The Superiority of Direct Neuro Volatility Forecasts Over GARCH and Machine Learning Forecasts for Financial Assets

***Abstract*—In traditional approaches, the volatility forecasts of financial assets are obtained in two steps: first, by modeling conditional variance using GARCH models and then by taking the square root of the conditional variance forecasts. There has been growing interest in neuro volatility forecasts using neuro GARCH models for conditional variance. The asymptotic variance of the sample standard deviation depends on excess kurtosis, which is very large for heavy-tailed distributions, leading to a significantly high asymptotic variance in the volatility estimate when using square root forecasts. In this study, we explore various neural network models, including RNN, LSTM, and GRU, for direct volatility forecasting. Results demonstrate that the data-driven neuro volatility model outperforms the GARCH neuro volatility model with lower forecast errors and better captures volatility changes during the testing period, while other models exhibit mostly linear forecasts and miss key fluctuations.**

***Index Terms*—Financial risk, neuro volatility models, volatility forecasts**

I. Introduction

Incorporating more robust risk measures in investment decisions ensures optimal capital allocation and helps minimize or mitigate future financial losses. When estimating risk, the most widely used measures are volatility, value-at-risk (VaR), and expected shortfall (ES). Volatility is the standard deviation of the log returns of a financial asset. If the underlying distribution of returns is known, both VaR and ES can be calculated using volatility forecasts. Following the financial crisis in 2008, there has been a shift to using more complex and robust risk measures. The governing body known as the Basel III Committee [1] provides recommendations and regulations for financial institutions to implement more sophisticated models to estimate risk and avoid financial losses.

Sample variance is used as an estimator for population variance, and the square root of the sample variance is used to estimate population standard deviation. The asymptotic variance of the sample standard deviation depends on excess kurtosis. Excess kurtosis is very large for heavy tail distributions, leading to a very large asymptotic variance of the estimate. It has been noted that the return distributions of most financial assets tend to be non-normal and exhibit heavy or fat tails t-distributions with degrees of freedom less than four [2]. In traditional approaches, volatility forecasts are derived by initially modeling the conditional variance using GARCH models and subsequently taking the square root of the conditional variance forecasts [3]. Thus, using generalized autoregressive conditional heteroskedasticity (GARCH) models to obtain volatility forecasts is inefficient. Recently, a direct estimate for the volatility is proposed in [2], and the asymptotic variance of the new estimate does not depend on excess kurtosis. Thus, for heavy-tailed distributions asymptotic variance is finite. Assuming normality simplifies mathematics significantly. However, when non-normality, which more accurately reflects real-world scenarios is considered, the complexity increases substantially. This puts risk managers in a challenging position. By assuming normality, they risk underestimating potential dangers, often with severe consequences. On the other hand, applying non-normal methods is far more complex. Moreover, probability statements (confidence intervals) for a parameter are made with an estimate, table value, and the standard deviation of the estimate. Thus, modeling variance or conditional variance is redundant and it is appropriate to model standard deviation directly.

The use of neural networks (NNs) to improve volatility forecasts has been extensively studied across various markets and different models. Early work [4] constructed a semi-nonparametric nonlinear GARCH model based on artificial neural networks (ANNs) and demonstrated that the ANN model captured volatility effects missed by traditional models like GARCH, exponential GARCH, and Glosten, Jagannathan, and Runkle GARCH in forecasting stock returns in major financial markets. Hamid and Iqbal [5] compared NN-based volatility forecasts with implied volatility from S&P 500 Index futures options. They found that NNs outperformed implied volatility forecasts and were closely aligned with realized volatility. Later, Aragonés et al. [6] explored whether neural networks could improve volatility forecasts for the Spanish stock market. Their results showed that NN forecasts improved the information content of implied volatilities and enhanced predictive performance. Wang et al. [7], applied NN to forecast prices, revealing that forecasting performance varied with volatility models and neuron configurations but was not significantly affected by activation functions. Monfared and Enke [8] explored different NN models to enhance the GJR(1,1) method for volatility estimation. In subsequent years, hybrid approaches emerged. For instance, Kristjanpoller et al. [9] tested a hybrid Neural Networks-GARCH model on Latin American stock exchanges, showing improved forecasting performance over traditional GARCH models. This approach was extended by Kristjanpoller and Minutolo [10], who demonstrated a 30% improvement in oil price volatility forecasts by incorporating financial variables into a hybrid model. Research on bitcoin volatility forecasting in [11] further developed the hybrid ANN-GARCH model with preprocessing techniques. Liu [12] compared LSTM recurrent neural networks (RNNs) with traditional GARCH models for financial volatility forecasting, finding that LSTMs performed better than GARCH models for large interval forecasts. Recently, Hoque et al. [13] explored neuro volatility models for direct volatility forecast and assessed their role in studying the resiliency of the financial system. This current study investigates more complex data-driven neuro volatility models with recent data and compares the results against the GARCH neuro volatility model.

