ISSN 1870-4069

Role of Sparse Training and Evolutionary Optimization in Volatility Forecasting Models

Juan Francisco Muñoz-Elguezabal, Diego F. Arriaza-Alonzo

Instituto Tecnológico y de Estudios Superiores de Occidente,
Departamento de Matemáticas y Física, Tlaquepaque,
Mexico

{francisco@iteso.mx, diego.arriaza}@iteso.mx

Abstract. ETH is the native cryptocurrency of the Ethereum network, renowned for its smart contracts and diverse decentralized ecosystem. This research addresses the challenge of short-term volatility forecasting of ETH/USDT on a 10-minute interval, leveraging order book data and public trade data from the previous 30 minutes. Order book data includes buy and sell orders over time, while public trades refer to executed orders. Features derived from these data sources are used as model predictors. The first experiment tested the average past volatility, GARCH(1,1), LSTM, and an Auto-Encoder using a single training, validation, and test set. The second experiment applied the same models within a Walk-Forward architecture. The third experiment utilized a Time Fold Sequential Validation (T-Folds SV) technique, creating 10 folds and omitting 50 minutes between training and validation sets to prevent leakage. By calculating Kullback-Leibler Divergence, 5 folds were selected that have the characteristics of being different from each other and provide unique information. As a consequence, RAM consumption was significantly reduced while maintaining comparable results to previous experiments. Hyperparameter optimization with less data is now possible and is performed by Genetic Algorithms. After three generations of 750 models for both LSTM and Auto-Encoder, the best hyperparameter values were found, with the optimized LSTM model outperforming its counterpart. An ablation study as the last experiment was analyzed by removing the early stopping criteria of the best models, resulting in worse performance, but not significantly.

Keywords: Ether, order book, public trades, LSTM, auto-encoder, T-Folds SV, Kullback-Leibler divergence, genetic algorithms.

1 Introduction

Ethereum is a decentralized blockchain that was first introduced in 2013 by Vitalik Buterin, a computer programmer and researcher in cryptocurrency. It is a network that acts as the foundation for applications, organizations, communities and digital assets that anyone can create and utilize. Its main feature is its smart contract functionality, which is also the main difference between Bitcoin and Ethereum.

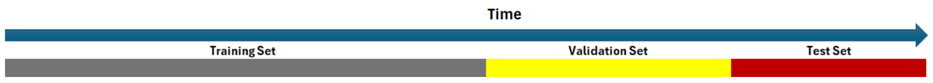

Fig. 1. One window training-validation-test setup.

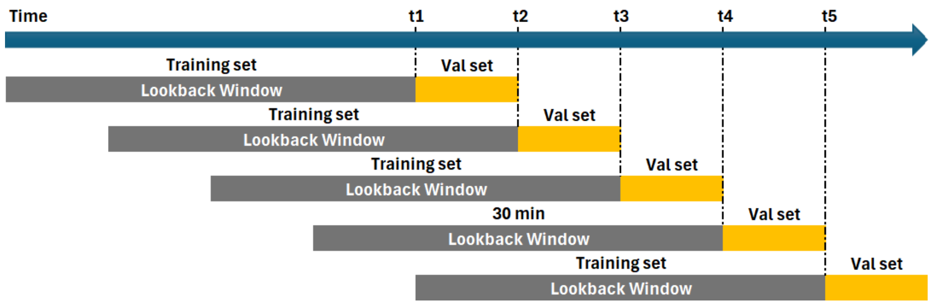

Fig. 2. Walk forward setup.

Smart contracts are computer programs that are executed when triggered by a transaction and act as building blocks for decentralized applications and organizations. Once a smart contract is published, it will be online and operational until Ethereum no longer exists. Examples of smart contracts are lending applications, insurance, decentralized trading exchanges, NFTs, etc. Ether is the native cryptocurrency of Ethereum. For ETH payments or use of Ethereum applications, a fee in ETH is charged, which is an incentive for a block producer to perform processing and verifications. Its main characteristics are that it is secured by cryptography, there is no intermediary service to make payments (it is peer-to-peer), there is no institution that can decide to print more ETH, or change the terms of use and it is divisible up to 18 decimal places, so it is not mandatory to buy 1 whole ETH.

ETH can also be exchanged for other currencies, products and services. The exchange of ETH with currencies is done on exchange services, where buy and sell orders are stored on the order book. Buy or bid orders represent an intention to buy a certain amount of ETH at some specified price while sell or ask orders represent the opposite. The exchange is done by matching orders by price from the order book into a trade transaction between buyers and sellers. Orders are one of the key components to understand the intention of the market as well as their influence on volatility. The cryptocurrency has gained relevance over the years and today is the second cryptocurrency with the highest market cap.

Due to its importance on the cryptocurrency market and its role on the Ethereum network, where at the end of year 2023 there are 96M accounts with ETH, 53.3M smart contracts, 410B value secured and 4k projects built, understanding and modeling ETH volatility is crucial for risk management and decision making. Therefore, the purpose of this work is to predict the short-term n-minutes volatility of ETH/USDT from the Binance Exchange with Order Book and Public Trades high frequency data. This work has no intention to produce financial models of volatility or the order book dynamics itself. The main contributions are to demonstrate the effectiveness and relevance of three crucial characteristics for Volatility Forecasting:

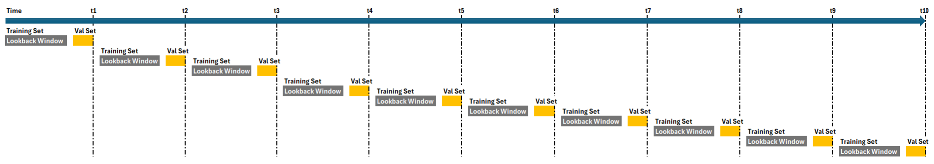

Fig. 3. T-folds SV.

The use of Order Books and Public Trades as source of information, the sparse training using T-Folds SV and KL Divergence for LSTM and Auto-Encoder models and Genetic Algorithms for Optimization method. The remainder of the paper is structured as follows. In section 2 an overview of the related work is presented. In section 3 information on how the data was collected and structured is provided as well as how the features of Order Book and Public Trades were calculated.

