

Team 10 - Conditional Time Series Forecasting with Convolutional Neural Networks

September 20, 2020

Antony Bogodist, Yawer Ijaz, Naveen Gupta, Antoine Pingel, Love Rawat, Eric Tu


GitHub Repository: <https://github.com/yawerijaz-mfe/mfe23ot2>

ABSTRACT

This paper presents a critique of the multivariate time series forecasting performance of an adapted deep convolutional WaveNet architecture, based on the research article “Conditional Time Series Forecasting with Convolutional Neural Networks” (Borovykh et al. 2018). We determined the paper overstates WaveNet architecture results and understates the performance of LSTM models. In particular, the LSTM models referenced in the paper for benchmarking had no hyperparameter tuning. We propose modified architectures as the ones presented in the paper and find that LSTM models had more predictable and superior performance to the modified WaveNet architecture across all tasks, as well as more consistent results. We found WaveNet architectures to be very sensitive to weight initialization, choice of activation function, and the presence of skip connections. Overall, the reduction in training time was minimal for the WaveNet for these simple models, but in all other aspects, the LSTM implementation outperformed consistently and showed more consistent results to changes in architecture and hyperparameter tuning.

INTRODUCTION

Many deep learning approaches towards time series analysis techniques have been made possible thanks to the increased availability of computing power. As financial time series data has become more intertwined with a greater set of economic and execution factors, machine learning and deep learning techniques offer a way to fit more and more complex models towards the goal of separating signals from noise. Deep learning in particular shows promise because of the complexity and quantity of potential factors driving the movement of financial time series.



A time series is an ordered sequence, where each point is assumed to be dependent on the previous ones. Recurrent Neural Networks (RNN) have traditionally been used for processing sequential data, as a key feature of the architecture are memory units used to store and forward information received from previous data points. In this manner, cells processing the input at time t will also have as input the output of the RNN at time $t-1$. LSTM (Long Short-Term Memory) are an adaptation of the RNN architecture to allow more persistent information capture due to the addition of dedicated long term memory components. However, due to the success of convolutional neural networks (CNN) in image-related tasks, as well as efficiency improvements in the lower number of parameters and the parallel nature in which CNN filters are trained and executed, an interest towards strategies involving the use of CNN's for sequential data has increased. We look to study a comparison of the advantages and disadvantages of CNNs as compared to RNNs to determine which architecture is best suited to financial time series forecasting.

The goal of this project is to test a recent WaveNet architecture proposed for use in time series forecasting, based on the research article “Conditional Time Series Forecasting with Convolutional Neural Networks” (Borovykh et al. 2018). The WaveNet implementation uses dilated Convolutional Neural Networks (CNN) layers to learn patterns in time series data series and maintain information with longer input horizons as opposed to memory units used within Recurrent Neural Networks (RNN) layers. The proposed WaveNet architecture modified the original WaveNet implementation from its original use case, speech generation, to time series forecasting. The gated activation function was replaced by a ReLU and/or Leaky ReLU to improve efficiency for the noisy non-stationary time series forecasting task. The use case addressed by the paper was conditional forecasting of S&P500 returns on the VIX and 10Y Treasury Yield, as well as to forecast major exchange rates conditional on other exchange rates (inc. EURUSD, GBPJPY...). The benchmark models used for comparison are a traditional VAR and LSTM model, and performance metrics reported as an improvement ratio over a Naïve approach, defined as the result of taking the previous value as the next prediction value. The paper showed results of the WaveNet architecture demonstrating statistically significant performance improvements in exchange regression and up/down movement classification tasks as compared to LSTM and VAR models.

DATA

In this section we present the data used in our models. As originally presented in the paper, we forecasted both:

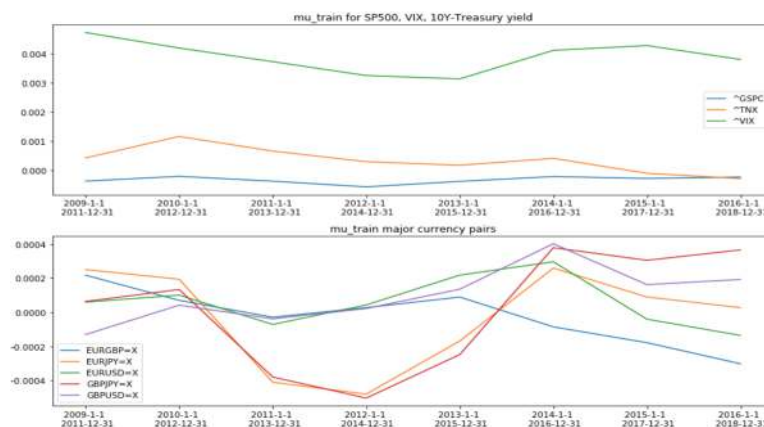
- S&P 500 returns using
 - S&P 500 prices
 - VIX index level
 - 10Y Treasury
 - Yield data

- EURUSD, EURJPY, GBPJPY, EURGBP, GBPUSD

We used a total time period of 13 years, from January 1, 2005 to December 31, 2018. Similarly to what was presented in the paper we used rolling periods split as followed:

- Training: 3 Years rolling periods
- Testing: Following 1 year non-overlapping

All the time series were normalized using the training sets means and standard deviations, again on a rolling basis. We verified all time series are stationary via Augmented Dickey-Fuller tests.



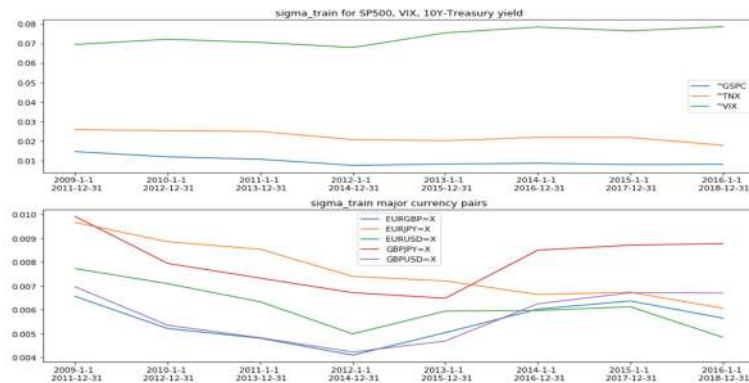


Figure 1: mean and standard deviation of training sets over time

METRICS

Two key performance metrics were proposed by Borovykh et al. The first is Mean Absolute Scaled Error (MASE), which equals the Mean Absolute Error ratio from the model of interest to the Mean Absolute Error using a Naive forecasting method. The Naive forecasting model can be thought of as a proxy of a volatility measurement, as the MAE of such a model would be higher when the previous day's value deviates from today's value, the equivalent of a higher volatility regime. In this sense, MASE can be viewed as an error measurement scaled to the volatility in that region. The second metric is HITS, which is simply the classification accuracy of up and down movements in the time series data. This metric is not scaled to the Naive method.

All metrics were evaluated on a rolling period basis. Daily data was used, therefore training data for each rolling period consisting of approximately 720 data points and 240 validation data points (slight adjustments due to differences in calendar years). A total of 10 rolling periods was analyzed, and the mean of the rolling periods reported, as compared to the original paper by Borovykh et al., whom only chose to present the top three rolling periods with highest WaveNet results.

MODELS AND RESULTS

Vector Autoregression (VAR)

A Vector Autoregression (VAR) was used as a benchmark for financial time series forecasting. Johanson cointegration tests verified the significance of cointegration effects, justifying the use

of VAR. VAR(2) and VAR(3) were found to be the best model for S&P500 and exchange rate regressions respectively. The results of the rolling period regression within each rolling period as their aggregated mean and standard deviation are shown in Figure 2.

	eurgbp	eurjpy	eurusd	gbpjpy	gbpusd		MASE	HITS
2012	0.441872	0.585727	0.507655	0.466181	0.404603	2012	0.387946	0.557940
2013	0.617121	0.675079	0.525497	0.663481	0.643530	2013	0.431177	0.572034
2014	0.523785	0.380064	0.426666	0.516418	0.481577	2014	0.488338	0.638655
2015	1.099829	0.680247	1.058590	0.646732	0.876024	2015	0.883152	0.543568
2016	0.985781	0.732828	0.638004	1.220883	1.079770	2016	0.660396	0.491597
2017	0.655825	0.599027	0.587448	0.654530	0.715007	2017	0.344458	0.446809
2018	0.436642	0.544123	0.548690	0.569268	0.630317	2018	0.911653	0.555085
2019	0.591908	0.482419	0.438451	0.538517	0.582099	2019	0.723167	0.602804
mean	0.669095	0.584939	0.591375	0.659501	0.676616	mean	0.603786	0.551061
std	0.229520	0.108068	0.188469	0.222480	0.202477	std	0.219581	0.056252