The remainder of the paper is organized as follows. Section II provides the details of the direct volatility estimate and neuro volatility models. In Section III, experiments and results are described. Finally, concluding remarks are given in Section IV.

II. Neuro Volatility Models

A. Volatility Estimate

The idea of deriving the direct estimate for volatility is based on the estimating function (EF) theory. In traditional methods like maximum likelihood, it becomes very difficult to get closed-form solutions when we have complicated distribution functions. Thus, in EF, rather than imposing conditions on the estimator, it focuses on the properties of the EF itself (e.g., unbiased EF).

Let P_t denote the price of an asset at time t, and let the log return at time t be represented as r_t, where $r_t = \log\left(\frac{P_t}{P_{t-1}}\right)$. For a sequence of independent random variables $r_t, t = 1, \ldots, n$, the variance of the sample variance is given by $Var(s^2) = (k+2)\frac{\sigma^4}{n}$, where k represents the excess kurtosis. It can be shown that the asymptotic variance of the sample standard deviation is $As.Var(s) = \frac{(k+2)}{4}\frac{\sigma^2}{n}$. For heavy tail distributions, excess kurtosis is infinite and thus the asymptotic variance of s is infinite (see [2] for more details).

Consider a class of martingale estimating functions generated by the martingale differences $h_t(\sigma) = |r_t - \mu| - \rho\sigma$ to estimate the volatility σ where μ is mean of r_t and $\rho = Corr(r_t - \bar{r}, sgn(r_t - \bar{r}))$ ($sign$ correlation). The optimal EF which maximizes the information is given by $g^*(\sigma) = \sum_{t=1}^{n} a^*_{t-1} h_t$ where $a^*_{t-1} = \rho / \left((1-\rho^2)\sigma^2\right)$. Then the optimal estimate of σ is given by $\hat{\sigma}_n = \frac{1}{n\rho}\sum_{t=1}^{n}|r_t - \mu|$ and $As.Var(\hat{\sigma}_n) = \frac{1}{n}\left(\frac{1-\rho^2}{\rho^2}\right)\sigma^2$. Note that the asymptotic variance of the new estimate does not depend on excess kurtosis, and thus, it is finite for heavy-tailed distributions. Additionally, a direct advantage of the estimate of ρ is that it can be used to identify a t-distribution with appropriate degrees of freedom (ν) (see [2] for more details).

B. Neuro Volatility Models

The simplest NN with no hidden layers is equivalent to linear regression. The coefficients associated with inputs/predictors are called weights and the forecasts can be obtained by linear combination of inputs. Once we introduce a hidden layer(s), an NN becomes non-linear and these are known as multilayer feedforward NN. In order to determine optimal weights, we can minimize a cost function. With a sufficient number of neurons in the hidden layers, a NN can successfully approximate a continuous non-linear function and translate inputs to a desired output.

For time series data, lagged values of the target series can be used as inputs to an NN and it is called an NN autoregression model or NNAR model. The notation NNAR(p, k) indicates a non-seasonal NNAR model with a single hidden layer. Here p is the number of lagged inputs and k is the number of nodes/neurons in the hidden layers. With seasonal data, observed values from the same season are used as inputs. Then NNAR model can be written more generally as NNAR$(p, P, k)_m$ where P is the autoregressive orders of the seasonal part and m is the seasonal period (daily/weekly).

