Forecasting S&P 500 Volatility using Long Short-Term Model (LTSM) Neural Networks

41405 49161 46496

Abstract

This paper investigates a compendium of LSTM models in forecasting accuracy for 30-Day volatility of the S&P500 vis-a-vis GARCH model. The result indicate that simplicity of a plain LSTM is the best although using CNN on polar encodings carries potential as well.

1. Introduction

Volatility is not a directly observable variable yet it is central to decision-making of every asset manager globally whether it is to preserve the purchasing power for pension funds or market speculators. The volatility of the S&P 500 index has a tendency to be mean-reverting, clustering and persistence. Therefore while the culmination of parametric models such as Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) have been able to improve on forecast accuracy on the preceding parametric models, it still fails on multiple occasions as it is unable to capture idiosyncrasies in the market. For instance, effects of non-Gaussian distributed noise, market microstructure, heteroscedasticity, exogenous and asymmetric effect of news combined together with different time scales

Hence we harness the ability for Long-Short Term Memory models to account for non-linear dependencies and having a variable with component of locality of time memorised based on *gates*, we strive to tackle the problem to forecast volatility accurately with this medium in conjunction with advancements seen in contemporary literature. This is especially crucial during structural breaks and regime changes, for instance, higher market volatility emerging from post-COVID 19 macroeconomic conditions. We expect the progress of research to only increase given that under a rising interest rate climate, allocation of capital becomes riskier and thus carrying a higher opportunity cost.

While a plain and simple LSTM model does show superiority over GARCH model, we fail to reliably outperform it with more sophisticated additions. One suspect for this is the depth of the data set itself albeit having longer timeline than other literature work. In summary, we see that at 4.98% MAPE, the simple model reigns over the GARCH at 25.8% and while 2D-CNN-LSTM architecture can reach lower MAPE figures, the estimates tends to be unstable. Therefore, we are able to supplement existing studies with potential for deep learning models to make accurate predictions in financial markets.

2. Related Work

Recurrent Neural Networks (RNN) are prominent in modelling sequential data thus fitting our purpose of prediction from the high-frequency financial time series data. However given their deficiency in the underlying feedback connections being limited to retaining short-term memory due to vanishing gradients problem, (Hochreiter & Schmidhuber, 1997) developed the *Long Short-Term Memory (LSTM)*. It addressed the problems of the contemporary solutions such as *Back-Propagation Through Time*. LSTM are able to capture non-linear dependencies characterising the data.

(Fischer & Krauss, 2017) investigated a vanilla-LSTM setup in predicting out-of-sample directional movements of S&P500, whereby it outperformed memory-free classification outcomes such as logistic classification and deep neural networks. RMSprop was perpetuated as the suited optimiser following previous literature and application of low dropout values of 0.1 ensured superior performance in conjunction with early-stopping to reduce risk of over-fitting. *Patience periods* were employed and the standard 80:20 training to validation proportion were used. This culminated in LSTM's out-performance and preferred forecasting method in terms of accuracy.

(Siami-Namini et al., 2018) align with this finding vis-a-vis ARIMA models with mean reduction in error rates of 84-87% relatively. (Liu et al., 2018) observed that LSTM may only be stacked upto five times prior to significantly increasing computational demand, thus followed a three-layer configuration with 8 features including moving average (MA) and exponential moving average (EMA), achieving 72% accuracy in prediction of stock prices.

(Bao et al., 2017) introduced the structure of employing *stacked autoencoders*, to train the Open, High, Low, Close (OHLC) variables layer-wise while the depth being useful for determining qualities such as invariance and abstraction

of extracted feature. Sunsequently the LSTM was applied after denoising the financial data through *Wavelet Transforms*, forming the *WSAEs-LTSM*. Moreover, the input was augmented with macroeconomic variables of exchange rate and interest rate to fully harness the their influence on stock markets. This advancement succeeded in delivering better performance in predicting stock prices across multiple indices globally and served as an amelioration to (Fischer & Krauss, 2017).

Literature in LSTM for volatility forecasting is still nascent in terms of citations. However, (Vidal & Kristjanpoller, 2020) in the context of forecasting volatility for gold, combined a pre-trained Convolutional Neural Networks, namely a VGG16 network with the LSTM to form the CNN-LTSM approach. This hybrid model reduced the MSE by 18% and 37% in comparison to vanilla-LSTM and vanilla-GARCH models respectively. Time series were transformed into RGB images using *Gramian Angular Fields (GAF)* and *Markov Transition Fields (MTF)* for each time step, yielding three channels. Essentially, a polar encoding of the data was conducted to get resulting angles upon which a Gram Matrix like operation was applied to preserve temporal dependency resulting in a GAF.

