CS230

Noise reduction for LSTM using Wavelet Transform and Singular Spectrum Analysis

Matthew W. Thomas
Institute for Computational and Mathematical Engineering
Stanford University
mwthomas@stanford.edu

Abstract

1 Introduction

Time series prediction has always been a problem of great interest to the CS community, partly due to the practical applications but also due to the inherent complexities of time-dependent data. There are many traditional methods to analyse time-series data however they have always been limited by the pre-processing and understanding required to implement them (they require a deep understanding of the characteristics of the specific time dependence). RNN as created by [RHW86] offered a way to simply represent time series dependence in a Neural Network architecture. However, RNN's have difficulty retaining the context of information as the gap separating the context information and the application location increase as detailed in [Col21], fortunately [Sch97] solved this problem by introducing the LSTM variant. These cells had the ability to learn the long-term dependencies that RNN cells had difficulty learning. These have become the default method for time-series analysis using deep networks as they have good predictive ability across a multitude of use cases.

High frequency financial data suffers from non-stationary and non-linear characteristics, traditional statistical methods have difficulty predicting these characteristics with consistency. To counter this, methods such as the Wavelet Transform and Singular Spectrum Analysis have been developed, they decompose the data, then recompose the time-series with the goal of increasing analytic methods predictive ability.

Having reduced the data to the section of data with predictive value I apply a simple LSTM model to the de-noised data. It outputs a sequence of predictions spanning 30 min to 1 day ahead, [Bro19]. I will then analyse the prediction error according to multiple metrics at 1hr, 3hr and 6hr intervals, this method was used in [Tan+21] to analyse the same noise reduction methods on less noisy data.

2 Related work

The main inspiration behind this project is [Tan+21], who performed a similar analysis on the DJIA. They found encouraging results using de-noising data on financial time-series, with the pre-processed data resulting in an LSTM model with around a 50% improvement in predictive ability over a simple LSTM with dropout. There have been significant other works suggesting the possibility of improvements to LSTM using such methods such as [LY21] and [MS20].

There are several competing methodologies, even with the field of sequential deep-learning models, with[MM21] using deep-learning to pre-analyse the data, to then re-analyse using traditional statistical techniques. They have found that these methodologies outperform the exclusively statistical

CS230: Deep Learning, Autumn 2021, Stanford University, CA.

approaches, while maintaining their explicability. However, since these methods are outside the purview of this paper I will explore the work no further.

Another interesting field of study is in LSTM models applied to error analysis of primary statistical models, this approach (used by [WL19] to predict hydrological data) uses the LSTM analysis to predict the errors realised by the first model, these predictions are then used to correct the predictions from the original model. This method is interesting as it suggests that the deep LSTM model is learning knowledge that is not statistically represented in the original data (as it is not predictable using statistical techniques).

Returning to financial time series, the current state of the art is running hybrid models which combines multiple approaches, these have consistently been shown to result in the best performance. These methods, such as the [Tan+21] approach have shown that while deep-learning and LSTM models are excellent at perceiving patterns in the data, in the non-stationary complex cases such as financial time-series, they fail to perform the necessary de-noising methods, as such using other methodologies to extract more meaningful information, then analysing it using a deep approach is currently providing the best performance.

3 Data-set and Features

My data set was extracted from bitcoin pricing over the past ten years, it was then complied and released to the community by [Zie21]. It contained pricing data at 1 minute resolution for nearly 10 years. For computational reason I have reduced the resolution to 30 min by averaging over 30 observations. Since it is time-series prediction I have used a single feature however, for inputting the data into the LSTM model I have split the data so that every observation references the relevant past observations and the relevant future prediction targets.

For the later models in my project I will be performing normalization methods on my data and measuring their effect on the performance. I will be using a time-based normalization method as a baseline, with each period normalised to with its individual mean and average. I will then proceed to use Wavelet transforms to analyse the dynamic signals inherent in financial data. I will be using Discrete Wavelet Transforms to analyse my data (I will delve into the theory in the later sections). I will then perform Singular Spectrum Analysis on my data, this will be used to isolate the Trend and Fluctuations, which will then be reconstructed and inputted into the LSTM model.