Theoretical explanation on how the models are formulated and their main properties is provided and evaluations metrics are also approached. In section 4details on how the models were setup in different experiments are given. Finally in section 5 and section 6 the results of the experiments are reported and discussed and conclusions on their performance are detailed.

2 Background and Related Work

The volatility estimation is a topic that has been extensively studied and developed over the years. In the specific case of cryptocurrencies, papers have focused on modeling BTC volatility using GARCH models. Vivian Naimy and Marianne Hayek compared the predictive ability of three GARCH models: GARCH (1,1), EWMA, and EGARCH (1,1). The results indicate that the asymmetric EGARCH (1,1) model outperforms the symmetric GARCH (1,1) and EWMA models in both in-sample and out-of-sample contexts. This suggests that BTC behavior differs from traditional currencies [9].

This paper, however, will only focus on GARCH to have a baseline model and it will be compared to only averaging the past volatility to measure its robustness. BTC will not be examined because it is the preferred cryptocurrency for analysis and not many papers focus on others such as Ether. On the survey made by Charandabi and Kamyar [2], they highlighted two papers.

The first one is the work of Miura, Pichl and Kaizoji called Artificial Neural Networks for Realized Volatility Prediction in Cryptocurrency Time Series [7], in which they aggregated RV values using 1-minute-sampled Bitcoin returns over 3-h intervals to predict future values using ANN (MLP, GRU, LSTM), SVM, and Ridge Regression and compare their results with Heterogeneous Auto-Regressive Realized Volatility (HARRV) model with optimized lag parameters.

Ridge Regression was able to outperform all the models. The other highlighted work was from Jang and Lee, in which they compared the Bayesian neural network with benchmark models, such as Linear Regression and SVR, on modeling and predicting the Bitcoin pricing process and concluded that the BNN outperformed the others. [4] Charandabi and Kaymer also pointed out the need for further research in areas like implementing data from less volatile cryptocurrencies and exploring new models.

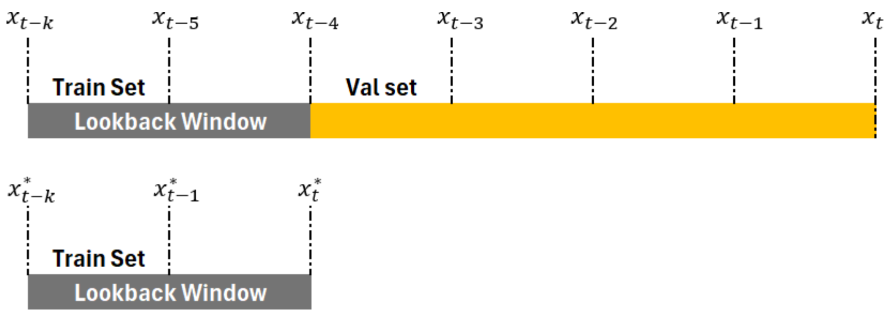

Fig. 4. Information leakage. X_{t-4}, X_{t-5} and X_{t-k} are input variables in the validation set, but they are the same inputs in the training set represented as X^*_t, X^*_{t-1} and X^*_{t-k} respectively.

As stated above, there have been papers that have worked with Deep Learning Models to forecast Time Series and Volatility. German Rodikov and Nino Antulov-Fantulin approached the challenge of predicting volatility by trying the LSTM model effectiveness against EWMA, HAR-RV, ARIMA, GARCH and GJR-GARCH. According to their results, LSTM outperformed all models in a rolling window between 5 and 12 periods. [10]. This work will also approach to work with the LSTM model, so it can be used to forecast 10 periods, specifically 10 minutes and it will also be compared to the baseline model GARCH.

Jung and Choi in their work called Forecasting Foreign Exchange Volatility Using Deep Learning Autoencoder-LSTM Techniques [5] predicted the FX volatility with a hybrid model that combined long short-term memory (LSTM) and autoencoder models, in which an LSTM model is utilized as an encoder and decoder inside an autoencoder network. They compared this hybrid model with the traditional LSTM model and based on their empirical results, the hybrid model outperformed the LSTM model. This work will also run an Auto-Encoder model similar to work by Jung and Choi because it will use LSTM layers on both the encoder and decoder and will also compare the Auto-Encoder model with the LSTM to see if it also outperforms it or not.

Guo, Bifet and Antulov-Fantulin point out that Order Book information cryptocurrency forecasting is still under-researched. Therefore, their proposal was to implement a temporal mixture model capable of adaptively exploit both volatility history and order book features. To forecast the volatility, they used a rolling strategy and divided the range of data into non-overlapping intervals, with each interval corresponding to one month. This paper is also contributing to forecast cryptocurrency volatility using Order Book information by modeling GARCH and LSTM. For training and validation Guo, Bifet and Antulov-Fantulin used two rolling strategies. The first one consists of training two months and validate with the immediate next month.

Then the training set moves to the next month as well as the validation set and the current training set is now taking in consideration the month used by the previous validation set for training. The second is similar, but instead of rolling both training set and validation set one month, the training set increase its periods by one month and only the validation set moves to next month.[3]. Regarding the training and validation of this paper, a Walk-Forward architecture will be applied as well as an approach similar to rolling strategy used by the three previous mentioned papers.

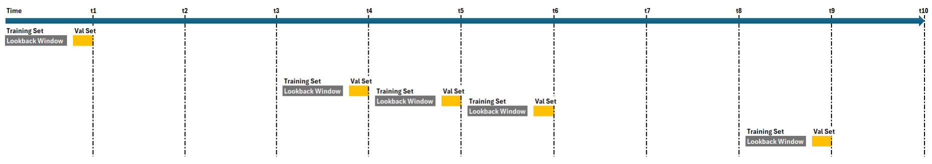

Fig. 5. KL Divergence T-fold SV.

The main difference of the last mentioned approach is that training sets will never be retrained on previous validation sets and there will be a purge of the observations of the training set that overlap with the validation set to avoid information leakage, which is highlighted by Marco Lopez de Prado[6] on his book Advances in Financial Machine Learning and is applied by Juan Francisco Muñoz and Juan Diego Sanchez on their poster T-Fold Sequential Validation Technique for Out-Of-Distribution Generalization with Financial Time Series Data[8].