After the transformed data is utilised passed through the architecture, the output vector is combined with the parallel-run LSTM's output to provide a forecast. The CNN-LTSM better captured greater movement in higher and lower range than base level during times of stable volatility and also significantly improved forecasts during volatility peaks. It is imperative to note the computational costs being high in implementation as well as the availability of 50-year daily data for gold.

(Liu, 2019) studied simpler single and two-layer LSTM model to forecast Apple Inc and S&P500 stocks while involving regularisation techniques akin to (Fischer & Krauss, 2017).It was deduced that single layer LSTM provided sufficient accuracy and two layers provided no improvement with the number of LSTM layers and hidden units there in required further investigating with high-frequency data. Notably, increasing the time steps and number of features as inputs increased the RMSE once number of parameters exceeded four.

(Kim & Won, 2018) developed a hybrid model by feeding the parameters of two GARCH type models (amongst GARCH, EWMA, EGARCH) alongside explanatory variables such as interest rate and oil price to learn idiosyncratic features on volatility. These are taken for time (t-22) to t and fed into a three-layer LSTM structure followed by two fully connected layers at time t to yield volatility estimate for t+1.

3. Data

The S&P 500 is an index which tracks the stock prices of 500 of the largest companies in the US. However, the index itself cannot be purchased or traded. Therefore, in order to forecast S&P 500 volatility, we decided to use the SPY Exchange-traded fund (ETF) which was designed to track the S&P 500. We would expect the price and volatility movements of the SPY ETF to follow very closely to the S&P 500, thereby acting as a reliable proxy to the core variable, S&P 500, for which we aim to forecast the volatility of.

The volatility figures were calculated using daily log returns, a standard practice for financial modelling due to its algorithmic and theoretical advantages. Most of the models were trained with past volatility solely, however we also considered using open, high, low, close (OHLC) stock price data to forecast volatility as purported by (Bao et al., 2017), albeit different architecture.

The ex-ante window length was arbitrarily chosen to train the various models was 50 day with subsequent optimising explained in later section ex-post once the best volatility model was selected.

The data split for every model setup is a ratio of 80:20 whereby 80% of the data is utilised for our training set and the remaining 20% serves our testing set. This is customary in practice as observed in literature reviews.

The data was then prepared according for both the test and validation sets. For a window length of 50 and a time series of volatility data, the data would be prepared as the following:

Let $data = [1, 2, ..., N]$ be our volatility data.

$$X[1] = data[1, 2, ..., 50] \text{ , } y[1] = data[51]$$

$$X[2] = data[2, 3, ..., 51] \text{ , } y[2] = data[52]$$

$$X[N-50] = data[N-50, ..., N-1] \text{ , } y[N-50] = data[N]$$

Each sample $X[i]$ represents 50 days of volatility data which is used to forecast the 51st day of volatility.

The dataset is licensed under Public Domain and is sourced from the owner Boris Marjanovic through the Kaggle platform.

3.1. Evaluation Metrics

We used the *mean absolute percentage error (MAPE)* as our main indicator of the model's performance and the *Huber loss* as our loss function as it is less sensitive to outliers when compared with the standard *mean square error (MSE)*

loss function and less sensitive to small errors when compared with the *mean absolute error (MAE)*. We believe the choice of Huber loss is ideal for stock volatility as stock volatility often contains "abnormal" periods of extremely high volatility figures such as during stock market crashes.

4. Architecture & Training

4.1. Vanilla LSTM

This sequential neural network model has one LSTM layer and one dense layer. The LSTM layer has 64 units with the hyperbolic tangent (tanh) activation function. We used a tanh layer over the standard ReLu layer as it allows our model to capture both positive and negative values where as the ReLu activation function converts all negative values to zero. This could result in a large number of neurons in our model becoming inactive as volatility is both positive and negative resulting a less accurate model. The dense layer has one output node which represents our 1 day volatility forecast.