Thus, using various data pre-processing measures I will consider how different methods for data de-noising/feature extraction perform when fed through an LSTM model.

4 Methods

LSTM Long Short Term Memory networks are a subset of RNN which specialise in learning long-term dependencies. They were defined by [Sch97] in 1997 and have shown themselves very capable of solving a large set of problems. In a RNN model we have a repeating chain of modules which processes each sequence passed as a data point. In an LSTM model the modules is considerably more complex than in an RNN, with four interacting layers.

An LSTM module contains several key components, the main component is the cell state, which runs the entire length of the chain with limit interactions. The LSTM can add/remove information from the chain using gates, i.e. optionally let information through. There are four stages to a LSTM module:

1. Forget Gate Layer: This is a sigmoid layer which looks at the previous prediction and the actual value, deciding how much of the previous information should be kept.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f$$

2. Input Layer Gate: This is a sigmoid layer which decides which values we will update, combined with a tanh layer which creates a vector of new candidates.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C$$

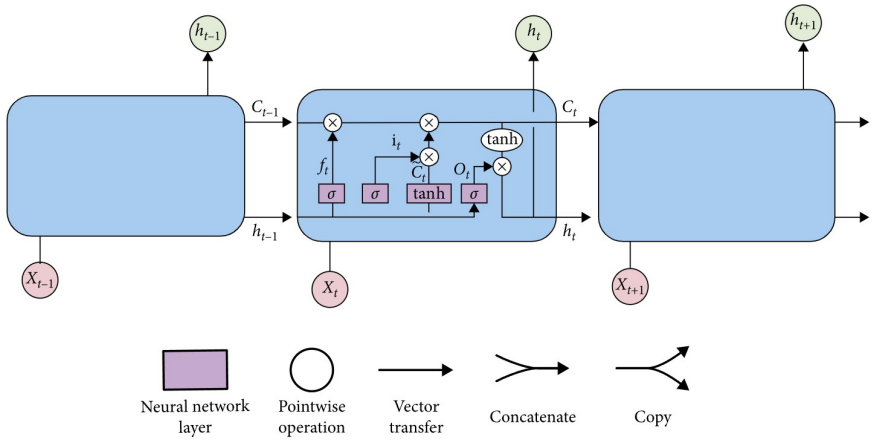

Figure 1: LSTM Cell

3. Cell State Update: Using a pointwise multiplication we update the cell values using i_t as the proportion of new infomation to include.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

4. Output Gate: This is a sigmoid gate to decide how which parts of the cell state we will output, followed by a tanh layers to normalize the cell state. These two elements multiplied together result in selectively outputing our cell state in the right scale for prediction.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

The majority of the understanding/equations expressed above are derived from [Col21] and [Sch97].

Using the equations described above, when strung into an LSTM layer with the specified number of neurons results in a neural network which can selectively remember information through an arbitrary number of modules.

Wavelet Transform The Wavelet transform is an extension of the Fourier transform (designed to isolate the periodicity of a signal). Fourier transforms and by extension wavelet transforms are used to decompose signals into a sum of simple signals. The Fourier transform is extremely effective when the frequencies do not vary in time, clearly an assumption that will not always be satisfied when considering financial time-series data. To overcome this limitation Wavelets were developed, these are versions of the Fourier transform where the mother function is finite in time (i.e. the considered function is bounded in its effect).

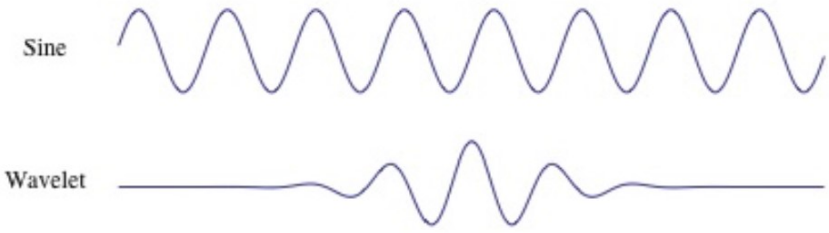

Figure 2: Wavelet vs Fourier frequency range

This allows us to localise Wavelets in different time periods and thus allow for changes in frequency over time.

