An Attention Free Long Short-Term Memory for Time Series Forecasting

Hugo Inzirillo
CREST - Institut Polytechnique de Paris
Simons
Email: hugo@simons.finance

Ludovic De Villelongue
Université Paris Dauphine-PSL
Simons
Email: ludovic.de-villelongue@dauphine.eu

***Abstract*—Deep learning is playing an increasingly important role in time series analysis. We focused on time series forecasting using attention free mechanism, a more efficient framework, and proposed a new architecture for time series prediction for which linear models seem to be unable to capture the time dependence. We proposed an architecture built using attention free LSTM layers that overcome linear models for conditional variance prediction. Our findings confirm the validity of our model, which also allowed to improve the prediction capacity of a LSTM, while improving the efficiency of the learning task.**

I. Introduction

Recently we have seen a strong interest in the development of models based on attention mechanisms, for encoder decoder based models. More recently, some researchers have focused on the development of more efficient mechanisms, and developed some alternatives such as the attention free mechanism [1] an efficient variant of Transformers [2]. The initial problem was to improve the performance of the encoder-decoder model. The one behind the attention mechanism was to allow the decoder to use the most relevant parts. However, the complex structure of these transformers does not seem to significantly improve the prediction when dealing with time series [3]. In this paper we have retained the usefulness of the attention mechanism and more particularly its capacity to use the most relevant parts of the input sequence. We have combined this strength with the power of a LSTM [4] to design a new framework for time series prediction for assets for which linear models are no longer sufficient to model the conditional volatility.

Digital assets became popular for their high volatility, typically where linear models may have difficulties on predictiont task. In this paper we focus on the modeling of volatility by feeding the outputs of a GARCH model and adjust the prediction using an Attention Free Long Short-Term Memory (AF-LSTM). The results are promising on various assets and make the AF-LSTM Layer a more efficient predicator for time series.

II. Deep Volatility

The assumption made about the errors of classical linear models is that they are homoscedastic, they follow a centered distribution with a standard deviation σ. This assumption is not realistic in financial applications. In fact, in practice, we find that the errors are heteroscedastic and this is more or less strong depending on the asset class. In other words, the variance is time dependent. When we look at certain asset classes, we observe volatility clusters and this is even stronger when we talk about digital assets. This characteristic is typical of conditional heteroscedasticity [5].

A. Autoregressive conditional hererocesdastic models

ARCH is a univariate and nonlinear model in which volatility is estimated with the square of past returns. Despite its appealing features, such as simplicity, nonlinearity, easiness, and adjustment for forecast, the ARCH model has certain drawbacks:

1) equal response to positive and negative shocks
2) strong assumptions such as restrictions on parameters
3) possible misprediction due to slow adjustments to large movements

B. GARCH Model

GARCH model [5] is also an approach to attempt to measure the volatility of certain financial assets. GARCH models has been seen for decades as one of the leading tools in terms of estimating the volatility of the financial returns. Three main reasons have made GARCH models so popular:

1) the variance of residuals is auto-correlated
2) it allows to manage volatility clusters
3) it represents the conditional variance for each step of time

A GARCH(P,Q) is defined as:

$$\sigma_t^2 = \omega + \sum_{p=1}^{P} \alpha_p \epsilon_{t-p}^2 + \sum_{q=1}^{Q} \beta_q \sigma_{t-q}^2 \tag{1}$$

where $\epsilon_t = \sigma_t \mu_t$, and $\mu_t \overset{i.i.d}{\sim} \mathcal{N}(0,1)$. If $\alpha + \beta < 1$ the volatility fluctuates around the unconditional variance and is given by:

$$\sigma^2 := Var(r_t) = \frac{\omega}{1-\alpha-\beta}$$

C. GJR-GARCH Model

The Glosten-Jagannathan-Runkle GARCH (GJR-GARCH) [6] is a variant of a GARCH model, the underlying assumption is that negative shocks in the previous period have a

stronger impact on the variance than positive shocks. The GJR-GARCH(P,Q) is defined as:

$$\sigma_t^2 = \omega + \sum_{p=1}^{P} \alpha_p \epsilon_{t-p}^2 \gamma_p \epsilon_{t-p}^2 \mathbf{1}_{\{\epsilon_{t-p}<0\}} + \sum_{q=1}^{Q} \beta_q \sigma_{t-q}^2 \quad (2)$$

where $\epsilon_t = \sigma_t \mu_t$, and $\mu_t \overset{i.i.d}{\sim} \mathcal{N}(0,1)$. γ_p equals to 0 implies that there is no asymmetric shock on the time series. If $\alpha + \frac{\gamma}{2} + \beta < 1$ the volatility fluctuates around the unconditional variance, given by:

$$\sigma_{GJR}^2 := Var(r_t) = \frac{\omega}{1 - \alpha - \frac{\gamma}{2} - \beta}$$

Deep learning models are proposed to estimate the conditional volatility of cryptocurrencies. Recurring neural networks (RNNs) are built to improve GARCH predictions. For this purpose, a LSTM using sequences of past values of volatility is taken as starting point and updated with embeddings corresponding to the GARCH predicted volatility.

D. LSTM

The LSTM framework [4] are RNNs with gated mechanisms designed to avoid vanishing gradient. The blocks of LSTM contain 3 non-linear gates that make it smarter than a classical neuron, as well as a memory for sequences. The 3 types of non-linear gates include :

- Input Gate (i_t) : decides which values from the input are used to update the memory
- Forget Gate (f_t) : handles what information to throw away from the block
- Output Gate (o_t): handles what will be in output based on input and memory gate.

During the training step, each iteration provides an update of the model weights proportional to the partial derivative and in some cases the gradient may be vanishingly small and weights may not be updated. The LSTM networks is defined in Figure 1 as:

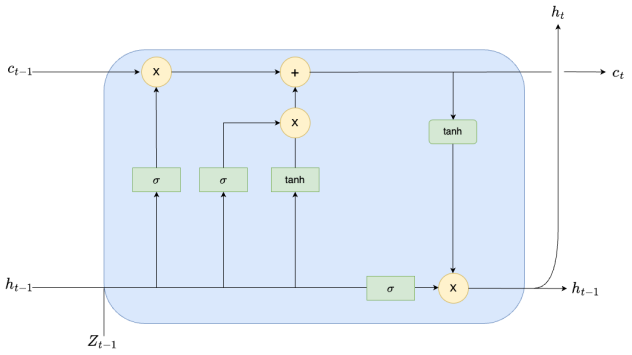

Fig. 1. LSTM Cell

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, z_{t-1}] + b_f), \\ i_t &= \sigma(W_i[h_{t-1}, z_{t-1}] + b_i), \\ \tilde{c}_t &= \tanh(W_c[h_{t-1}, z_{t-1}] + b_c), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\ o_t &= \sigma(W_o[h_{t-1}, z_{t-1}] + b_o), \\ h_t &= o_t \odot \tanh(c_t). \end{aligned} \quad (3)$$

where $\Theta \triangleq \{W_f, W_i, W_c, W_o\}$ are the weights of the model and z_{t-1} is the input vector at time t. At each time step, the memory cell takes the current input z_{t-1}, the previous hidden state h_{t-1} and the previous memory state c_{t-1}.

The goal is to estimate $\hat{y}_t \triangleq f(h_t; \theta)$. In this case it is equivalent to estimate:

$$\mathbb{E}[\sigma_t^2 | \mathcal{F}_{t-1}] = \mathbb{E}[\sigma_t^2 | (\sigma_{t-1}^2, ..., \sigma_{t-q}^2), (\epsilon_{t-1}^2, ..., \epsilon_{t-p}^2)].$$

The loss function given by $L : Y \times Y \to \mathbb{R}_+$ measures the distance between two outputs. In this case, the $l_2 - loss$ is used. It is given by:

$$\begin{aligned} L(y, f(z;\theta)) &= (y - f(z_{0:T-1};\theta))^2, \\ &= \frac{1}{T}\sum_{t=1}^{T}(y_t - f(z_{t-1};\theta))^2. \end{aligned} \quad (4)$$

The number of parameters of each layer in the model should be carefully selected to handle overfitting or underfitting situations.