Figure 1. Training and Validation loss for Vanilla LSTM

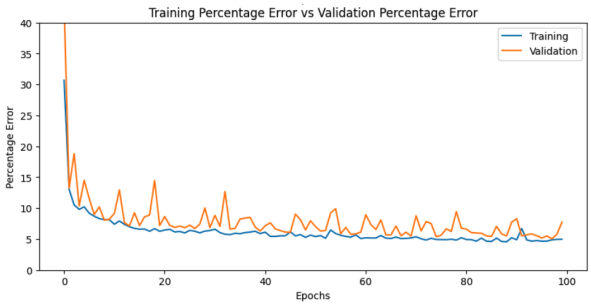

Figure 2. Training and Validation MAPE for Vanilla LSTM

The Adam optimizer with a learning rate of 0.0002 was used. The learning rate was chosen to such that convergence is achieved in 100 epochs whilst making sure the model does not converge too fast as this might cause our model to overshoot the optimal point.

The model was trained with a batch size of 32 for 100 epochs. We see the performance of the model in Figure 1 and 2. We notice a relatively smooth convergence to a validation MAPE of around 5%. We also notice that model is relatively well fitted with no clear indication of over fitting or under fitting.

4.2. Vanilla LSTM Model with OHLC Data

Taking advice from (Bao et al., 2017), we will train the same LSTM model in section 4.1 using OHLC data instead. The data was prepared in a similar way to how we prepared our volatility data however this time with input shape (batch size, window length = 50, number of features = 4) instead of (batch size, window length = 5, number of features = 1). Where the 4 features are open, high, low and close.

Figure 3. Training and Validation loss for Vanilla LSTM with OHLC data incorporated

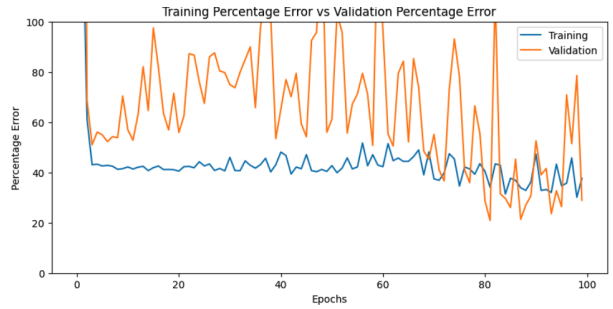

Figure 4. Training and Validation MAPE for Vanilla LSTM with OHLC data incorporated

The rationale for using OHLC data was because (Bao et al., 2017) managed to achieve lower mean squared error with a different setup of using encoders. The Adam optimizer with a learning rate of 0.0002 was used. We noticed in Figure 3 and 4 that the model performed worse using OHLC data. We noticed more volatility between each epoch and signs of overfitting. The MAPE was negatively affect as well going from 7% in the previous model to around 40%.

This is likely because OHLC data does not provide information that is as relevant as past volatility data.

4.3. Multi-Layered LSTM with RMSprop optimiser

This multi-layered LSTM neural network model has two LSTM layers and one dense layer. The first LSTM layer has 64 units with a hyperbolic tangent (tanh) activation function and returns sequences. The second LSTM layer has 32 units with a rectified linear activation function. A dropout layer was added with a rate of 0.2 to prevent over fitting. Finally, a dense layer with one output was used to make the final prediction.

Figure 5. Training and Validation loss for Multi-layer LSTM with RMSprop optimiser

Figure 6. Training and Validation MAPE for Multi-layer LSTM with RMSprop optimiser

Motivated by (Fischer & Krauss, 2017), the RMSprop optimiser with a learning rate of 0.001 was used in contrast to the Adam with a learning rate of 0.0005 employed in the next model as the sole change. In Figure 5 and 6, we notice that using the RMSprop optimiser exhibits clear deterioration in performance when compared to the performance of the Adam optimiser which we will discuss in the following section. We also noticed significant over fitting in this model even after a dropout layer was added. We attempted to increase the dropout rate to prevent over fitting however it did not have any significant impact on the model.

4.4. Multi-Layered LSTM with Adam optimiser

This model is the same as the previous model with the only difference being that an Adam optimiser was used instead.

We noticed that a simple change in optimiser led to a significant improvement in our model. The volatility between each epoch was far less volatile and it achieved a much improved validation MAPE of around 10% when compared to the validation MAPE of around 40% that was achieved using the RMSprop optimiser as we can see in Figure 7 and 8. Thus for our future models, we solely used the Adam optimiser. We also noticed that over fitting is still present although less than with the RMSprop optimiser.