3 Data and Methods

In this section, it is described how the data was collected and its source of information as well as calculations of Order Book and Public Trades features, Return and Volatility. Lastly, a theoretical framework addresses the Kullback-Leibler Divergence and the method used to optimize the hyperparameters, which is the Genetic Algorithm process.

3.1 Data Collection

There are two sources of information: Order Book Snapshots and Public Trades. Both were obtained by the website Tardis.dev. For every day, there were about 750-700K samples for Order Book data and around 500K samples for Public Trades data and both were from the exchange Binance only. The frequency of order books data was of 100 milliseconds and microseconds for Public Trades. The raw Order Book data provided the top 25 asks and bid prices as well as the top 25 asks and bids amounts. As for the raw Public Trades data, it displayed the trade price, the trade amount and the liquidity taker, which were only two possible values: buy or sell.

3.2 Order Books

The Order Book is a list of buy and sell orders that are waiting to be traded. All the features of an Order Book can be calculated on different levels (depth of orders) and can be uses as input variables for volatility modeling. As an example, it is possible to calculate the Volume Weighted Average Price (VWAP) of the first 5 levels, the first 10 levels or only the first level called Top of the Book (TOB). These are some of the following features equations:

$$\text{VWAP} = \frac{\sum_{i=1}^{n} \text{bid price}_i \times \text{bid volume}_i + \sum_{i=1}^{n} \text{ask price}_i \times \text{ask volume}_i}{\sum_{i=1}^{n} \text{bid volume}_i + \sum_{i=1}^{n} \text{ask volume}_i}, \quad (1)$$

Table 1. Hyperparameter values.

Hyperparameters	**Values**
Epochs	[50,100,200]
Learning Rate	[0.0001, 0.001, 0.01, 0.05, 0.1]
Batch Sizes	[32, 64, 128, 256, 512]
Dropout	[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]
Optimizer	[RMSprop, SGD, Adam, Nadam, Adadelta]
Loss function	[MAE, MSE, HUBER, LOGCOSH]
Layers	[1-3]
Unit in Layers	[50, 100, 200]

$$\text{Total Depth} = \sum_{i=1}^{n} \left(\text{bid volume}_i + \text{ask volume}_i\right), \tag{2}$$

$$\text{Imbalance} = \sum_{i=1}^{n} \frac{\text{bid volume}_i}{\text{bid volume}_i + \text{ask volume}_i}, \tag{3}$$

$$\text{Total Spread} = \sum_{i=1}^{n} \frac{\text{Depth}_i}{\text{Total Depth}} \times \left(\text{price ask}_i - \text{price bid}_i\right), \tag{4}$$

$$\text{Spread Volume} = \sum_{i=1}^{n} \left(\text{ask volume}_i - \text{bid volume}_i\right), \tag{5}$$

$$\text{Midprice} = \frac{\text{ask}_{\text{TOB}} + \text{bid}_{\text{TOB}}}{2}. \tag{6}$$

It is important to highlight that the equation 4 was used by Alexander Aidov and Olesya Lobanova on their research [1] to prove that it exists an inverse relation between the limit order book depth and spread.

3.3 Public Trades

The information of Public Trades was resampled into a 1 minute timeframe to capture the information of the prices dynamics during a specific time interval. This allowed to capture the first price (Open), maximum price (High), minimum price (Low), last price (Close) and the accumulated volume, which creates a data consolidation that is referred commonly as OHLCV.

OHLCV Features. Features of differences and relative differences between open, close, high and low prices are calculated to show the price behavior. Some examples are High_t - Low_t, Open_t - Low_t, Close_t - Open_t and High_t - Open_t. A calculation of price percentage differences between the current price and the previous price also provides information on price movements. As an example it is possible to calculate the percentage difference between the high price and the previous high price.

Table 2. One window experiment results.

One Window Results												
	Training				**Validation**				**Test**			
	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
Mean	0	1.18E-07	3.44E-04	1.95E-04	0	9.91E-08	3.15E-04	1.78E-04	-0.0147	8.89E-08	2.98E-04	1.57E-04
GARCH	0.4981	5.93E-08	2.44E-04	1.51E-04	0.3715	6.23E-08	2.50E-04	1.70E-04	0.3536	5.67E-08	2.38E-04	1.73E-04
LSTM	0.9611	4.59E-09	6.78E-05	3.32E-05	0.7878	2.10E-08	1.45E-04	4.63E-05	0.8373	1.43E-08	1.20E-04	5.38E-05
Auto-Encoder	0.9083	1.08E-08	1.04E-04	4.85E-05	0.8655	1.33E-08	1.15E-04	4.33E-05	0.898	8.96E-09	9.47E-05	4.73E-05

The previous mentioned features can also be auto-regressive and take into consideration the features t-k periods. As an example, it is possible to obtain $\{\text{High}_t - \text{Low}_t\}_{t-k}$, $\{\text{Open}_t - \text{Low}_t\}_{t-k}$, $\{\text{Close}_t - \text{Open}_t\}_{t-k}$ and $\{\text{High}_t - \text{Open}_t\}_{t-k}$ for values of $k = 1,\ 2,\ \ldots K$ with K as proposed memory parameter. Then it is possible to perform some operations like Simple Moving Average SMA_t, lag LAG_t and Standard Deviation SD_t. Lastly, for the volume features, buy volume and sell volume are calculated multiplying the numbers of sides (buy taker or sell taker) with the volume traded. All these features can be used as input variables for volatility modeling.

Return, Log Return and Volatility. Return and Log Return are a comparison between the current price and the previous price:

$$\text{return}_t = \frac{\text{price}_t}{\text{price}_{t-1}} - 1, \tag{7}$$

$$\log \text{return}_t = \log \frac{\text{price}_t}{\text{price}_{t-1}}. \tag{8}$$

Volatility is the measure of price fluctuations during a certain time and it is obtained calculating the standard deviation of returns or log returns. For the purpose of this work, log returns and 30 periods were selected to calculate the volatility:

$$\text{Volatility}_{t=0:30} = \sqrt{\frac{1}{30}\sum_{i=1}^{30}(r_i - \bar{r})^2}. \tag{9}$$

3.4 Kullback-Leibler Divergence

It is a measure of the difference between two probability distributions. It is commonly used in information theory and statistics to quantify how much a reference probability distribution (denoted P) differs from another probability distribution (denoted Q). For discrete distribution refer to the formula 10 and for continuous distribution refer to the formula 11:

$$D_{KL}(P||Q) = \sum_{x \epsilon \chi} P(x) \log\left(\frac{P(x)}{Q(x)}\right), \tag{10}$$

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx. \tag{11}$$

Table 3. Mean walk-forward results.