Figure 2: VAR performance for exchange rates in MASE (left) and for S&P 500 prediction in MASE and HITS (right)

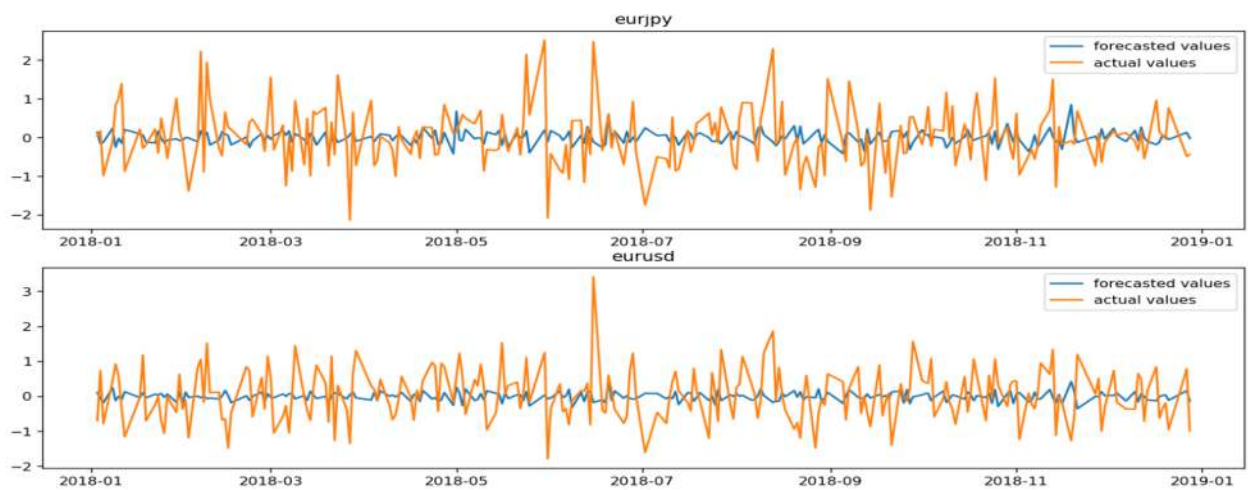


Figure 3: A depiction of the forecasted VAR values (blue) and actual values (orange) of exchange rates.

The results of the VAR show relatively large differences in performance between rolling periods, despite recalibrating on the previous 3 years of data. There is a clear improvement over a Naive forecasting method, and thus verifies the usefulness as a benchmark performance to evaluate the LSTM and WaveNet models.

LSTM

Long Short Term Memory networks are an extension of the RNN architecture, with the added feature of specific long-term memory units to improve recall over longer sequences. This comes at a cost of increasing the number of trainable parameters, extending training time and the potential for overfitting. An overview of the LSTM architecture is shown in Figure 5. We apply the LSTM architecture on the conditional prediction problem, providing as input a multivariable time series for prediction, and tune hyperparameters to optimize performance on the dataset.

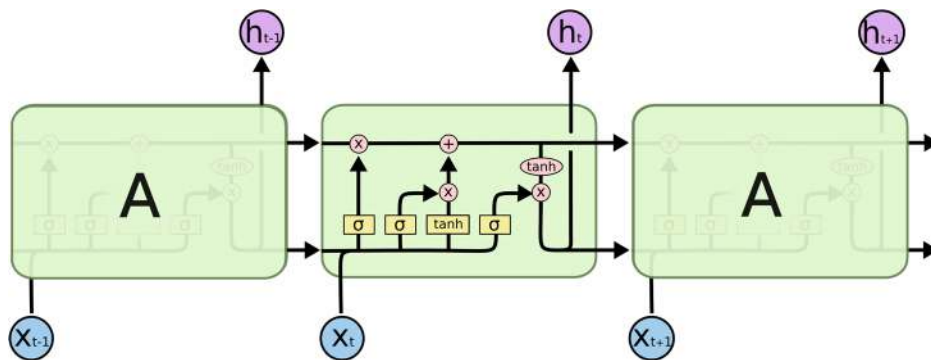


Figure 5: A schematic diagram showing the repeating module in an LSTM contains four interacting layers.