A widely used NNAR model specifically designed for time series forecasting is introduced in [14] and this model can be implemented in R using the `nnetar` function from the `forecast` package. It fits a single-layer NN to the target variable by considering its lagged values as inputs. Additionally, the model can be extended to incorporate external features when training the neural network, enhancing its forecasting capability. When there are multiple hidden layers in the NN, the inputs are combined using the linear relationship,

$$z_j = b_j + \sum_{i=1}^{p} w_{i,j} x_i,$$

where b_j and $w_{i,j}$ are weights that the network estimates. The weights are originally set randomly and are adjusted using observed function values, x_i. In order to ensure accurate weights, the network is trained multiple times, each time with different random starting weights, and the results are then averaged. This occurs for each neuron in the hidden layer. z_j is then adjusted using a nonlinear function such as a sigmoid function,

$$s(z) = \frac{1}{1+e^{-z}}.$$

The sigmoid function is a popular choice because it tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers. For the neuro volatility models, each hidden neuron in the first layer would transform the observed volatility ($V_t = |r_t - \bar{r}|/\hat{\rho}$ where $\hat{\rho} = corr(r_t - \bar{r}, sign(r_t - \bar{r}))$) by

$$a_l = g(w_{l,0}^{(1)} + \sum_{j=1}^{p} w_{l,j}^{(1)} V_{t-j})$$

where g is the sigmoid function, $l = 1, 2, ..., k$ is the index of the lth node, $w_{l,0}$ is the intercept/bias term, $w_{l,j}$'s are the weights, and the superscript (1) indicates that the outputs from layer one, the input layer, are being transformed. Subsequent transformations are calculated in a similar manner but with the outputs from the previous layer as inputs.

$$a_l^{(k)} = g(w_{l,0}^{(k-1)} + \sum_{j=1}^{p} w_{l,j}^{(k-1)} a_l^{(k-1)})$$

Figure 1 illustrates a general NN structure with a single hidden layer. The network predicts the target series value at time $t+1$, denoted as X_{t+1}, based on lagged values of target series $(X_t, X_{t-1}, \ldots, X_{t-p})$. When lagged values of observed volatility ($V_t = |r_t - \bar{r}|/\hat{\rho}$) are used as inputs and the NN is obtained from `nnetar`, it is called data-driven neuro volatility (DD.Neuro.Volatility/DD.Neuro) model in this study.

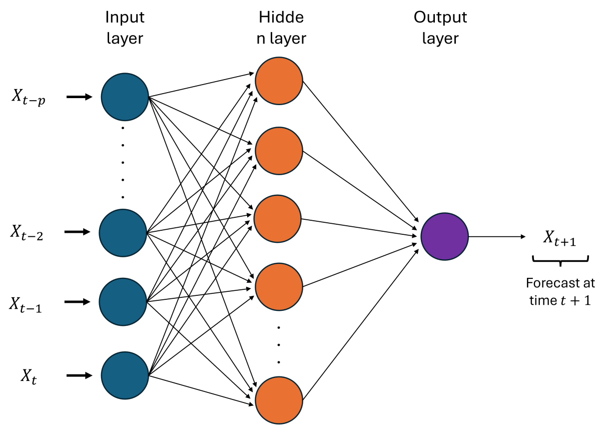

Fig. 1. NNAR Model with Single Hidden Layer

1) Recurrent Neural Network (RNN): For more complex NN models, we may have more than one hidden layer and in some cases loops in the hidden layer which leads to identifying more features. RNN is one such model and it can remember previous inputs and use that information to influence the current output. However, when training RNN on a long sequence, the gradient (which is used to update the network parameters during training) can become very small as they are backpropagated through time. This means the network struggles to learn dependencies between distant time steps in the sequence as the influence of earlier inputs diminishes rapidly. Also, RNNs process sequences one step at a time and thus they can be slow. This is addressed by introducing variations of RNNs such as long short-term memory (LSTM) and gated recurrent units (GRU).

2) Long Short-Term Memory (LSTM): LSTMs have a more complex structure with memory cells and three "gates" (input, forget, and output) that control the information flow into and out of the memory cell. The input gate decides what new information to store in the cell's memory and what information to discard from the cell's memory is decided by the forget gate. The output gate decides what information to output from the cell. By controlling gates, LSTM retains important information over long sequences and prevents the vanishing gradient problem.

Bidirectional LSTMs (BiLSTMs) extend the standard LSTM by processing input sequences in both forward and backward directions. This means that, at any point in time, the network has access to both past and future context, allowing it to capture more comprehensive dependencies in the data. By utilizing two LSTM layers—one reading the sequence from start to end and the other from end to start—bidirectional LSTMs improve prediction accuracy, particularly in tasks where the entire input sequence contains important information for predicting future values.