For a continuous signal we can define a Wavelet as described in [21]:

$$S(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(t)\bar{\phi}(\frac{t-b}{a})dt$$

To translate this to the discrete Wavelet we use exclusively discrete values for the scale and translation factors, a & b. To implement a discrete Wavelet transform we implement it as a filter bank, with each layer isolating and removing the smallest scale frequency.

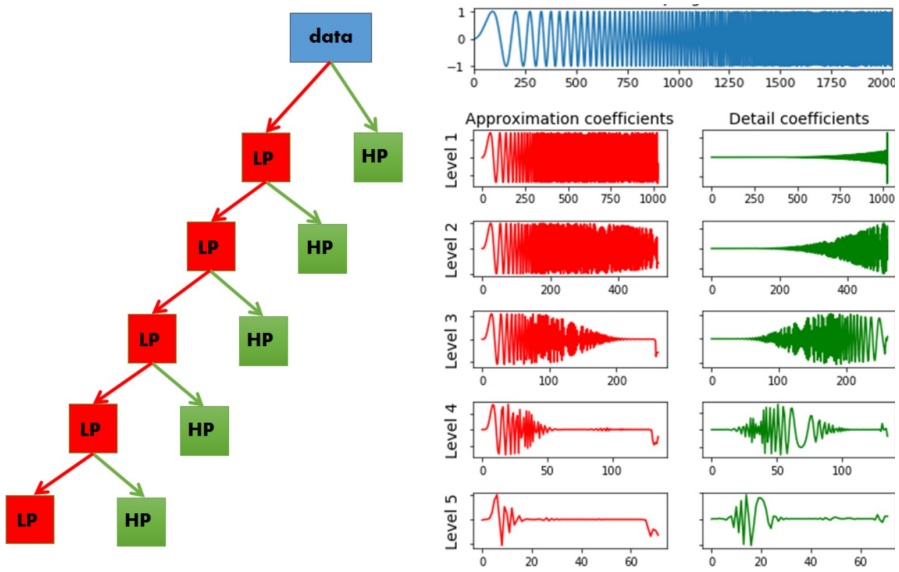

Figure 3: Wavelet Filter Bank implementation

There are many possible Wavelets that could be used to transform the time-series, in this paper I will by using a Sym wavelet. This Wavelet is an symmetric orthogonal function of a db wavelet (i.e. better symmetry for the LSTM model). Using a Sym wavelet I decompose the time-series into 4 detail components and a low-frequency layer, then use the low-frequency layer to recompose my time-series data. This allow me to eliminate the high-frequency noise from my data.

The reconstructed data will then be passed through the LSTM model utilised in the above section.

Singular Spectrum Analysis Singular Spectrum Analysis functions by decomposing the time-series into its component signals, it then reconstructs the time-series based upon certain signals with high singular values. To do this we represent our time-series as a time-delay matrix, i.e. each row represents an observation through the end of the embedded dimension. We then take the covariance matrix of this time-delay matrix and decompose this into its SVD and find the covariance matrix's singular values as explained in [jda18].

Consider X as the time-delay matrix, with S as the covariance matrix of X. we then decompose S to find $\delta_i (i = 1, 2, ..., m)$. We can then extract the points with high singular values and use these as useful signals, discarding the elements with low singular values.

In the table above you can see that the δ_1 contains the 99.98% of the information and as such it will be used to reconstruct the time-series.

The reconstructed data will then be passed through the LSTM model developed in the LSTM section. [1]

[1]For the code related to Singular Spectrum Analysis I am indebted to [kie21] for his excellent package PYMSSA

i	Singular Values	Variance Explained
1	7445616.7	0.999885
2	62139.3	0.000070
3	33056.5	0.000020
4	22686.1	0.000009
5	17001.1	0.000005
6	13328.3	0.000003
7	10779.5	0.000002
8	8980.4	0.000001
9	7780.6	0.000001
10	6787.4	0.000001
11	5970.1	0.000001
12	5230.5	0.000000
13	4644.7	0.000000
14	4412.3	0.000000
15	4148.5	0.000000

Table 1: Singular Spectrum Decomposition

5 Results

For the majority of my hyper-parameters I have used the default values within TensorFlow, there are two reasons for this:

1. The absolute accuracy of my methods is not the target value, I am more interested in the comparative accuracy of the de-noising methods.
2. The default values for the Adam optimizer have been rigorously tested across domains and provide solid results.