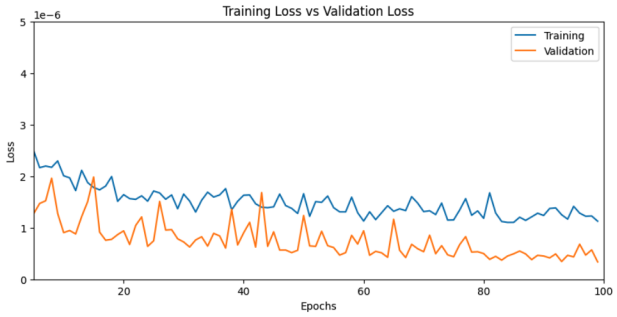

Figure 7. Training and Validation loss for Multi-layer LSTM with ADAM optimiser

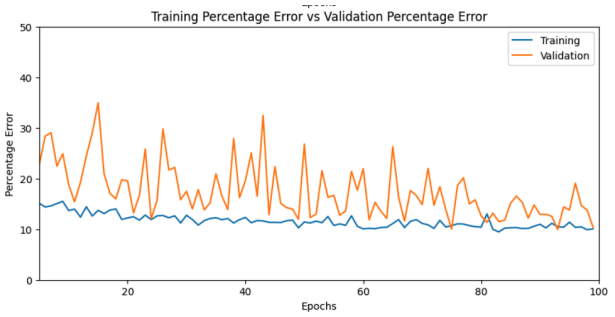

Figure 8. Training and Validation MAPE for Multi-layer LSTM with ADAM optimiser

4.5. 2D-CNN-LSTM Model

This was guided by the recent finding of (Vidal & Kristjanpoller, 2020) which promulgates the conversion of the time series data into an image by the use of Gramian Angular Fields (GAF), a polar transformation. Therefore, one CNN-branch takes this as input and in parallel we have the presence of an LSTM branch too after which both outputs are concatenated and combined in a dense layer for the desired prediction.

Figure 9. Training and Validation loss for 2D-CNN-LSTM Model

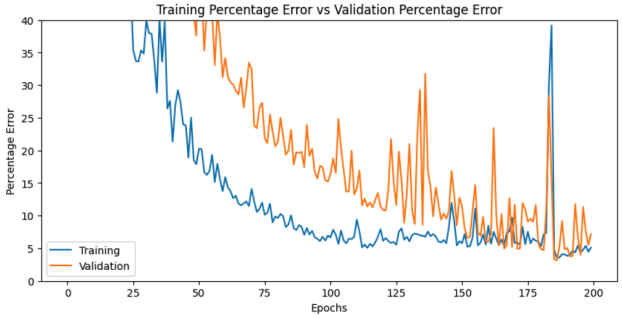

Figure 10. Training and Validation MAPE for 2D-CNN-LSTM Model

The models were trained with a semi optimised learning rate that ensured convergence but convergence which is not too fast. Models were trained over 100 epochs unless more or less epochs were necessary. In this case, it was necessary to run the model over 200 epochs to ensure convergence of training loss.

This model performed relatively well when it reached convergence . However, as Figure 11 and 10 shows, the validation MAPE was substantially volatile with certain epochs being 30% MAPE and the next being 3% MAPE even when the training loss had converged. This behavior clearly indicates an instability intrinsic to the model and unfortunately no equilibrium was achieved following changes in learning rates or other parameters either. This means that the 3% MAPE is not a significant result as the model can easily produce 30% MAPE on a test set.

5. Window Length

A LSTM layer requires a 3 dimensional input, batch size, number of time steps and number of features. The number of time step (which we refer to as window length) could have a significant impact on a model's performance, thus we want to optimise this parameter to further improve our model.

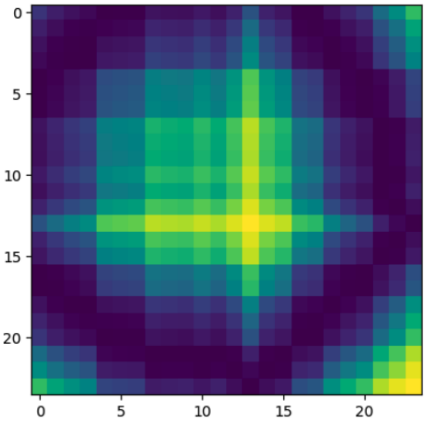

Figure 11. A specimen of a Gramian Angular Field conversion of the first data point in training set - demonstrates that it is abstract and no clear intuition is available apart from a mathematical transformation occurring.