Hyperparameters and architecture:

The LSTM model consists of two LSTM layers with 5 LSTM cells and fully connected dense layer for the S&P 500 regression task. The receptive field is of 16 inputs, to match the WaveNet implementation. We tested additional architectures using 1 and 256 input sizes, which led to inconsistent and overall less performant regression. Hyperparameter tuning in line with

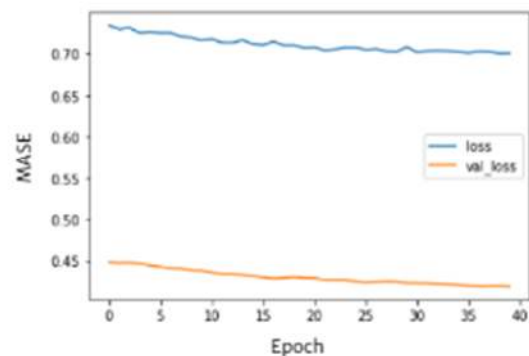
simplification of the architecture appeared to lead to the highest performance. In particular, regularization through dropout layers of value 0.5 smoothed validation loss behavior and l1 regularization of value .001 significantly improved training time. Reducing the adam optimizer learning rate to .001 also led to smoother training behavior (smoother loss function). Lowering the number of LSTM cells and increasing the number of layers led to the most improvement in loss metrics. For the classification task, the addition of an additional layer improved performance. We added a third LSTM layer with 25 cells to additionally test the impact on training time. Lastly, the exchange rate regression model increased all LSTM cell counts to 25; the added model complexity increased performance, but we lowered the number of layers to 2 to improve training time.

Results:

The LSTM implementation showed higher MASE and lower HITS, but more consistent performance across rolling periods with significantly lower standard deviation than VAR. Since the performance is consistent across rolling periods, it seems to imply the LSTM model is performing consistently on a volatility adjusted metric, since Naive forecasting can be viewed as a measure of volatility.

S&P 500 Regression and Classification

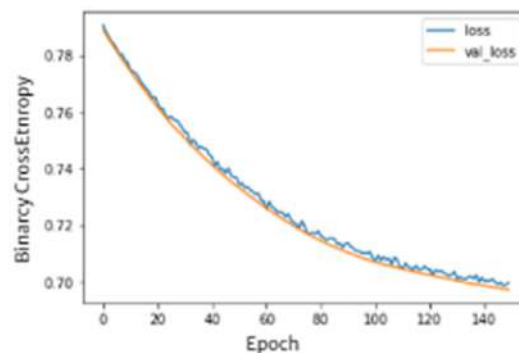
	Naive(Train)	Naive(Val)	MASE-Train(Std)	MASE-Val(Std)
2012	1.048	0.614	0.66 (0.67)	0.67 (0.61)
2013	1.049	0.673	0.66 (0.69)	0.67 (0.55)
2014	1.034	0.749	0.67 (0.71)	0.66 (0.63)
2015	1.123	1.371	0.66 (0.60)	0.71 (0.65)
2016	1.082	1.094	0.68 (0.63)	0.66 (0.67)
2017	1.054	0.521	0.68 (0.66)	0.65 (0.67)
2018	0.994	1.247	0.67 (0.73)	0.73 (0.76)
2019	0.923	1.030	0.69 (0.79)	0.67 (0.69)
Mean	1.038	0.912	0.671	0.678
Std	0.056	0.295	0.011	0.026



Training Time: 3 min 5 seconds

Figure 6: Comparison of LSTM MASE metrics on S&P500 data with Naive forecasts (left) and an example of the training behavior in one rolling period (right)