E. Deep GARCH

With the expanding need to improve volatility modeling precision, GARCH models have been a redundant concern in finance and economic literature. However, the relation between the conditional variances being non linear, these models have found their limits. Some researcher works focusing on introducing non linearity in these models rely on neural networks to introduce non-linearity into classical models [7], [8]. Kim and Won [9] mixed the LSTM with multiple GARCH-type models to introduce non linearity [10]. In this paper we seek to improve the predictive ability of GARCH models. We do not question the estimation of the model but we try to correct the prediction made at each time step. To do this, we recover the conditional variances obtained after a first estimation made by the GARCH model and then we apply filters and linear transformations made possible by the use of several AF-LSTM layers.

F. Attention Free Block

The transformer architecture has made it possible to develop new models capable of being trained on large dataset while being much better than recurrent neural networks such as LSTM. The Attention Free Transformer [1] introduces the attention free block, an alternative to attention block [2], which eliminates the need for dot product self attention.

The transformer is an efficient tool to capture the long term dependency. Just like the transformer, the AF Block includes interactions between queries, keys and values. The difference is that the AF block firstly combines key and value together with a set of learned position biases described in [1]. Then the query is combined with the context vector. Let Q, K, V denote the query, key and value, respectively.

$$\begin{aligned} Y_t &= \sigma_q(Q_t) \odot \frac{\sum_{t'=1}^{T} exp(K_{t'} + w_{t,t'}) \odot V_{t'}}{\sum_{t'=1}^{T} exp(K_{t'} + w_{t,t'}))}, \\ &= \sigma_q(Q_t) \odot \sum_{t'=1}^{T} (\text{Sofmax}(K) \odot V)_{t'}. \end{aligned} \tag{5}$$

where $Q = ZW_q$, $K = ZW_k$ and $V = ZW_v$. The activation function σ_q is the sigmoid function and $w_t \in \mathbb{R}^{T \times T}$ is a learned matrix of pair-wise position biases.

G. Attention-free LSTM

As explained in the models described above, the expectation of volatility is conditioned by past values. However some of these variables may at some point not be relevant for the prediction. To reproduce this observable phenomenon, we added an Attention Free Block to build an Attention-free LSTM layer (AF-LSTM). The Attention-free LSTM layer processing step can be split in several parts. The input at each time step will firstly be processed by an Attention Free layer defined as:

$$\tilde{Z}_{t-1}^{(l)} = AF_1^{(l)}(W_1^{(l)}; z_{t-1}), \tag{6}$$

where $Z_t \in \mathbb{R}^q$, q the sequence length of our input and $AF_1^{(l)}$ is the Attention Free of the $l-th$ layer:

$$\begin{aligned} x_{t-1}^{(l)} &= \text{Concat}(W_x^{(l)} Z_{t-1}^{(l)} + b_x^{(l)}), \\ Q_{t-1}^{(l)} &= W_q^{(l)} x_{t-1}^{(l)}, \\ K_{t-1}^{(l)} &= W_k^{(l)} x_{t-1}^{(l)}, \\ V_{t-1}^{(l)} &= W_v^{(l)} x_{t-1}^{(l)}, \\ \eta_{t-1}^{(l)} &= \sum_{t=1}^{T} \text{Softmax}(K_{t-1}^{(l)}) \odot V_{t-1}^{(l)}, \\ \tilde{Z}_{t-1}^{(l)} &= \sigma(Q_{t-1}^{(l)}) \odot \eta_{t-1}^{(l)}. \end{aligned} \tag{7}$$

we apply the *Layer Normalization (LN)* function [11]

$$LN(x; \psi; \phi) = \psi \frac{(x - \mu_x)}{\sigma_x} + \phi,$$

and filter our $\tilde{Z}_{t-1}^{(l)}$ using the *ReLU* activation function, $ReLU(x) = max(0, x)$.