In section 4 we found that the model which produced the performed the best was the vanilla LSTM model. We will therefore use this model as our model to optimise.

The optimisation process is as follows:

1. Prepare our data for a list of different window lengths
2. Train the model over all the window lengths in the list.
3. Select the window length that produced the lowest validation MAPE.

The results can be seen in Figure 12 and 13.

From Figure 12 we notice that the validation errors seem to be lower for shorter window lengths. Thus we further tested the models performance for window lengths 1 to 25 as seen in Figure 13 and found that a window length of 7 days was optimal. The optimal window length of 7 days produced a validation MAPE of 4%. However, we do take note that the model will likely perform just as well with other short window lengths like 2 or 5.

6. Outcome

6.1. Baseline Model

The architectures' performances were compared with an GARCH model, considered as a staple in finance literature for the forecasting of future volatility. The GARCH(a,b) model was optimised for parameters (a,b) and we deduced that the GARCH(2,1) model produced the lowest cross validation error.

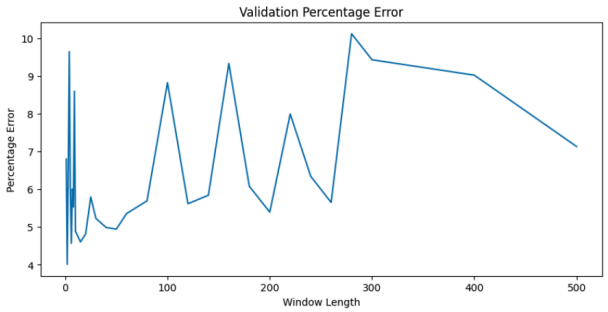

Figure 12. Validation MAPE over different window lengths for Vanilla LSTM model

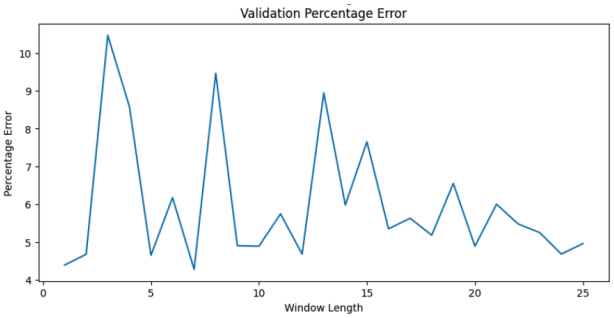

Figure 13. Validation MAPE over short window lengths for Vanilla LSTM model

In detail, the the cross validation entailed selecting from the six top-performing GARCH parameter pairs for which the lowest AIC and BIC was produced. The set for choosing (a,b) was given by [[9,1], [8,1], [10,1], [1,1], [2,1], [2,2]].

From this the GARCH(2,1) had the lowest cross validation error hence formulating our benchmark to make comparisons against the proposed LSTM models. Following the model selection, the GARCH(2,1) model was trained with data up to time t to produce a forecast for t+1 for a fair comparison. This was conducted for 530 days as an additional 30 days were required to calculate 500 days of 30 day volatility.

Ultimately, the benchmark produced a MAPE of 25.8%. Therefore 25.8% MAPE was the baseline error to be reduced for a model to be a robust candidate for forecasting volatility.

6.2. Numerical Results

The labels refer to the models below:

- 1 - Vanilla LSTM
- 2 - Vanilla LSTM with OHLC Data

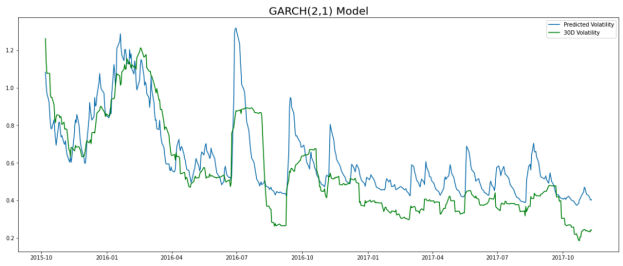

Figure 14. Predicted volatility by GARCH(2,1) compared with realised 30-Day volatility

- 3 - Multi-layer LSTM with RMSprop
- 4 - Multi-layer LSTM with Adam
- 5 - 2D-CNN-LSTM
- 6 - GARCH(2,1) (Benchmark)
- 7 - Optimised Vanilla-LSTM (Final model)