3) Gated Recurrent Unit (GRU): GRUs have a simpler architecture compared to LSTMs but still effectively handle long-range dependencies in sequences. GRUs have two main gates: the update gate and the reset gate. The update gate determines how much of the past information should be passed to the future, while the reset gate controls how much of the past information should be forgotten. GRUs retain essential information over longer time steps and help mitigate the vanishing gradient problem. Unlike LSTMs, GRUs merge the cell state and hidden state, resulting in fewer parameters and a more streamlined model.

Bidirectional GRUs (BiGRUs) function similarly to bidirectional LSTMs but with the simplified architecture of GRUs. In this configuration, two GRU layers process the input sequence: one in the forward direction and the other in the backward direction. This dual-layer approach allows the model to learn both past and future dependencies in the sequence, providing a more complete understanding of the data. The bidirectional GRU leverages its streamlined gating mechanisms to efficiently capture long-range dependencies from both directions, enhancing forecasting performance while maintaining a simpler structure compared to bidirectional LSTMs.

C. GARCH Model

GARCH(p,q) model for any return r_t is

$$r_t = \sigma_t Z_t$$
$$\sigma_t^2 = \omega + \sum_{i=0}^{p} \alpha_i r_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2,$$

where Z_t represents a sequence of random variables with zero mean and unit variance. The conditional variance σ_t is a time-varying, positive function that is measurable with respect to the information set available at time $t-1$. The parameters must satisfy $\omega > 0$, $\alpha_i \geq 0$ for $i = 1, \ldots, p$, and $\beta_j \geq 0$ for $j = 1, \ldots, q$. One-step-ahead conditional variance forecast is $\sigma_{t+1}^2 = \omega + \alpha_1 r_t^2 + \beta_1 \sigma_t^2$, and if log returns follow a student t-distribution with degrees of freedom less than four, σ_{t+1}^2 cannot be defined.

In this study, the GARCH neuro volatility (GARCH.Neuro.Volatility/GARCH.Neuro) model is set up by first obtaining conditional variance forecasts using lagged values of conditional variance ($(r_t - \bar{r})^2$) as inputs to the NNs. Thus, in Figure 1, the target series X_t is replaced by $(r_t - \bar{r})^2$ when the network is used to obtain conditional variance forecasts. Once the forecasts of conditional variance are obtained, their square root values are taken to produce the volatility forecasts. These forecasts are referred to as GARCH neuro volatility forecasts in this study, and they are obtained using only `nnetar`.

III. Experimental Results

The experimental results section of this paper focuses on volatility forecasts of the CBOE Volatility Index (VIX) and

Bitcoin (BTC) using NN models. The VIX measures the market's expectation of future volatility based on $S\&P$ 500 index options. A high VIX value typically indicates heightened market risk and potential price swings, while a low VIX suggests stability and lower perceived risk. Bitcoin is a decentralized digital currency, introduced in 2008 [15]. It functions on a peer-to-peer network, leveraging blockchain technology to facilitate secure and transparent transactions, eliminating the necessity for intermediaries such as banks. Volatility for both VIX and BTC is calculated using daily adjusted closing prices and data are obtained for the study period 2022-01-01 to 2024-06-05 (after COVID) from Yahoo! Finance. The number of observations for adjusted closing prices of VIX and Bitcoin varies due to differences in their trading schedules. The VIX, being tied to the stock market, is only available on trading days, which excludes weekends and holidays. In contrast, Bitcoin operates in a decentralized market and is traded 24/7, meaning data is available for every day, including weekends. As a result, for the same time period, there will be more data points for Bitcoin than for the VIX. All experiments were conducted on a machine equipped with an Intel Core(TM) i5-8265U processor, featuring 4 physical cores (8 logical cores) clocked at 1.60GHz, 8GB of RAM, and running the Windows 11 operating system.