	Naive(Train)	Naive(Val)	HITS(Train)	HITS(Val)
2012	0.494	0.526	0.533	0.483
2013	0.509	0.422	0.530	0.540
2014	0.492	0.452	0.519	0.531
2015	0.471	0.459	0.538	0.550
2016	0.452	0.456	0.548	0.531
2017	0.458	0.475	0.503	0.559
2018	0.459	0.519	0.510	0.515
2019	0.485	0.521	0.524	0.595
Mean	0.477	0.479	0.525	0.538
Std	0.019	0.036	0.014	0.031



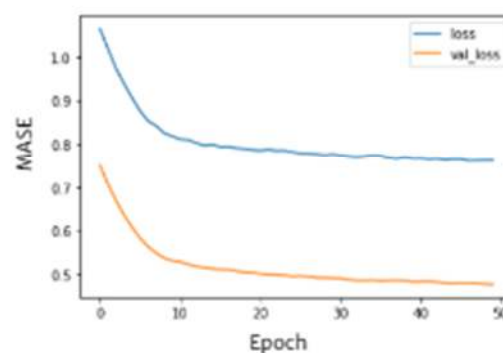
Training Time: 12 minutes 16 seconds

Figure 7: Comparison of LSTM HITS metrics on S&P 500 data with Naive forecasts (left) and an example of the training behavior in one rolling period (right)

Exchange Rate Classification

GBPUSD=X

	Naive(Train)	Naive(Val)	MASE-Train(Std)	MASE-Val(Std)
2012	1.10	0.62	0.69 (0.55)	0.72 (0.58)
2013	1.14	1.10	0.68 (0.55)	0.65 (0.54)
2014	1.15	0.79	0.67 (0.55)	0.69 (0.66)
2015	1.11	1.37	0.67 (0.60)	0.69 (0.59)
2016	1.11	1.74	0.67 (0.60)	0.70 (0.82)
2017	0.95	0.98	0.67 (0.79)	0.67 (0.62)
2018	1.00	0.88	0.65 (0.74)	0.70 (0.55)
2019	1.01	0.78	0.68 (0.73)	0.77 (0.70)
Mean	1.07	1.03	0.67	0.7
Std	0.07	0.34	0.01	0.03



EURJPY=X

	Naive(Train)	Naive(Val)	MASE-Train(Std)	MASE-Val(Std)
2012	1.10	0.92	0.68 (0.59)	0.70 (0.53)
2013	1.11	1.12	0.67 (0.59)	0.67 (0.63)
2014	1.13	0.60	0.66 (0.59)	0.69 (0.65)
2015	1.08	1.05	0.66 (0.64)	0.71 (0.66)
2016	1.02	1.08	0.67 (0.67)	0.67 (0.69)
2017	1.04	0.91	0.67 (0.69)	0.67 (0.55)
2018	1.06	0.86	0.66 (0.63)	0.67 (0.60)
2019	1.09	0.71	0.65 (0.64)	0.72 (0.75)
Mean	1.08	0.91	0.66	0.69
Std	0.03	0.17	0.01	0.02

EURUSD=X

	Naive(Train)	Naive(Val)	MASE-Train(Std)	MASE-Val(Std)
2012	1.10	0.82	0.70 (0.56)	0.69 (0.54)
2013	1.11	0.88	0.70 (0.58)	0.63 (0.56)
2014	1.12	0.71	0.67 (0.58)	0.66 (0.63)
2015	1.13	1.70	0.65 (0.58)	0.70 (0.60)
2016	1.08	0.99	0.66 (0.64)	0.67 (0.62)
2017	1.07	0.95	0.68 (0.66)	0.66 (0.52)
2018	1.08	0.84	0.67 (0.61)	0.69 (0.52)
2019	1.15	0.70	0.67 (0.55)	0.73 (0.64)
Mean	1.10	0.95	0.68	0.68
Std	0.03	0.30	0.02	0.03

GBPJPY=X

	Naive(Train)	Naive(Val)	MASE-Train(Std)	MASE-Val(Std)
2012	1.07	0.73	0.67 (0.61)	0.70 (0.59)
2013	1.08	1.04	0.67 (0.62)	0.70 (0.64)
2014	1.10	0.78	0.68 (0.61)	0.70 (0.70)
2015	1.06	0.91	0.68 (0.65)	0.75 (0.65)
2016	1.01	1.80	0.70 (0.67)	0.69 (0.80)
2017	0.91	0.87	0.70 (0.84)	0.67 (0.58)
2018	0.95	0.82	0.68 (0.79)	0.66 (0.53)
2019	0.99	0.73	0.67 (0.75)	0.72 (0.79)
Mean	1.02	0.96	0.68	0.7
Std	0.06	0.33	0.01	0.03