Table 1. Results table displaying all the evaluation metrics for all the candidate models implemented

MODEL	LOSS	MAPE	VAL LOSS	VAL MAPE
1	2.6672E-07	4.9801	2.1702E-07	7.7474
2	1.0602E-05	37.6414	3.0655E-06	28.9146
3	9.1000E-06	31.4368	4.5728E-06	54.1386
4	1.1268E-06	10.0646	3.3705E-07	10.3906
5	2.2949E-07	5.1027	1.3675E-07	7.1779
6	-	22.5400	-	25.8200
7	6.0897E-07	4.6946	2.7807E-07	4.2770

We determine that the best model is the model that is able to produce the lowest validation MAPE whilst reaching it with a relatively stable convergence. Looking at the provided metrics, it appears that Model 5, has the lowest MAPE for validation data, with a training MAPE of 5.1027% and a validation MAPE of 7.1779%. This is followed by Model 1 with a training MAPE of 4.9801% and a validation MAPE of 7.7474%. However if we observe Figure 9 and 10, we notice that model 5 is very unstable and the MAPE of 5.1027% was likely to instability rather than the model's actual performance. This was the main motivation for why we chose model 1 to be our best performing model that we optimised.

Model 2 has a much higher training and validation MAPE than the other models, indicating that using OHLC data instead of volatility data was not as effective for predicting future volatility.

Model 3 and 4 have a higher training and validation MAPE than the Vanilla LSTM model (Model 1), indicating that a multi layered LSTM approach does not perform well.

We also notice that model 3 performs far worse than model 4 with a validation MAPE of 54.1386% as compared to 10.3906%, indicating that the RMSprop optimizer is not a good optimizer for the dataset. Thus, we determined that the Adam was a far better optimiser for our data set.

Model 7 was our optimized vanilla-LSTM model where we found that the model produced the lowest validation MAPE of 4.2770% at a window length of 7 days.

It is also important to note that the optimized GARCH model, namely GARCH(2,1) model (model 6) had a MAPE of 25.8%, which serves as a baseline error to beat. We see that models 1,5 and 7 performed significantly better than the GARCH(2,1) model (model 6). However, we cannot conclude that model 5 is in fact a better model for forecasting volatility due to its high volatility between epochs during training. Thus we can only conclude for certain that the vanilla LSTM model performs better than the GARCH(2,1) model.

Overall, the simpler LSTM models performed better than the more complex models, and using OHLC data instead of volatility data led to worse performance. While 2D-CNN-LSTM model seemingly performed the best and followed closely by the Vanilla LSTM model, the better validation results arise from instability as opposed to predictive power, thus its low validation MAPE result cannot be taken seriously.

We also note that the final validation MAPE of 4.2770% for our optimized vanilla-LSTM model is very low as this result can be interpreted as an expectation that forecasted volatility would only be 4% higher or lower than the true volatility.

However changes could be made to try to further improve the performance of this model. One such change would be to optimize the number of LSTM nodes. In our model, we chose an arbitrarily "safe" option of 64 nodes. Increasing the number of LSTM nodes can improve our model's ability to capture complex patterns or decreasing LSTM nodes can improve our model's performance if the data is overfitted to our model.

Thus, finding the optimal number of LSTM nodes would likely lead to improved accuracy. However, optimizing the number of LSTM nodes leads to computational challenges. Firstly, increasing the number of nodes in an LSTM layer increases the number of parameters in the model, which can significantly increase the computational requirements during training and prediction. This can cause issues such as slower training times, higher memory usage, and longer prediction times.

Secondly, optimizing both window length and the number of LSTM nodes at the same time means that we must run the model for a whole grid of values rather than just a list of values in the case of optimizing a single parameter. This far more computationally expensive. This was one of the main reasons why we did not choose to optimize the number of LSTM nodes as well. We also take note that optimizing over multiple parameters increases the chances of overfitting to the validation data as we use the validation MAPE to select our best models. Thus, nested cross validation methods must be used instead of regular cross validation which leads to further increases in computational requirements.

7. Conclusion

Following the implementation of the several architecture with idiosyncratic changes supported by literature as well as inspired by experimentation, surprisingly we determined that the simplest one-layer LSTM model maintained the superiority in using past volatility data for forecasting unrealised volatility. Although, there is external evidence as aforesaid by (Liu, 2019) that adding more layers does not materially improve accuracy rates.