TABLE I
SUMMARY STATISTICS OF LOG RETURNS

Asset	Mean	SD	Excess kurtosis	ν
VIX	-0.0004	0.0594	0.8997	9.4030
BTC	0.0005	0.0290	4.6883	3.3517

For both assets, volatility is especially high during 2022. However, occasional significantly high volatility is observed in 2023 and 2024. This highlights the non-linear and non-stationary nature of volatility for both VIX and BTC, making it appropriate to use NNs for volatility forecasting, as NNs can effectively approximate continuous non-linear functions. Table I provides summary statistics of log returns for VIX and BTC, and using the $sign$ correlation, the degrees of freedom (ν) of log returns are calculated. It is important to note that for BTC, the degrees of freedom are less than four. As a result, the GARCH model cannot be defined for BTC, and if the conditional variance is used to estimate volatility, the asymptotic variance of the volatility estimate becomes infinite. Additionally, the log returns of VIX also exhibit some kurtosis, indicating a heavy-tailed distribution.

First, five single-layer NN models, GARCH neuro volatility (GARCH.Neuro) model, data-driven neuro volatility (DD.Neuro) models, RNN, LSTM, and GRU, are trained with observed daily volatility. Table II provides the computation time (in seconds) for these single-layer neural networks to train the model using observed volatility from 2022-01-01 to 2024-05-14 and to obtain volatility forecasts from 2024-05-15 to 2024-06-05. Among the considered models, the computation time for the data-driven neuro volatility model is higher than for the GARCH neuro volatility model, RNN, LSTM, and GRU. The computation time for RNN is the shortest when obtaining volatility forecasts for VIX, while for BTC, LSTM has the shortest computation time. When comparing the computation time between VIX and BTC, except for LSTM, the computation time is higher for BTC, which can be attributed to the larger number of data points for BTC over the same period.

TABLE II
COMPUTATION TIME (SECONDS) FOR SINGLE LAYER NEURAL NETWORKS

Asset	GARCH.Neuro	DD.Neuro	RNN	LSTM	GRU
VIX	70.31	75.19	2.23	3.42	2.46
BTC	123.45	138.72	3.28	3.17	4.06

When training the single-layer NNs, the number of lags and the number of neurons in the hidden layer for the RNN, LSTM, and GRU models are set based on the lagged inputs (p) and the number of hidden neurons (k) from the data-driven neuro volatility (DD.Neuro.Volatility/DD.Neuro) model. The models are compiled using the Adam optimizer and the mean squared error (MSE) loss function. Each model is trained on the training data for 20 epochs with a batch size of 32. After obtaining the volatility forecasts, they are compared against the observed volatility for the testing period, and the forecast errors (RMSE - Root mean squared error, MAE - Mean absolute error, and MAPE - Mean absolute percentage error) are summarized in Table III. With the same number of neurons in the hidden layer and considering the same lagged inputs, except for MAPE, the forecast errors with the data-driven neuro volatility (DD.Neuro) model are smaller than the forecast errors of the GARCH neuro volatility model, RNN, LSTM, and GRU for VIX. For both VIX and BTC, considering RMSE and MAE, the forecast error is highest with the GARCH neuro volatility model, and for RNN, LSTM, and GRU, the forecast errors are close for BTC.

TABLE III
FORECAST ACCURACY OF SINGLE-LAYER NEURAL NETWORKS

Asset	Model	RMSE	MAE	MAPE
VIX	GARCH.Neuro	1.1015	0.8347	78.4829
	DD.Neuro	0.6844	0.5940	255.2584
	RNN	0.7609	0.6527	292.1085
	LSTM	0.7327	0.6469	299.7216
	GRU	0.7233	0.6424	290.7569
BTC	GARCH.Neuro	0.7243	0.4903	102.8868
	DD.Neuro	0.5667	0.4092	476.7304
	RNN	0.5391	0.3488	515.8429
	LSTM	0.5366	0.3580	543.2991
	GRU	0.5362	0.3521	533.0779

It is also important to investigate if these models can capture changes in volatility during the testing period. Figure 2 shows volatility forecasts from NNs alongside observed volatility during the testing period. For VIX, the data-driven neuro volatility model (red line) successfully predicts the decreases and increases in volatility during the first few days of the testing period. LSTM and GRU forecasts are mostly linear and fail to capture the fluctuations in volatility. RNN forecasts are non-linear but do not accurately predict the changes in volatility during the testing period. Decreases and increases in volatility for BTC during the testing period are successfully predicted by the data-driven neuro volatility model. However, forecasts from the remaining NNs are mostly linear and fail to capture the fluctuations in volatility for BTC. For both VIX and BTC, the GARCH neuro volatility model underestimates volatility and fails to predict changes in volatility.