GBPUSD=X

	Naive(Train)	Naive(Val)	MASE-Train(Std)	MASE-Val(Std)
2012	1.10	0.62	0.69 (0.55)	0.72 (0.58)
2013	1.14	1.10	0.68 (0.55)	0.65 (0.54)
2014	1.15	0.79	0.67 (0.55)	0.69 (0.66)
2015	1.11	1.37	0.67 (0.60)	0.69 (0.59)
2016	1.11	1.74	0.67 (0.60)	0.70 (0.82)
2017	0.95	0.98	0.67 (0.79)	0.67 (0.62)
2018	1.00	0.88	0.65 (0.74)	0.70 (0.55)
2019	1.01	0.78	0.68 (0.73)	0.77 (0.70)
Mean	1.07	1.03	0.67	0.7
Std	0.07	0.34	0.01	0.03

Training Time: 4 minutes 52 seconds each

Figure 8: Comparison of LSTM MASE metrics for exchange rate forecasting with Naive forecasts (left) and an example of the training behavior in one rolling period (right)

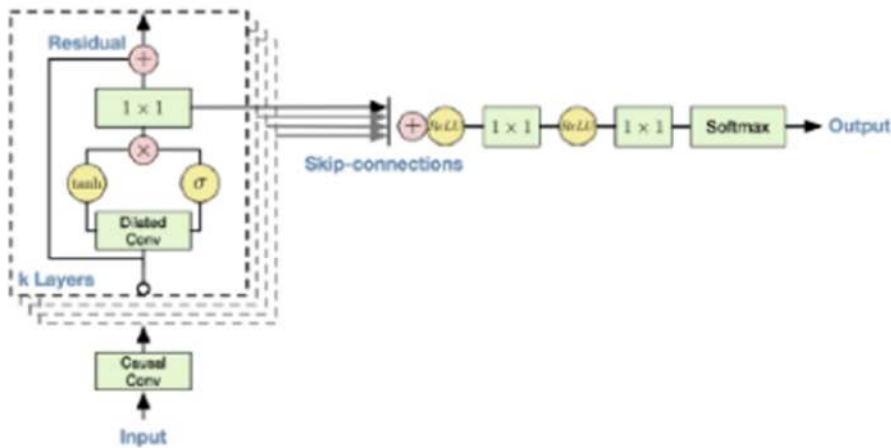
Potential Problems:

One issue seen during training was validation loss often being consistently lower than training loss. Our initial hypothesis was the introduction of regularization may have caused higher training loss values (since regularization effects are not used for validation); however, removing regularization yielded the same results. In addition, this observation is seen in the WaveNet implementation as well, leading us to believe it is an inherent aspect of the data.

Wavenet Model

Description: Wavenet is a powerful new predictive technique that uses many Deep Learning algorithms from Computer Vision and Audio Signal Processing models and applies them to longitudinal (time series data). It was created by researchers at London-based artificial intelligence firm DeepMind.

One of the most useful features of deep neural network architectures is their capability to yield good results in very different domains. Image recognition, natural language processing, or regression tasks can be implemented with similar solutions, model parts, or models. To build a state of the art model in any domain it is important to know what are the new techniques of other fields.



Overview of the residual block and the entire architecture.

Unconditional WaveNet

Hyperparameters and architecture:

The model consists of 4 modified WaveNet blocks followed by 1 Convolutional Layer and 1 Dense Layer. The receptive field is of 64 inputs and has thus been extended as compared with the original model in order to capture longer dependencies. We use one filter with a size of 2 as presented in the original paper.

We chose to use the Adam optimizer as in the original paper, and trained our model for 150 epochs as the modern tends to stop learning at this point. We used a learning rate of 0.001. We

also chose to use an L2 regularizer.

Results:

We fit SP500 and EURUSD into WaveNet unconditionally. Using MASE and HITS metrics, we likewise ran the model through rolling windows. We observed on both measures, the unconditional WaveNet outperformed the naive benchmark. The MASE on the validation set is reduced by 20% and 25% for both instruments. Of the 8 years we studied, WaveNet underperformed Naive benchmark 3 times, and EURUSD once. This may be due to the more volatile nature of the stock index. While the relative performance as measured by HITS on the two time series was more even on both instruments, neither exhibited noticeable improvement upon LSTM.