To analyse the correct base model I ran a simple LSTM model as described in the LSTM section. However, while experimenting with this original model I noticed that the model experienced significant improvements if the dropout parameter was increased, i.e. losing more of the information. This implied that my original model was over-fitting my data significantly. This can be seen from the below table. To account for this I have results for the optimal dropout parameter (0.3) and the original model.

Period	RMSE	MAE	MAPE	SDAPE
1 Hr ahead	1297.43	967.6715	10.670044	10.664136
3 Hrs ahead	1490.81	1134.2731	12.510202	12.507472
6 Hrs ahead	1760.74	1404.0759	15.681024	15.680698
9 Hrs ahead	1808.52	1439.8011	16.065358	16.064216

Table 2: LSTM

As you can see the original model has significant errors, in the range of 10%-15% over the range of predictions. In the table below you can see the improvement that is garnered by introducing and optimising a dropout parameter.

Period	RMSE	MAE	MAPE	SDAPE
1 Hr ahead	566.35	384.8798	4.207800	4.187588
3 Hrs ahead	620.45	406.3204	4.393708	4.374286
6 Hrs ahead	1251.27	932.2986	10.289185	10.281942
9 Hrs ahead	1329.25	987.6834	10.857811	10.850313

Table 3: LSTM with 0.3 Dropout

The improvements seen through the introduction of a dropout parameter most significantly affect the short term predictions, however they do result in improvements looking up to nine hours ahead.

From the above tables you can see that both Wavelet and SSA preprocessing result in similar results. Interestingly both methods result in better long term predictions while harming the one hour ahead

Period	RMSE	MAE	MAPE	SDAPE
1 Hr ahead	875.76	618.6862	7.055658	7.056260
3 Hrs ahead	578.69	390.3572	4.362071	4.359473
6 Hrs ahead	1021.77	731.0919	8.137015	8.143330
9 Hrs ahead	1103.99	786.9672	8.730211	8.736698

Table 4: LSTM with Wavelet Preprocessing

Period	RMSE	MAE	MAPE	SDAPE
1 Hr ahead	864.37	610.4713	6.970908	6.975339
3 Hrs ahead	562.70	378.7778	4.237757	4.239022
6 Hrs ahead	1013.43	724.0047	8.081805	8.092020
9 Hrs ahead	1095.28	781.1367	8.692516	8.702932

Table 5: LSTM with SSA Preprocessing

prediction quite noticeably. The one hour error rises from around 4% to around 7% against the LSTM with dropout, whereas the nine hour error drops from roughly 11% to 8%.

Overall, the preprocessing methods have, on balance, improved the predictive accuracy of LSTM models on high-noise financial time-series.

6 Conclusion/Future Work

In conclusion, I have found that the de-noising methods both improved and harmed the predictive accuracy of LSTM models depending on the time period considered. This discrepancy could be due to many factors however, I believe the most significant can be explained as variance ratio of noise to signal as a predictive factor over time. In the shorter time periods the LSTM model is predicting the noise parameters, i.e. focusing on the extremely short term variations, and whereas the SSA and Wavelet models do not have the information in required to do this. However, this results in a better predictive accuracy when this information is no longer relevant, which appears to be over time periods longer that 3 hours.

These results are consistent with the results found in [Tan+21], however their results show significantly greater improvements. This may be due to the higher resolution of their data or due to the lack of noise in the DJIA index data.

There are many extensions which could be performed on this work (given more time/computational resources). Ideally I would like to run the same set of models on data with much high resolution. Due to the memory requirement of storing and operating on such high resolution data (while it would have been possible to compute these models using a remote system and a data loader the time constraint prevented me from doing so). I believe that under these circumstances the de-noising methods would prove even more successful, as the standard/dropout LSTM models would have to contend with much higher levels of noise while the preprocessing methods would efficiently handle it.