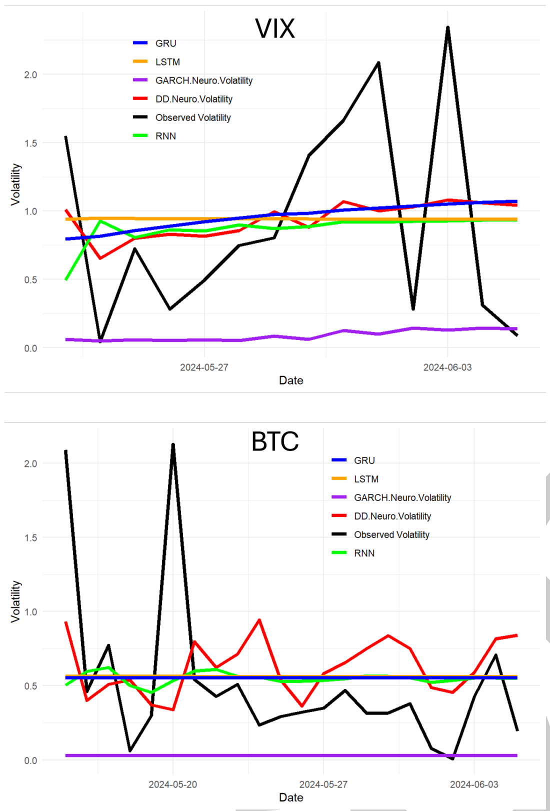

Fig. 2. Volatility Forecasts using single-layer Neural Networks (top: VIX and bottom: BTC)

Next, the study investigates the volatility forecasts for VIX and BTC using NNs with two hidden layers. Note that `nnetar` always fits a single hidden layer NN, while RNN, LSTM, and GRU can be modified to have two or more hidden layers. One additional hidden layer is introduced to the RNN, LSTM, and GRU models considered earlier, with the number of neurons in each hidden layer set equal to the number of hidden neurons (k) from the data-driven neuro volatility model. The computation times (in seconds) for these new models are given in Table IV Despite the lower computation time, the forecast accuracy of RNN has improved for VIX, though forecast errors for the data-driven neuro volatility model are still lower than those of RNN. In terms of forecast errors, RNN shows no significant change for BTC. For both VIX and BTC, the forecast errors of volatility using LSTM and GRU with two hidden layers do not show significant change.

TABLE IV
COMPUTATION TIME (SECONDS) FOR NEURAL NETWORKS WITH TWO HIDDEN LAYERS

Asset	RNN	LSTM	GRU
VIX	1.92	4.78	4.19
BTC	3.24	6.44	7.61

TABLE V
FORECAST ACCURACY OF NEURAL NETWORKS WITH TWO HIDDEN LAYERS

Asset	Model	RMSE	MAE	MAPE
VIX	GARCH.Neuro	1.1015	0.8347	78.4829
	DD.Neuro	0.6844	0.5940	255.2584
	RNN	0.7322	0.6424	272.8766
	LSTM	0.7245	0.6323	280.2030
	GRU	0.7379	0.5985	236.8305
BTC	GARCH.Neuro	0.7243	0.4903	102.8868
	DD.Neuro	0.5667	0.4092	476.7304
	RNN	0.5377	0.3006	427.4123
	LSTM	0.5396	0.3759	574.3121
	GRU	0.5383	0.3679	559.8520

With more complex NNs (two hidden layers), it is also important to investigate whether they are now capable of predicting volatility patterns during the testing period. Figure 3 shows the volatility forecasts of RNN, LSTM, and GRU with two hidden layers, along with the volatility forecasts of the GARCH neuro volatility model and the data-driven neuro volatility model. For BTC, the volatility forecasts of RNN do not show significant changes compared to Figure 2 and still underestimate the volatility. However, RNN demonstrates improvement in predicting the volatility patterns of VIX, especially during the first several days of the testing period.

