec-2020].

**.**