Mean Walk-Forward Results												
	Training				**Validation**				**Test**			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.0000	8.30E-08	2.88E-04	1.87E-04	-0.6877	5.22E-08	2.28E-04	2.01E-04	-0.0086	8.84E-08	2.97E-04	1.58E-04
1	0.0000	7.23E-08	2.69E-04	1.70E-04	-1.8301	5.60E-08	2.37E-04	2.13E-04				
2	0.0000	5.50E-08	2.35E-04	1.52E-04	-0.0725	4.94E-07	7.03E-04	2.51E-04				
3	0.0000	1.26E-07	3.54E-04	1.73E-04	-0.0308	1.05E-07	3.25E-04	1.86E-04				
4	0.0000	1.25E-07	3.54E-04	1.70E-04	-0.0196	6.14E-08	2.48E-04	1.56E-04				
5	0.0000	1.27E-07	3.57E-04	1.75E-04	-0.0405	1.22E-07	3.49E-04	2.00E-04				
6	0.0000	1.40E-07	3.74E-04	1.92E-04	-0.0181	4.65E-08	2.16E-04	1.45E-04				
7	0.0000	1.42E-07	3.76E-04	1.95E-04	-0.0166	4.95E-08	2.23E-04	1.56E-04				
8	0.0000	1.41E-07	3.75E-04	1.94E-04	-0.0202	2.52E-07	5.02E-04	2.09E-04				
9	0.0000	1.07E-07	3.28E-04	1.89E-04	-0.0022	7.15E-08	2.67E-04	1.71E-04				

The KL Divergence has the property that is non-negative $D_{KL}(P||Q) \geq 0$ and when $D_{KL}(P||Q) = 0$ it means that the distribution of P and Q is the same. Another property is its asymmetry, which means that $D_{KL}(P||Q)! = D_{KL}(Q||P)$ For the purpose of this work it will be used to understand the similarity of the distributions of the volatility across 10 T-Fold-SV. If one of the folds the distribution shows a large dissimilarity with another fold distribution, it may indicate that at least one of the folds contains unique information that the other one does not have. Not necessarily both of them will contain unique information because of the asymmetry of Kullback-Leiber Divergence.

On the contrary, there will be also cases where the two folds will show a large similarity, which may indicate that the volatility is behaving with no major changes between one fold and the other. Consequently, training both folds will not necessarily feed the model with new information to be trained. On Time Series it is common to see periods with stability and in contrast, periods with spikes and abrupt changes. The contrast between both scenarios may have patterns behind that the models can capture and therefore predict when one of the scenarios is going to happen.

3.5 Genetic Algorithms

They are an optimization method inspired by natural evolution. They create and evolve a population of possible solutions to a problem. Each solution is represented as an individual of the population and these individuals evolve through time by crossovers and mutations. The population is generated randomly or by an heuristic approach and then each individual is evaluated by a fitness function, which assigns a numeric value to the individual solution and determines its ability to survive. The best individuals with higher fitness are more likely to be selected as parents for the creation of children through genetic operators. These children will represent the next population.

One of the genetic operators involves a crossover between the parents, in which a probability threshold will determine whether the parents properties are exchanged between each other or not. The other genetic operator is a mutation in which, through a probability threshold, one property is exchanged for another property. Over generations, individuals with higher fitness are more likely to reproduce, thus passing on their characteristics to subsequent generations.

Table 4. GARCH walk-forward results.

GARCH Walk-Forward Results												
	Training				**Validation**				**Test**			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	-20.1258	1.75E-06	1.32E-03	9.42E-04	-0.1457	3.54E-08	1.88E-04	9.63E-05	0.8231	1.55E-08	1.25E-04	7.94E-05
1	-10.4023	8.25E-07	9.08E-04	4.90E-04	-3.7275	9.35E-08	3.06E-04	7.51E-05				
2	-7.2526	4.54E-07	6.74E-04	3.54E-04	0.3128	3.16E-07	5.62E-04	2.28E-04				
3	0.5201	6.03E-08	2.46E-04	1.31E-04	0.4004	6.13E-08	2.48E-04	1.67E-04				
4	0.5463	5.69E-08	2.39E-04	1.22E-04	0.3489	3.92E-08	1.98E-04	1.47E-04				
5	0.5145	6.19E-08	2.49E-04	1.37E-04	0.3804	7.26E-08	2.69E-04	1.79E-04				
6	0.5475	6.34E-08	2.52E-04	1.39E-04	0.0775	4.22E-08	2.05E-04	1.56E-04				
7	0.5986	5.68E-08	2.38E-04	1.24E-04	0.3215	3.31E-08	1.82E-04	1.27E-04				
8	0.6564	4.84E-08	2.20E-04	1.00E-04	0.7425	6.37E-08	2.52E-04	1.33E-04				
9	0.3936	6.51E-08	2.55E-04	1.06E-04	0.5586	3.15E-08	1.77E-04	1.05E-04				

This process of natural selection and reproduction leads to convergence towards optimal or suboptimal solutions to the problem.

4 Experiments

4.1 One Window Experiment

For a reference on how well other models perform an average of the volatility is calculated on the training set. For a baseline model GARCH(1,1) with constant mean and Standardized Skew Student's t Distribution is proposed. Regarding the Deep Learning models, the LSTM model is built with one hidden layer of 100 neurons and regarding the Auto-Encoder structure, one layer of LSTM with 100 neurons are placed for the Encoder and one layer of LSTM with 100 neurons are placed for the Decoder.

For both models the MSE is the loss function and the metrics are MSE and MAE for monitoring. The optimizer used for training is Adam, the batch size is 675, the learning rate is 0.001 and a batch normalization is applied. An early stopping criteria of 30 epochs is applied in case the MAE validation does not improve during convergence. For this first experiment, a classic architecture of only one Training window of 60 percent of the dataset, one Validation window of 30 percent of the dataset and one Test window of 10 percent of the dataset will be configured as it is shown in the Figure 1.