More complex architectures, specifically bidirectional LSTM (BiLSTM) and bidirectional GRU (BiGRU), are considered next to obtain volatility forecasts, and BiLSTM and BiGRU are studied with both a single hidden layer and two hidden layers. The computation time for these models is provided in Table VI. Due to the increased complexity, computation time has increased compared to the single and two-hidden-layer LSTM and GRU models. However, it is important to note that the computation time for BiLSTM for VIX and BiGRU for BTC is slightly smaller compared to the LSTM and GRU models with two hidden layers. For BTC, the computation times for BiLSTM and BiGRU with two hidden layers are approximately double that of the LSTM and GRU models with two hidden layers.

TABLE VI
COMPUTATION TIME (SECONDS) FOR BIDIRECTIONAL NEURAL NETWORKS

Asset	BiLSTM	BiGRU	BiLSTM TwoLayers	BiGRU TwoLayers
VIX	4.11	4.31	11.31	14.22
BTC	8.91	7.47	13.10	12.76

The forecast accuracy of bidirectional LSTM and GRU

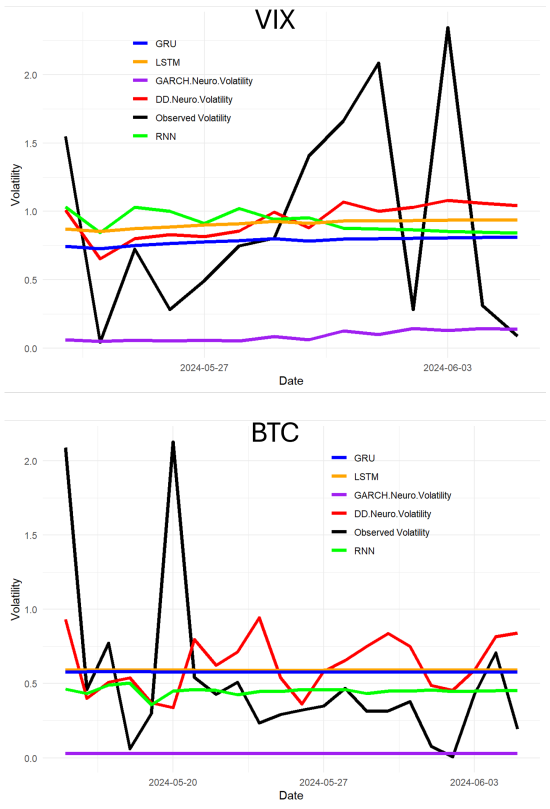

Fig. 3. Volatility Forecasts using Neural Networks with Two Hidden Layers (top: VIX and bottom: BTC)

models is provided in Table VII, and the volatility forecasts of these bidirectional models are shown in Figure 4 alongside the observed volatility for the testing period. When comparing forecast errors for VIX, the data-driven neuro volatility model produces smaller errors than the bidirectional models, and the performance of BiLSTM and BiGRU with two hidden layers is similar. For BTC, the forecast errors of the bidirectional models are very close. It is important to note that due to the complex structure of bidirectional networks, these models have a significantly higher number of parameters than the data-driven and GARCH neuro volatility models. However, the bidirectional models barely capture the volatility patterns during the testing period compared to the data-driven neuro volatility model.

IV. Conclusions

When estimating risk, the most important measure is volatility. In traditional approaches, volatility is obtained by modeling the conditional variance of log returns and then taking the square root of the conditional variance forecasts. However, this requires the assumption of normality of log returns, and it has been noted that for most financial assets, log returns tend to be non-normal. Thus, obtaining volatility forecasts using the GARCH model is inefficient. In this study, a model for directly estimating volatility is considered to obtain volatility forecasts with several NN models. Experimental results show that the data-driven neuro volatility models outperform the GARCH neuro volatility model, RNN, LSTM, and GRU. Moreover, the data-driven neuro volatility model successfully predicts changes in volatility during the testing period, while forecasts from the other models are mostly linear and fail to capture the fluctuations in volatility during an unobserved period.