	Naive(train)	Naive(val)	MASE (Train)	MASE (Val)
2012	1.039775	0.636707	0.641632	0.685726
2013	1.046328	0.699716	0.698880	0.715448
2014	1.043066	0.712625	0.669142	0.686473
2015	1.124616	1.355250	0.704342	0.809976
2016	1.085428	0.921592	0.677339	0.686705
2017	1.046045	0.528632	0.676285	0.648999
2018	0.986519	1.214588	0.689123	0.774090
2019	0.902741	1.000863	0.689069	0.663966
mean	1.034315	0.883747	0.680726	0.708923
std	0.061999	0.273223	0.018443	0.052020

	Naive(train)	Naive(val)	HITS (Train)	HITS (Val)
2012	0.492754	0.544554	0.442029	0.435644
2013	0.513158	0.434146	0.498538	0.473171
2014	0.491924	0.468599	0.511013	0.526570
2015	0.471366	0.466667	0.447871	0.433333
2016	0.454282	0.458937	0.486212	0.560386
2017	0.460870	0.480392	0.436232	0.485294
2018	0.463557	0.512195	0.498542	0.492683
2019	0.484581	0.497268	0.524229	0.557377
mean	0.479061	0.482845	0.480583	0.495557
std	0.018677	0.032193	0.031684	0.046182

Tab x: UCWN SP500

Training time: 2min52s

	Naive(train)	Naive(val)	MASE (Train)	MASE (Val)
2012	1.096027	0.804199	0.679886	0.721511
2013	1.107303	0.863413	0.696355	0.628980
2014	1.123224	0.727859	0.675172	0.670152
2015	1.124137	1.662735	0.652918	0.723229
2016	1.077888	0.968299	0.647302	0.673153
2017	1.075094	0.926216	0.672895	0.660332
2018	1.070469	0.834674	0.639822	0.712631
2019	1.141404	0.625492	0.652854	0.744439
mean	1.101943	0.926611	0.664651	0.691803
std	0.024639	0.296062	0.018063	0.036790

	Naive(train)	Naive(val)	HITS (Train)	HITS (Val)
2012	0.491304	0.514851	0.436232	0.460396
2013	0.508772	0.400000	0.516082	0.595122
2014	0.459618	0.429952	0.566814	0.589372
2015	0.453744	0.500000	0.443465	0.476190
2016	0.441219	0.454106	0.461538	0.483092
2017	0.453623	0.446078	0.501449	0.450980
2018	0.466472	0.507317	0.504373	0.439024
2019	0.464023	0.530055	0.497797	0.524590
mean	0.467347	0.472795	0.490969	0.502346
std	0.020669	0.043556	0.040059	0.057160

Tab x: UCWN EURUSD

Training time: 2min36s

Conditional WaveNet

Hyperparameters and architecture:

The model consists of 6 modified conditional WaveNet blocks followed by 1 Convolutional Layer and 1 Dense Layer. The receptive field is of 64 inputs and has thus been extended as compared with the original model in order to capture longer dependencies. We use one filter with a size of 2 as presented in the original paper.

The activation function was changed from a ReLU to a Leaky ReLU as the model had a hard time learning with the simple ReLU. We chose to use the Adam optimizer as in the original paper, and trained our model for 150 epochs as the modern tends to stop learning at this point. We used a learning rate of 0.001. We also chose to use an L2 regularizer.

Results:

As volatility is a defining feature of the stock market, we apply conditional WaveNet on SP500, with the VIX index as an extra input, as suggested by the author of the original paper. For the EURUSD series, we used EURJPY as the conditioning input, as an attempt to capture the interdependence of various currency pairs.

Comparing the results from conditional and unconditional WaveNet, we saw that conditioning provided more benefit to SP500 forecasting. With the addition of VIX condition, the validation MASE was brought down by 2%, and HITS up by 1%. Unfortunately, the same improvement did not occur on the currency series - the converging rate is much slower, and the terminal loss is considerably higher than the unconditional WaveNet and the Naive benchmark. We learned that Conditional WaveNet may be sensitive to the choice of conditioning, and tuning one involved much more effort than Unconditional WaveNet or LSTM.