4.2 Walk-forward with Sliding Window Experiment

For this experiment, a classical Training-Validation architecture called Walk-Forward Validation with a sliding window is applied. It starts with an initial training set of a fixed number of timesteps and then tests the model on the subsequent fixed-size validation set. The training set is then shifted forward by the same number of timesteps as the validation set, and this process continues until the model has been validated on the final validation set. On the Figure 2 can be found a visual representation of this training technique. On this experiment LSTM and Auto-Encoder structure setup for loss function, batch size, metrics and early stopping epoch criteria remain the same as in the first experiment.

Table 5. LSTM walk-forward results.

LSTM Walk-Forward Results												
	Training				**Validation**				**Test**			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.8530	1.28E-08	1.13E-04	6.87E-05	0.7891	4.96E-09	7.04E-05	4.69E-05	0.8661	1.18E-08	1.08E-04	4.37E-05
1	0.8484	1.13E-08	1.06E-04	6.31E-05	0.7349	4.03E-09	6.35E-05	4.80E-05				
2	0.8346	8.62E-09	9.29E-05	5.74E-05	0.9368	2.89E-08	1.70E-04	5.19E-05				
3	0.9032	1.17E-08	1.08E-04	5.53E-05	0.8985	9.32E-09	9.66E-05	3.54E-05				
4	0.9049	1.12E-08	1.06E-04	4.88E-05	0.9301	3.81E-09	6.17E-05	3.25E-05				
5	0.9170	9.86E-09	9.93E-05	4.45E-05	0.8911	1.23E-08	1.11E-04	4.19E-05				
6	0.9215	1.04E-08	1.02E-04	4.29E-05	0.9266	3.13E-09	5.60E-05	3.16E-05				
7	0.9243	1.02E-08	1.01E-04	4.04E-05	0.9062	4.50E-09	6.71E-05	3.17E-05				
8	0.9246	1.02E-08	1.01E-04	3.77E-05	0.5539	5.58E-08	2.36E-04	5.52E-05				
9	0.8202	1.48E-08	1.22E-04	3.84E-05	0.8773	8.06E-09	8.98E-05	3.90E-05				

The main differences are the input sequence and the learning rate. Feature selection is applied, in which 156 from 207 feature variables are removed. The reason behind this action was because RAM memory was not sufficient and many input variables were correlated between each other. Therefore, the dimension of the input sequence will be impacted. Regarding the learning rate, it will be an initial value of 0.001 that will decay exponentially with a factor of 0.96 every 1000 steps. Regarding the GARCH model, its structure remains the same and as for the Mean exercise model, one mean will be calculated for each training window. For Test Set, the models will be trained on a Training Set that includes the last Validation Set to keep emulating the sliding window.

4.3 T-Fold SV Experiment

For this experiment, only LSTM and Auto-Encoder models will be conducted because they are the most time-consuming and resource-intensive. They will maintain the same criteria for their setup that was used on the Walk-Forward Experiment. Both also have many hyperparameters that need to be optimized and therefore, many models need to be run to find the best values. This process can be lengthy if the resources are limited and if the time required for a model to be trained takes a lot. GARCH will not be evaluated on this experiment due to the lack of hyperparameters compared to the previous mentioned models and its poor performance on its results obtained in the One-Window Experiment and Walk-Forward Experiment, as shown in tables 2 and 4 respectively.

For the case of the Mean exercise model, it is not required to optimize hyperparameters and was only used as reference for the previous experiments. LSTM and Auto-Encoder structure setup for loss function, batch size, metrics and early stopping epoch criteria remain the same as in the Walk-Forward experiment. exponentially with a factor of 0.96 every 1000 steps. It is important to highlight that information leakage between training set and validation set occurs when the last is immediately after the first and this is due to the serial correlation of $X_t \approx X_{t+1}$ and $Y_t \approx Y_{t+1}$ [6]. To address this issue, 50 minutes were omitted between the training set and the validation set, as well as for the validation set and the next training set. The criteria to choose 50 minutes was because the input sequence models is 30 minutes and the output sequence was 10 minutes.

Table 6. Auto-encoder walk-forward results.

Auto-Encoder Walk-Forward Results												
	Training				**Validation**				**Test**			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.8545	9.63E-09	9.81E-05	5.08E-05	0.8149	3.84E-09	6.20E-05	3.19E-05	0.8858	1.00E-08	1.00E-04	4.95E-05
1	0.8511	8.49E-09	9.21E-05	4.46E-05	0.8706	1.59E-09	3.99E-05	2.57E-05				
2	0.8539	5.88E-09	7.67E-05	3.82E-05	0.9317	2.81E-08	1.67E-04	5.60E-05				
3	0.9094	9.56E-09	9.78E-05	4.00E-05	0.8917	1.05E-08	1.03E-04	4.52E-05				
4	0.9103	9.83E-09	9.92E-05	3.85E-05	0.9004	5.36E-09	7.32E-05	4.17E-05				
5	0.9184	9.19E-09	9.59E-05	3.87E-05	0.8495	1.42E-08	1.19E-04	4.96E-05				
6	0.9157	1.04E-08	1.02E-04	4.15E-05	0.8519	5.99E-09	7.74E-05	4.27E-05				
7	0.9126	1.09E-08	1.05E-04	4.35E-05	0.8782	4.87E-09	6.98E-05	3.70E-05				
8	0.9082	1.14E-08	1.07E-04	4.50E-05	0.8260	3.28E-08	1.81E-04	5.61E-05				
9	0.8601	1.24E-08	1.11E-04	4.59E-05	0.8353	1.06E-08	1.03E-04	4.83E-05				

With 50 minutes it is safe to determine that there will not be any leakage, although the purge of minutes should be at least 41 minutes. For this case more minutes were added for some slack. As it is mentioned in subsection 3.3, there are also auto-regressive features calculated from Public Trades features, which may cause that they are used in both the Training set and the Validation set. A representation of this information leakage problem can be found in the Figure 4.

Juan Francisco Muñoz and Juan Diego Sánchez addresses on their poster called T-Fold Sequential Validation Technique for Out-Of-Distribution Generalization with Financial Time Series Data the complexities of using cross-validation for financial time series data, which possess temporal structures that violate the assumption of independence and identical distribution inherent in traditional CV methods.