Hence we have:

$$\tilde{z}_{t-1}^{(l)} = ReLU(LayerNorm(\tilde{Z}_{t-1}^{(l)})), \tag{8}$$

On the right side of 2 the input is the same as the left size. We denote $\bar{Z}_{t-1}$ the output of the attention-free mechanism $AF_2^{(l)}$. $\bar{Z}_{t-1}$ will resume the information of the input tensor hence it will be multiplied by the ouput of the left side to only select the observations that have a positive impact on the prediction.

$$\begin{aligned} \bar{Z}_{t-1}^{(l)} &= AF_2^{(l)}(W_2^{(l)}; Z_{t-1}), \\ \bar{z}_{t-1}^{(l)} &= LN(\bar{Z}_{t-1}; \psi^{(l)}; \phi^{(l)}). \end{aligned} \tag{9}$$

The output from the two channels will be multiplied to apply the filter on our input sequence.

$$\begin{aligned} \eta_{t-1}^{(l)} &= (\tilde{z}_{t-1}^{(l)} \odot \bar{z}_{t-1}^{(l)}), \\ \zeta_{t-1}^{(l)} &= LN(\eta_{t-1}^{(l)}; \psi^{(l)}; \phi^{(l)}). \end{aligned} \tag{10}$$

$\zeta_{t-1}^{(l)}$ will be the input of a parametrized function $g_w^{(l)}$ such as:

$$h_t^{(l)} = g_w^{(l)}(\zeta_{t-1}^{(l)}), \tag{11}$$

in Fig 2 $g_w^{(l)}(.)$ is a LSTM defined in Fig 1. Using the output of $g_w^{(l)}(.)$ we can estimate the output of the $l-th$ layer $\sigma_t^{(l)}$ given by:

$$\sigma_{t-1}^{(l_{final})} = W_y^{(l_{final})} h_{t-1}^{(l_{final})} + b_y^{(l_{final})}, \quad l := \{1, ..., l_{final}\}. \tag{12}$$

The ouput of our network is defined as σ_t which is a proxy of the 5d rolling volatility.

III. Methodology

A. Empirical Risk Minimization

Considering the training data $\mathcal{D} \triangleq \{Z^{(i)}\}_{i=1}^{m} \in \mathcal{Z}^m$ and a function $f : \mathcal{Z} -> \mathcal{Y}$, the ERM is defined by:

$$\hat{\mathcal{R}}_{\mathcal{D}}(f) := \mathcal{L}(f, Z^{(i)}). \tag{13}$$

Lets define $\mathcal{A}_{erm}$ as an empirical risk minimization algorithm. Hence, given a set of $\mathcal{F}$, the objective function is:

$$A_{erm}(\mathcal{D}) = \arg\min_{f \in \mathcal{F}} \quad \hat{\mathcal{R}}_{\mathcal{D}}(f), \tag{14}$$

where f is the residual mean squared error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} \left(Y_t - \hat{Y}_t\right)^2}{n}}. \tag{15}$$

B. Data

The data retrieved are the close prices of a single cryptocurrency asset from inception. The log returns of this time series are calculated and fed to the GARCH model. The distribution chosen for the standardized residuals is a normal one. The lag parameters are defined to perform a first order GARCH model. Once the model is fitted, the forecasted log returns are given by the following formula:

$$\begin{aligned} r_t &= \sigma_t \epsilon_t, \\ \sigma_t^2 &= \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2. \end{aligned} \tag{16}$$