It is imperative to remark that as the model grew progressed in terms of sophistication of architecture, the models exhibited the tendency to overfitting more frequently. The validation MAPE increased to higher and higher levels. We posit that this stemmed from operating on a very simple and relatively rudimentary data set in contrast to that of cutting-edge literature. Hence it is natural that better performance would be attained with a simple model to avoid fitting noise needlessly.

We found that the shorter window lengths performed the best. In particular, a window length of 7 produced the lowest MAPE of 4.2770% . However, it is likely that any short window length would perform just as well. This means that only the most recent volatility data is needed to forecast volatility with a high degree of accuracy which aligns with market behaviour. Though it may not seem intuitive at first as we expect that longer window lengths means more information thus better performance, we realise that older information may not be as relevant which can negative affect the model's performance.

8. Proximate Investigation

To further our investigation to an interdisciplinary avenue of combining parametric models as inputs to the LSTM structure is to use the insight of (Yang et al.) to formulate a hybrid-LSTM model. LSTM is significantly influenced by features of the training sample with accuracy at highest when training and testing sample resembled each other.

While hybrid-LSTM is better for abrupt changes, this aligns with our best-performing model, Vanilla-LSTM being fit

for a relatively large window length of 50 days since the probability distribution of returns tends to become more stable across time as effect of noise attenuates. For shorter intervals, a composition such as EGARCH-LSTM for the S&P 500 should be evaluated around high volatility events such as first quarter of 2020 during COVID-19. However even this recent advancement only produces better results in 65.9% of the scenarios.

We can also incorporate the latest finding which also uses GAF in place of time series alone in (Chen et al., 2023) as part of *Hybrid Deep Neural Network*. The architecture used volatility-related feature embeddings derived from feeding the GAF into a CNN and concatenating it with numerical data of returns followed by a VGG-like deep network. This uses elements of our attempted architectures yet this abstains from LSTM model itself while yielding markedly improved results. Therefore elements of this arrangement may better bring stability to our training of 2D-CNN-LSTM or even a better alternative for further investigating.

9. Statement About Individual Contributions

The team convened on a biweekly basis with continual communication through social media group channel. The literature review was conducted by Aditya and further discussed by him. In addition he composed the Latex document. Vladimir and Jathniel were responsible for the implementation of the deep networks by programming on Jupyter. This iterative process incorporated Aditya's points based on prevailing literature in order to improve neural network architecture. All three of the authors are in accordance that equal contributions were made by each towards this project.

References

Bao, W., Yue, J., and Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):1–24, 07 2017. doi: 10.1371/journal.pone.0180944. URL https://doi.org/10.1371/journal.pone.0180944.

Chen, W.-J., Yao, J.-J., and Shao, Y.-H. Volatility forecasting using deep neural network with time-series feature embedding. *Economic Research-Ekonomska Istraživanja*, 36(1):1377–1401, 2023. doi: 10.1080/1331677X.2022.2089192. URL https://doi.org/10.1080/1331677X.2022.2089192.

Fischer, T. and Krauss, C. Deep learning with long short-term memory networks for financial market predictions. Technical report, 2017.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Kim, H. Y. and Won, C. H. Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, 103:25–37, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.03.002. URL https://www.sciencedirect.com/science/article/pii/S0957417418301416.

Liu, S., Liao, G., and Ding, Y. Stock transaction prediction modeling and analysis based on lstm. In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 2787–2790, 2018. doi: 10.1109/ICIEA.2018.8398183.

Liu, Y. Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. *Expert Systems with Applications*, 132:99–109, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2019.04.038. URL https://www.sciencedirect.com/science/article/pii/S0957417419302635.

Siami-Namini, S., Tavakoli, N., and Siami Namin, A. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1394–1401, 2018. doi: 10.1109/ICMLA.2018.00227.

Vidal, A. and Kristjanpoller, W. Gold volatility prediction using a cnn-lstm approach. *Expert Systems with Applications*, 157:113481, 2020. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113481. URL https://www.sciencedirect.com/science/article/pii/S0957417420303055.

Yang, A., Ye, Q., and Zhai, J. Volatility forecasting with hybrid-long short-term memory models: Evidence from the covid-19 period. *International Journal of Finance & Economics*, n/a(n/a). doi: https://doi.org/10.1002/ijfe.2805. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ijfe.2805.