Their proposed method called Time Fold Sequential Validation Technique (T-Fold SV) mitigates the issues of information leakage and the masking of non-deterministic relationships between features and target variables by decomposing the global probability distribution into local distributions. This allows for identifying each sample's contribution to the learning process and maintaining information sparsity. By controlling these factors, the method relaxes the stringent i.i.d. assumption, thereby enhancing the parametric stability and accuracy of predictive models. [8].

For this experiment the T-Fold SV is used to divide validation sets and training sets into 10 T-Folds SV. This paper does not pretend to deep dive on the technique, it only limits to its usage as tool. The Figure 3 illustrates how the T-Folds SV were split showing that Training sets never used Validation sets for training. This is a key difference from the Walk-Forward technique used in the experiment of subsection 4.2. Ensuring that Training sets are never used in Validation sets reinforces the avoidance of information leakage.

To make the training process more efficient, KL Divergence is applied to the 10 T-Folds SV of volatility to select only the folds that were above of a divergence threshold. This also part of the technique of T-Folds SV. An information matrix 12 was developed containing each of the KL Divergence of the 10 volatility distributions and the logic behind is to train only folds that provide unique information with the expectation of achieving similar or better results than training the whole dataset.

Table 7. LSTM T-Fold SV results.

LSTM T-Fold SV Results												
	Training				**Validation**				**Test**			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.69018	1.67E-08	1.29E-04	9.55E-05	0.5204	5.90E-08	2.43E-04	1.38E-04				
3	0.14589	1.65E-08	1.28E-04	9.85E-05	0.5084	1.02E-08	1.01E-04	7.71E-05				
4	0.1805	1.25E-08	1.12E-04	9.55E-05	0.2762	1.54E-08	1.24E-04	9.99E-05	0.8344	1.45E-08	1.21E-04	5.24E-05
5	0.95844	1.28E-08	1.13E-04	5.49E-05	0.9433	3.40E-09	5.83E-05	3.25E-05				
8	0.90746	3.99E-09	6.31E-05	3.56E-05	0.8336	8.54E-09	9.24E-05	5.39E-05				

$$\begin{bmatrix} D_{KL}(1,1) & D_{KL}(1,2) & \dots & D_{KL}(1,\ 10) \\ D_{KL}(2,\ 1) & \cdots & \cdots & \vdots \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ D_{KL}(10,\ 1) & \cdots & \cdots & D_{KL}(10,\ 10) \end{bmatrix}. \tag{12}$$

The threshold to determine whether a distribution was not similar to another distribution was > 1. After this calculation the folds 1, 3, 4, 5 and 8 were the only folds selected for Training. 3,4 and 5 were the distributions with higher dissimilarity among other distributions. 0 and 8 folds have less dissimilarity among others distributions, but were also chosen to also consider folds from the beginning and from the end of the time series. The illustration of the final folds can be referred to the Figure 5.

4.4 Evolutionary Algorithms with T-Fold SV Experiment

In this experiment the hyperparameters to be evaluated on the Genetic Algorithms are shown in Table 1. Both LSTM and Auto-Encoder will follow the same Genetic Algorithm setup. The phases are the following: First population, parent selection, cross-over, mutation, and new population generation.

– **First Population:** Hyperparameters for each model will be randomly selected using uniform probability. There will be a total of 750 models evaluated using MSE metric.
– **Parameter Selection:** After evaluating all 750 models, parents are selected based on their results. Selection is weighted according to model performance, favoring models with better results.
– **Cross-Over:** In each iteration, two parents are randomly selected. They exchange hyperparameters to generate new children. If a random number is below the threshold of 0.9, then the exchange happens, otherwise, the children will be the same as the parents. The number of exchanged hyperparameters is also randomized1.
– **Mutation:** It occurs if a random number is below a threshold of 0.1. If triggered, a hyperparameter value is randomly exchanged within its range.

Table 8. Auto-encoder T-Fold SV results.

Auto-Encoder T-Fold SV Results												
	Training				Validation				Test			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.7848	1.18E-08	1.09E-04	7.05E-05	0.8533	3.64E-08	1.91E-04	1.02E-04	0.8730	1.12E-08	1.06E-04	5.24E-05
3	0.7491	6.29E-09	7.93E-05	3.44E-05	0.804	5.39E-09	7.34E-05	3.94E-05				
4	0.879	1.66E-09	4.08E-05	2.54E-05	0.9116	2.73E-09	5.22E-05	2.81E-05				
5	0.8593	4.71E-08	2.17E-05	6.16E-05	0.929	3.27E-09	5.72E-05	3.41E-05				
8	0.8971	3.91E-09	6.25E-05	3.67E-05	0.8673	6.96E-09	8.34E-05	4.88E-05				

Table 9. Computational resources results.

Computational Resources				
Experiment	**Model**	**RAM**	**RAM GPU**	**Run Time**
One-Window	**LSTM**	**11.8 GB**	**4.1 GB**	**5:36 min**
	Auto-Encoder	14.2 GB	4.1 GB	4:02 min
Walk-Forward	LSTM	14.5 GB	2.1 GB	12:48 min
	Auto-Encoder	15.3 GB	2.1 GB	34:03 min
T-Fold SV	LSTM	5.4 GB	0.6 GB	1.59 min
	Auto-Encoder	5.4 GB	0.6 GB	3.04 min

– **New Population:** These processes are iterated until the desired number of models for the next population is generated. Three populations of 750 individuals each will be created.

4.5 Ablation Studies

In this work it is also investigated the effects of removing early stopping for LSTM and Auto-Encoder models. This criteria is implemented to stop running iterations during Training when Validation results stop improving or start getting worse. This removal will be applied on the models with their optimized hyperparameters.

5 Results

5.1 Evolutionary Algorithms with T-Fold SV Results

On this subsection the tables with the top 10 best results of the third generation will be displayed and then the results of the final model for both LSTM and Auto-Encoder will be shown. Results of the tables with the 10 bests results are scaled and use MSE as error metric and the results of the final models with the optimal hyperparameters are with their original scale. The results of the LSTM of the third generation can be referred to the table 10. Given the results, the final model will run with 100 epochs, although it is not clear, which value is the optimal value. As for the batch size 64 will be the chosen value.