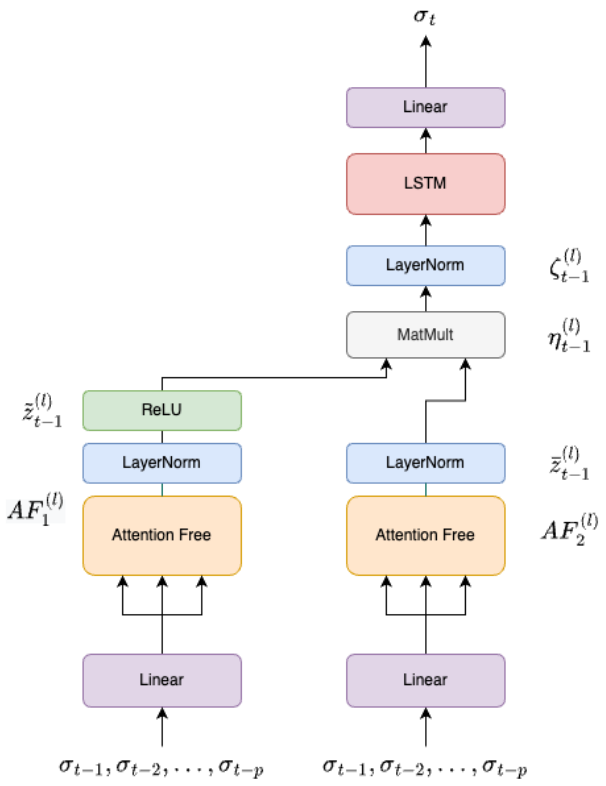

Fig. 2. Attention-Free LSTM Layer

Then, the volatility calculated on a five period rolling window is compared with the forecasted volatility obtained by dividing the GARCH forecasted log returns with ϵ_t. The volatility at time step t is calculated by using the moving average of the previous 4 time steps' returns (from time step t-5 to time step t-1).The train and test sets are composed of real volatilities and volatilities forecasted by the GARCH model. They are split according to the 80/20 rule on a five period window and plugged to the LSTM and AF-LSTM. Once the train and test sets are created, two scalers are initialized on x and y to scale and translate them to a (0,1) range. Afterwards, the train set for x and y are fitted and transformed, and the test set for x and y only transformed. The fit method is calculating the mean and variance of each of the features present in the data. The transform method is transforming all the features using the respective mean and variance. The parameters learned by the model using the training data will help to transform the test data. Once it is done, the train and test sets arrays are converted to tensors and inserted into dictionaries with the scalers.

The LSTM network is fed with the following parameters :

- *Input size*: 2 (number of expected features in the input)
- *Hidden size*: 64 (number of features in hidden state h)

The AF-LSTM network is fed with the following parameters:

- *Input size*: 2 (number of expected features in the input)
- *Hidden size*: 64 (number of features in hidden state h)
- *Maximum sequence length*: 1000 (maximum number of timesteps to be fed in)
- *Dim*: 2 (embedding dimension of the keys, queries and values)
- *Hidden size* : 64 (hidden dimension used inside the Attention Free Transformer)

After creation of the LSTM and AF-LSTM networks with an Adam optimizer, the mean squared error loss function is computed on 1000 epochs with a learning rate of 0.001. Once the predictions are obtained, the scalings of the predicted and actual volatilities are inversed to get back to true volatility values. For comparison purposes, the results are transformed to single arrays. The root mean squared errors (RMSE) for the train and test sets can then be calculated. Finally, the actual volatilities and predicted volatilities for the train and test sets are plotted using the two models.

C. Estimation

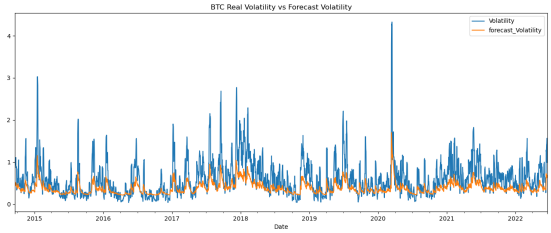

Fig. 3. Actual Volatility vs Forecasted Volatility by GARCH(1,1)

A scalability matter is encountered when comparing the GARCH forecasted volatility with the actual rolling volatility in Figure 3. The GARCH (1,1) model is thus a weak learner for such a time series.