It may also be interesting to perform a similar analysis with a sequence to sequence fitting method. In the current methodology I fitted the data by targeting one time increment ahead, and then extrapolating the predictions. I would be very interested to see how the systems performed once it has been fitted to a sequence target, i.e. with a loss function including the future predictive errors. In this case the system should be able to improve its predictive accuracy over the long time-horizons, as it will be optimizing for this error, as opposed to exclusively focusing on the short-term movements.

A further interesting extension could be performed by incorporating Volume data into the predictive model, this should increase predictive validity as it gives the model more information to learn from. Using multi-variate time series to predict prices can be done simply as in [BD20], however it may or may not be beneficial to also de-noise the volume data, as I have done for the pricing data.

References

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors - Nature". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.

[Sch97] Sepp Hochreiter; Jurgen Schmidhuber. "LONG SHORT-TERM MEMORY". In: *Neural Computation* (1997).

[jda18] jdarcy. "Introducing SSA for Time Series Decomposition". In: *Kaggle* (Mar. 2018). URL: https://www.kaggle.com/jdarcy/introducing-ssa-for-time-series-decomposition.

[Bro19] Jason Brownlee. *Making Predictions with Sequence*. 2019. URL: https://machinelearningmastery.com/sequence-prediction/. (accessed: 10.02.2021).

[WL19] Z. Wang and Y. Lou. "Hydrological time series forecast model based on wavelet denoising and ARIMA-LSTM". In: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (Mar. 2019), pp. 1697–1701. DOI: 10.1109/ITNEC.2019.8729441.

[BD20] Samit Bhanja and Abhishek Das. "Deep Neural Network for Multivariate Time-Series Forecasting". In: *Proceedings of International Conference on Frontiers in Computing and Systems*. Singapore: Springer, Nov. 2020, pp. 267–277. ISBN: 978-981-15-7833-5. DOI: 10.1007/978-981-15-7834-2_25.

[MS20] Sidra Mehtab and Jaydip Sen. "Analysis and Forecasting of Financial Time Series Using CNN and LSTM-Based Deep Learning Models". In: *ResearchGate* (Dec. 2020). URL: https://www.researchgate.net/publication/346647852_Analysis_and_Forecasting_of_Financial_Time_Series_Using_CNN_and_LSTM-Based_Deep_Learning_Models.

[21] *A guide for using the Wavelet Transform in Machine Learning*. [Online; accessed 5. Nov. 2021]. Aug. 2021. URL: https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning.

[Col21] Colah. *Understanding LSTM Networks – colah's blog*. [Online; accessed 29. Oct. 2021]. Oct. 2021. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs.

[kie21] kieferk. *pymssa*. [Online; accessed 18. Nov. 2021]. Nov. 2021. URL: https://github.com/kieferk/pymssa.

[LY21] Wuwei Liu and Jingdong Yan. "Financial Time Series Image Algorithm Based on Wavelet Analysis and Data Fusion". In: *J. Sens.* 2021 (Mar. 2021), p. 5577852. ISSN: 1687-725X. DOI: 10.1155/2021/5577852.

[MM21] Angelo Garangau Menezes and Saulo Martiello Mastelini. "MegazordNet: combining statistical and machine learning standpoints for time series forecasting". In: *arXiv* (May 2021). eprint: 2107.01017. URL: https://arxiv.org/abs/2107.01017v1.

[Tan+21] Qi Tang et al. "Prediction of Financial Time Series Based on LSTM Using Wavelet Transform and Singular Spectrum Analysis". In: *Math. Prob. Eng.* 2021 (May 2021), p. 9942410. ISSN: 1024-123X. DOI: 10.1155/2021/9942410.

[Zie21] Zielak. *Zielak's Bitcoin Historical Data w/o NaN*. [Online; accessed 29. Oct. 2021]. Oct. 2021. URL: https://www.kaggle.com/kognitron/zielaks-bitcoin-historical-data-wo-nan.