Table 10. LSTM top 10 third generation results.

Epochs	Batch	Lr Rate	Dropout	Optimizer	Loss	No. Layers	Units by Layer	Results
100	64	0.01	0.5	nadam	mae	1	[200, 0, 0]	6.16E-05
50	64	0.01	0.4	nadam	mae	1	[50, 0, 0]	6.38E-05
150	32	0.01	0.4	nadam	logcosh	1	[200, 0, 0]	6.39E-05
100	128	0.01	0.7	nadam	mae	1	[200, 0, 0]	6.44E-05
100	32	0.01	0.2	nadam	mae	1	[200, 0, 0]	6.47E-05
150	64	0.05	0.1	nadam	mae	1	[200, 0, 0]	6.56E-05
150	128	0.01	0.6	adam	mae	1	[200, 0, 0]	6.59E-05
150	64	0.01	0.7	nadam	mae	1	[50, 0, 0]	6.90E-05
100	64	0.01	0.7	adam	logcosh	1	[200, 0, 0]	7.29E-05
100	64	0.01	0.4	nadam	mae	1	[50, 0, 0]	7.49E-05

Table 11. LSTM with hyperparameter optimization results.

LSTM with Hyperparameter Optimization Results												
	Training				**Validation**				**Test**			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.9279	5.66E-09	7.53E-05	4.77E-05	0.958	1.17E-08	1.08E-04	5.75E-05				
3	0.8084	5.52E-09	7.43E-05	3.43E-05	0.8476	4.64E-09	6.81E-05	3.54E-05				
4	0.9023	1.54E-09	3.93E-05	2.81E-05	0.9147	2.62E-09	5.12E-05	3.08E-05	0.9176	7.24E-09	8.51E-05	3.28E-05
5	0.8373	7.07E-08	2.67E-04	5.06E-05	0.9592	2.24E-09	4.74E-05	2.50E-05				
8	0.9276	2.99E-09	5.47E-05	2.62E-05	0.8892	5.77E-09	7.60E-05	3.80E-05				

With respect of the number of layers and units, 1 layer and 200 units are sufficient. For the other hyperparameters, the final values will be 0.01 for the Learning Rate, mae for the Loss Function, 0.5 for the Dropout Value and nadam for the Optimizer. The value for Dropout could be another value, thus it is not clear which one is the optimal. The results of the models of the third generation for Auto-Encoder can be referred to the table 12. Given the results, the final model will run with 150 epochs, although it is not clear whether it should be 100 or 150.

Due to the fact that there is an early stopper for epochs, there is no need to try both of them. Values of 64 and 128 will be chosen for the batch size. The final results will only show the value with the best results. With respect of the number of layers, 1 layer is sufficient and the units assigned to both Encoder and Decoder will be 200. For the other hyperparameters, the final values will be 0.0001 for the Learning Rate, 0.4 for the Dropout Value and Adam for the Optimizer. For the case of loss function, there is no a clear criteria for which one is the best, therefore all of them were chosen and similar to the batch size, the final results will only show the value with the best result. Finally, the best model for loss function and batch size were MAE and 64 respectively.

5.2 Ablation Study Results

5.3 Results Interpretation

In the One-Window experiment, the baseline GARCH model, when compared to Deep Learning models, fails to predict volatility accurately. This is evident from its higher MSE, RMSE, and MAE values across all windows.

Table 12. Auto-encoder top 10 third generation results.

Epochs	Batch	Lr Rate	Dropout	Optimizer	Loss	Encoder Layers	Encoder Units	Decoder Layers	Decoder Units	Results
150	128	0.0001	0.4	adam	logcosh	1	[50, 0, 0]	1	[200, 0, 0]	7.35E-05
100	64	0.0001	0.4	adam	huber	1	[200, 0, 0]	1	[200, 0, 0]	7.46E-05
150	64	0.0001	0.6	adam	mse	1	[100, 0, 0]	2	[200, 200, 0]	7.56E-05
100	128	0.0001	0.6	adam	logcosh	1	[50, 0, 0]	1	[200, 0, 0]	7.58E-05
150	32	0.0001	0.4	nadam	huber	1	[200, 0, 0]	1	[200, 0, 0]	7.69E-05
150	64	0.0001	0.5	nadam	huber	1	[200, 0, 0]	3	[100, 100, 100]	7.71E-05
100	256	0.001	0.7	adam	mse	1	[200, 0, 0]	1	[200, 0, 0]	7.74E-05
150	32	0.0001	0.3	adam	mse	1	[200, 0, 0]	1	[200, 0, 0]	7.77E-05
100	128	0.001	0.4	nadam	mae	1	[50, 0, 0]	1	[200, 0, 0]	7.79E-05
100	32	0.0001	0.1	nadam	logcosh	2	[200, 200, 0]	1	[200, 0, 0]	7.83E-05

However, GARCH outperforms the Mean Exercise in all metrics and sets, except for MAE in the Test Set, indicating that classic models can provide some insights into volatility changes over time but still lag behind more sophisticated models. Deep Learning models exhibit lower errors on metrics such as MAE and RMSE, proving their robustness to outliers while maintaining overall low error values. An important highlight is the significant difference of the Auto-Encoder over LSTM on error metrics results as evidenced in table 2. For example, the Auto-Encoder outperforms LSTM by 5.34E-09 on MSE and 0.65E-05 on MAE in the test results.

Both LSTM and Auto-Encoder consumed 13.8GB and 14.2GB of RAM system respectively as well as 4.2 GB GPU RAM. In the Walk-Forward experiment it is noticeable on GARCH Results in table 4 that although on Test set shows a significant improvement compared to the One-Window Training, it does not show consistency among Validation Folds, which is an indicator that this baseline model is not the optimal model for volatility forecasting.

Compared to the folds of the Mean Exercise that are found in table 3, GARCH folds results outperforms them almost all on Training, Validation and Test sets, with the exception of the MAE results on the first three fold of the Training Set, which were also the folds with the worst results for the GARCH model. These results reinforces the idea of the previous paragraph that classic models are able to provide some information about how volatility changes over time. Regarding the Deep Learning models, results in tables 5 and 6 show that both models have consistency among their folds on all metrics and on all sets, suggesting robustness and high accuracy in volatility prediction. Only LSTM on its fold 8 shows inconsistency and a bad performance on the Validation set.