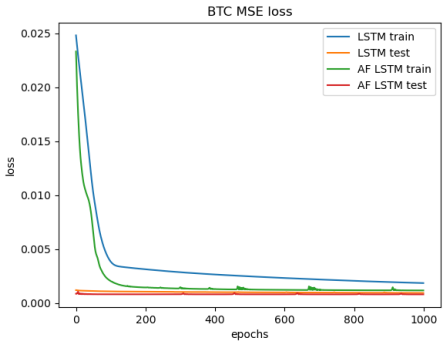

Fig. 4. LSTM and AF-LSTM Loss Functions for Train and Test Sets

The loss function in Figure 4 is enlightning the better performance of the AF-LSTM in taking information into account. On the one hand, regarding the train set, the loss of the AF-LSTM is lower than the LSTM's loss on all epochs. On the other hand, regarding the test set, both losses are

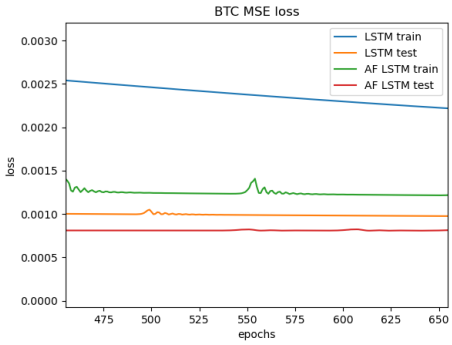

Fig. 5. LSTM and AF-LSTM Loss Functions Zoomed for Train and Test Sets

bottommost and are decreasing at a truly slight rate. On a smaller interval, it is clearly noticeable that the AF-LSTM has a lower loss than the LSTM on the test set, as enlightened in Figure 5. It is the case for the integrality of the epochs.

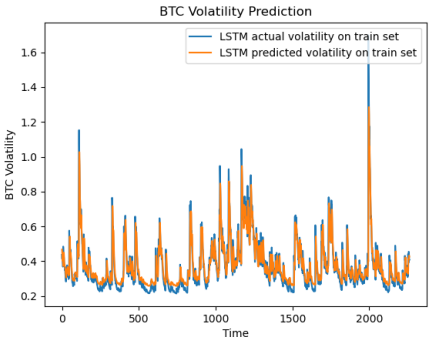

Fig. 6. Actual Volatility vs LSTM Forecasted Volatility for Train Set

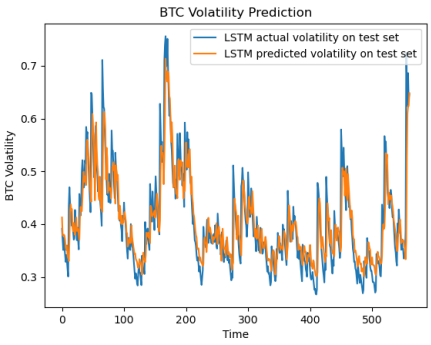

Fig. 7. Actual Volatility vs LSTM Forecasted Volatility for Test Set

According to the predictions in Figure 6 and Figure 8, the AF-LSTM is able to catch more patterns than the LSTM. In fact, the lack of amplitude perceived in the LSTM is partially corrected by the AF-LSTM on the train set. When we look at the predictions on the test set in Figure 7 and Figure 9, the main difference to notice is the reduction of the offset between