	Naive(train)	Naive(val)	MASE (Train)	MASE (Val)
2012	1.039775	0.636707	0.642117	0.703567
2013	1.046328	0.699716	0.657574	0.673735
2014	1.043066	0.712625	0.714153	0.714476
2015	1.124616	1.355250	0.669492	0.708557
2016	1.085428	0.921592	0.691197	0.654764
2017	1.046045	0.528632	0.684790	0.646493
2018	0.986519	1.214588	0.675324	0.745127
2019	0.902741	1.000863	0.690482	0.664451
mean	1.034315	0.883747	0.678141	0.688896
std	0.061999	0.273223	0.020819	0.032032

	Naive(train)	Naive(val)	HITS (Train)	HITS (Val)
2012	0.492754	0.544554	0.479710	0.430693
2013	0.513158	0.434146	0.529240	0.546341
2014	0.491924	0.468599	0.508076	0.526570
2015	0.471366	0.466667	0.502203	0.461905
2016	0.454282	0.458937	0.490566	0.512077
2017	0.460870	0.480392	0.501449	0.553922
2018	0.463557	0.512195	0.514577	0.492683
2019	0.484581	0.497268	0.525698	0.573770
mean	0.479061	0.482845	0.506440	0.512245
std	0.018677	0.032193	0.015688	0.045356

Tab x: CWN SP500

Training time: 4min12s

	Naive(train)	Naive(val)	MASE (Train)	MASE (Val)
2012	1.096027	0.804199	0.701527	0.678087
2013	1.107303	0.863413	0.703484	0.618751
2014	1.123224	0.727859	0.646494	0.648084
2015	1.124137	1.662735	1.664555	3.380252
2016	1.077888	0.968299	0.666089	0.668948
2017	1.075094	0.926216	0.729693	0.655010
2018	1.070469	0.834674	0.675005	0.690574
2019	1.141404	0.625492	0.666346	0.747697
mean	1.101943	0.926611	0.806649	1.010925
std	0.024639	0.296062	0.325200	0.896206

	Naive(train)	Naive(val)	HITS (Train)	HITS (Val)
2012	0.491304	0.514851	0.521739	0.490099
2013	0.508772	0.400000	0.529240	0.575610
2014	0.459618	0.429952	0.469897	0.410628
2015	0.453744	0.500000	0.518355	0.533333
2016	0.441219	0.454106	0.522496	0.507246
2017	0.453623	0.446078	0.484058	0.465686
2018	0.466472	0.507317	0.511662	0.517073
2019	0.464023	0.530055	0.515419	0.502732
mean	0.467347	0.472795	0.509108	0.500301
std	0.020669	0.043556	0.019497	0.045334

Tab x: CWN EURUSD

Training time: 4min55s

CONCLUSION

The paper that we based our work on is, to our knowledge, unique on the subject which made it hard to compare results. The paper displayed great results for CWN and UCW benchmarked to a fairly simple LSTM, but in our implementations, we could not reproduce these results with simple models and had to enhance those. We observed very different results for different seeds (weight initialization) and the paper states that they trained their model using different seeds, discarded the results from bad ones, and averaged results from the best results. The predictions across the board seemed to be rather close to 0 (no risk taking from the model). Finally the “enhanced” LSTM showed better results, especially on classification.

The WaveNet showed some interesting features and seducing ideas, but the architecture implemented in the paper felt too simple to capture short-term and long-term non linear dependencies in the data. We were only able to recreate comparable performance in the WaveNet architecture through careful modification of input size, activation function selection, and testing several different weight initialization schemes. In addition, conditioning on different inputs needed a readjustment of parameters to recreate optimum performance. While we found credence towards the argument that LSTM models become more unwieldy to train with more complex models, in our simpler model, the training times are comparable, and the difficulties in reconstructing the neural network architecture and larger fluctuations in performance with different hyperparameters leads us to suggest an LSTM implementation over the WaveNet model. Furthermore, since the performance of LSTMs were found to outperform the WaveNet on all fronts, the LSTM model seems best suited for the task, as well as the easiest to implement quickly.