TABLE VII
Forecast Accuracy of Bidirectional Neural Networks

Asset	Model	RMSE	MAE	MAPE
VIX	GARCH.Neuro	1.1015	0.8347	78.4829
	DD.Neuro	0.6844	0.5940	255.2584
	BiLSTM	0.7331	0.6501	298.9558
	BiGRU	0.7437	0.6634	316.4223
	BiLSTM.TwoLayer	0.7250	0.6409	287.6405
	BiGRU.TwoLayer	0.7298	0.6559	304.8922
BTC	GARCH.Neuro	0.7243	0.4903	102.8868
	DD.Neuro	0.5667	0.4092	476.7304
	BiLSTM	0.5324	0.3234	478.5087
	BiGRU	0.5347	0.3413	514.8504
	BiLSTM.TwoLayer	0.5362	0.3533	535.6504
	BiGRU.TwoLayer	0.5351	0.3396	509.5531

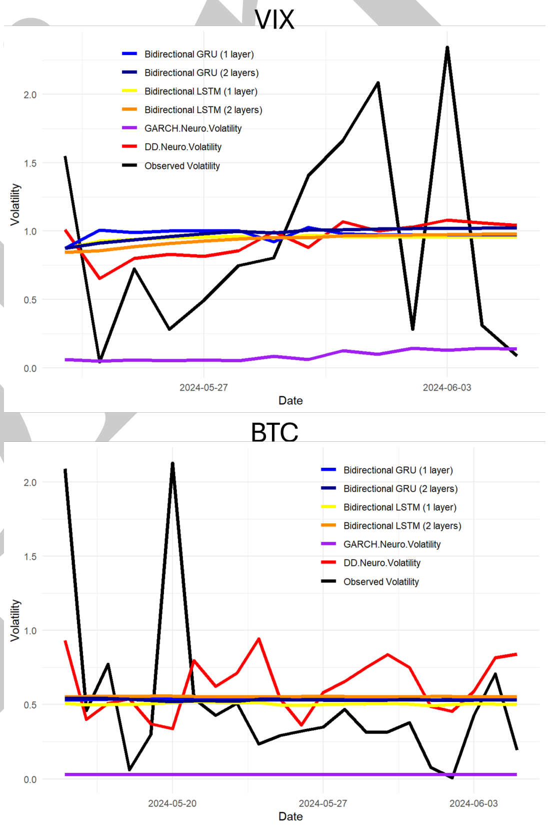

Fig. 4. Volatility Forecasts using Bidirectional Neural Networks (top: VIX and bottom: BTC)

References

[1] BaselIII. Current data collection exercises. https://www.bis.org/bcbs/publ/d570.pdf, 2024. (Accessed on 09/28/2024).

[2] Thavaneswaran, A., Paseka, A., & Frank, J. (2020). Generalized value at risk forecasting. Communications in Statistics-Theory and Methods, 49(20), 4988-4995.

[3] Engle, R. F. (1983). Estimates of the Variance of US Inflation Based upon the ARCH Model. Journal of money, credit and banking, 15(3), 286-301.

[4] Donaldson, R. G., & Kamstra, M. (1997). An artificial neural network-GARCH model for international stock return volatility. Journal of Empirical Finance, 4(1), 17-46.

[5] Hamid, S. A., & Iqbal, Z. (2004). Using neural networks for forecasting volatility of S&P 500 Index futures prices. Journal of Business Research, 57(10), 1116-1125.

[6] Aragonés, J. R., Blanco, C., & Estévez, P. G. (2007). Neural network volatility forecasts. Intelligent Systems in Accounting, Finance & Management: International Journal, 15(3-4), 107-121.

[7] Wang, C. P., Lin, S. H., Huang, H. H., & Wu, P. C. (2012). Using neural network for forecasting TXO price under different volatility models. Expert Systems with Applications, 39(5), 5025-5032.

[8] Monfared, S. A., & Enke, D. (2014). Volatility forecasting using a hybrid GJR-GARCH neural network model. Procedia Computer Science, 36, 246-253.

[9] Kristjanpoller, W., Fadic, A., & Minutolo, M. C. (2014). Volatility forecast using hybrid neural network models. Expert Systems with Applications, 41(5), 2437-2442.

[10] Kristjanpoller, W., & Minutolo, M. C. (2016). Forecasting volatility of oil price using an artificial neural network-GARCH model. Expert Systems with Applications, 65, 233-241.

[11] Kristjanpoller, W., & Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. Expert Systems with Applications, 109, 1-11.

[12] Liu, Y. (2019). Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. Expert Systems with Applications, 132, 99-109.

[13] Hoque, M. E., Bowala, S., Saumyamala, A., Thavaneswaran, A., & Thulasiram, R. (2024, July). Novel Resilient Model Risk Forecasts based on Neuro Volatility Models. In 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 210-219). IEEE.

[14] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts..

[15] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Satoshi Nakamoto.