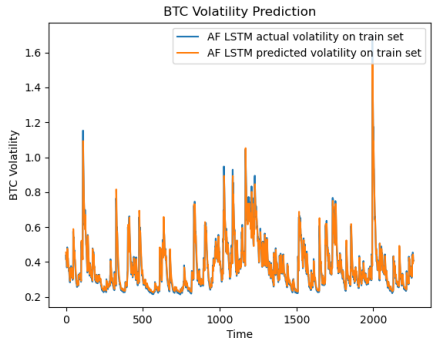

Fig. 8. Actual Volatility vs AF-LSTM Forecasted Volatility for Train Set

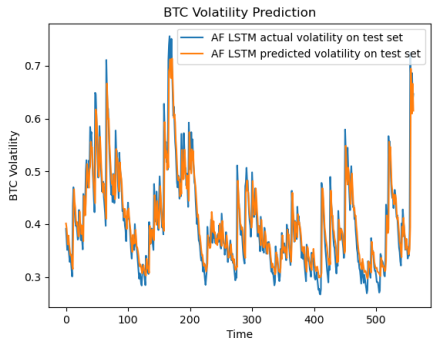

Fig. 9. Actual Volatility vs AF LSTM Forecasted Volatility for Test Set

the LSTM and AF-LSTM results compared to the train set. However, the superiority of the AF-LSTM in approximating the rolling volatility is still demonstrated.

The RMSE confirm those results:

Dataset	LSTM RMSE	AF LSTM RMSE
Train Set	0.061	0.051
Test Set	0.044	0.042

IV. Results

By reducing the time and space complexity of the prediction algorithm, the AF-LSTM Figure 9 has a real advantage over the LSTM in terms of sequential computation time on a large data set. In fact, the Attention Free Mechanism has a linear memory complexity with respect to both the context size and the dimension of features, making it compatible to both large input and model sizes. Moreover, the loss function and predictions are improved as additional information are captured by this model. The AF-LSTM thus demonstrates competitive performance while providing increased efficiency at the same time.

V. Conclusion

The GARCH(1,1) derives its value based solely on the most recent updates of ϵ and σ, which is helpful if the price changes are relatively similar. However, this model's weakness emerges

when there are more sudden jumps in log returns. To consider the repercution of theses shocks on the volatility, the LSTM model should be used, as it tries to mimick the real data with lags. To boost the performance and increase the power of prediction, using an Attention Free LSTM layer can be very useful. It eliminates the quadratic complexity of the self attention mechanism while selecting the past observation that will have a positive impact on the prediction. In fact, instead of carrying out the dot product for the creation of the attention matrix, a weighted average of the values is carried out for each target position. It maintains all the advantages of the dot product without the computation cost.

REFERENCES

[1] S. Zhai, W. Talbott, N. Srivastava, C. Huang, H. Goh, R. Zhang, and J. M. Susskind, "An attention free transformer," *CoRR*, vol. abs/2105.14103, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[3] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" 2022.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[5] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[6] L. R. Glosten, R. Jagannathan, and D. E. Runkle, "On the relation between the expected value and the volatility of the nominal excess return on stocks," *The Journal of Finance*, vol. 48, no. 5, pp. 1779–1801, 1993.

[7] N. Nikolaev, P. Tino, and E. Smirnov, "Time-dependent series variance learning with recurrent mixture density networks," *Neurocomputing*, vol. 122, pp. 501–512, 2013, advances in cognitive and ubiquitous computing.

[8] W. Kristjanpoller and M. C. Minutolo, "Gold price volatility: A forecasting approach using the artificial neural network–garch model," *Expert Systems with Applications*, vol. 42, no. 20, pp. 7245–7251, 2015.

[9] H. Y. Kim and C. H. Won, "Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models," *Expert Systems with Applications*, vol. 103, pp. 25–37, 2018.

[10] W. K. Liu and M. K. P. So, "A garch model with artificial neural networks," *Information*, vol. 11, no. 10, 2020.

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[12] A. Charles and O. Darné, "The accuracy of asymmetric garch model estimation," *International Economics*, vol. 157, pp. 179–202, 2019.

[13] H. Nessrine and B. Regaieg, "The glosten-jagannathan-runkle-generalized autoregressive conditional heteroscedastic approach to investigating the foreign exchange forward premium volatility," *International Journal of Economics and Financial Issues*, vol. 6, pp. 1608–1615, 09 2016.

[14] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," 2019.

[15] M. Buczyński and M. Chlebus, "Garchnet -value-at-risk forecasting with novel approach to garch models based on neural networks," 06 2021.