Regarding the Test set, LSTM improves its results compared to the first experiment and Auto-Encoder results decline slightly but not significantly. For the T-Fold-SV experiment LSTM results were mixed. The folds 0 and 5 had a good performance contrary to the folds 3, 4 and 8. Test Results were very similar to the One-Window experiment, specifically, the differences in MSE and MAE were 0.02E-08 and 0.14E-05, respectively. The memory usage and Test results made this experiment eligible for hyperparameter optimization. The results with the original scale for Training, Validation and Test can be found in the table 7. The Auto-Encoder outperforms significantly the LSTM model in this experiment.

Table 13. Auto-encoder with hyperparameter optimization results.

Auto-Encoder with Hyperparameter Optimization Results												
	Training				Validation				Test			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.8567	1.20E-08	1.09E-04	7.03E-05	0.9269	2.12E-08	1.46E-04	8.09E-05				
3	0.7519	6.36E-09	7.98E-05	3.44E-05	0.7921	7.51E-09	8.67E-05	4.34E-05				
4	0.8735	1.70E-09	4.12E-05	2.57E-05	0.8785	3.74E-09	6.12E-05	2.99E-05	0.9043	8.41E-09	9.17E-05	4.58E-05
5	0.8902	3.78E-08	1.95E-04	5.37E-05	0.9499	2.53E-09	5.03E-05	3.14E-05				
8	0.8912	4.01E-09	6.33E-05	3.43E-05	0.8396	8.07E-09	8.99E-05	5.18E-05				

Table 14. LSTM ablation study results.

LSTM Ablation Study Results												
	Training				Validation				Test			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.8240	9.37E-09	9.68E-05	5.8E-05	0.8479	2.53E-08	1.59E-04	7.26E-05				
3	0.4219	2.08E-08	1.44E-04	4.5E-05	0.8069	4.34E-09	6.59E-05	3.34E-05				
4	0.8594	2.44E-09	4.94E-05	2.58E-05	0.5947	1.11E-08	1.05E-04	4.36E-05	0.8801	1.05E-08	1.03E-04	4.55E-05
5	0.7301	7.77E-08	2.79E-04	5.9E-05	0.9428	3.17E-09	5.63E-05	2.95E-05				
8	0.9773	9.69E-10	3.11E-05	1.6E-05	0.7769	1.20E-08	1.09E-04	5.3E-05				

It is able to obtain consistent results among all the T-Folds-SV, which also made it a candidate for Hyperparameter optimization, supporting the hypothesis that selected T-Folds-SV were appropriate and that the model remains robust and effective for the research problem. The results with the original scale for Training, Validation and Test can be found on table 8. The highlight of this experiment was their results on RAM consumption that were significant reduced compared to the previous experiments. There was a consumption of 5.4 GB for RAM System and 0.6 GB for GPU RAM for both models. For a comparison between experiments refer to the table 9.

The RAM usage for the Auto-Encoder was reduced to approximately one-third of the previous experiments usage, and the RAM usage for the LSTM was reduced to slightly more than half compared to the One-Window Experiment and to almost one-third compared to the Walk-Forward Experiment. For GPU RAM, both models reduced usage by nearly six times compared to the One-Window Experiment and by nearly four times compared to the Walk-Forward Experiment. Lastly, regarding the runtime, although the Walk-Forward experiment is a classic architecture for time series validation, it is computationally expensive and takes the most time for a single iteration. LSTM needed 12:48 min and Auto-Encoder 34:03 min for one single run.

T-Fold-SV experiment, on the contrary, manages to be the least time-consuming for both models. More generations are needed for the Evolutionary Algorithms with T-Fold SV experiment to achieve better parametric stability for hyperparameters on both LSTM and Auto-Encoder models, especially the last one. For the LSTM model, for example, table 10 shows no clear best values for Dropout and Epochs. However, with the selected optimal values, LSTM shows significant improvement over the previous experiments. Th results in table 11 for Validation sets are better than the results from the results of T-Folds SV with no hyperparameter optimization found in table 7.

Table 15. Auto-encoder ablation study results.

Auto-Encoder Ablation Study Results												
	Training				**Validation**				**Test**			
fold	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**	R^2	**MSE**	**RMSE**	**MAE**
0	0.8460	1.57E-08	1.25E-04	8.88E-05	0.9250	2.04E-08	1.43E-04	8.96E-05				
3	0.8119	7.13E-09	8.44E-05	3.97E-05	0.8497	5.87E-09	7.66E-05	4.30E-05				
4	0.8911	2.65E-09	5.15E-05	3.13E-05	0.8721	3.06E-09	5.53E-05	3.04E-05	0.8976	9.00E-09	9.49E-05	4.63E-05
5	0.8306	5.46E-08	2.34E-04	6.53E-05	0.9559	2.50E-09	5.00E-05	2.92E-05				
8	0.8662	4.71E-09	6.86E-05	3.53E-05	0.8606	7.82E-09	8.84E-05	4.99E-05				

As an example, the fold 4 shows an RMSE improvement of 7E-05. Test set results also surpass those from previous experiments, with an MSE of 7.24E-09 compared to 1.45E-08 from the T-Fold SV with no hyperparameter optimization experiment, 1.18E-08 from Walk-Forward experiment and 1.43E-08 obtained on the One-Window experiment. For the Auto-Encoder, more generations are needed to stabilize hyperparameters due probably to the model's complexity. As shown in table 12, there are no clear optimal values for loss function, batch size and epochs. The criteria stated on subsection 5.1 was needed to determine their best values.

The results for Auto-Encoder are shown in table 13 and they showed an improvement over the T-Fold SV experiment with no hyperparameter optimization in Validations sets on all metrics. The MAE result on fold 4 was the only exception, with a result of 3.08E-05 compared to a result of 2.81E-05 obtained from the T-Fold SV with no hyperparameter optimization experiment.

In the Test Set, the final model surpasses its previous experiments with an MSE of 8.41E-09 compared to 1.12E-08 from the T-Fold SV with no hyperparameter optimization experiment, 1.00E-08 from Walk-Forward experiment and 8.96E-09 obtained on the One-Window experiment. Notably, LSTM's top 10 results in Table 10 are better than the Auto-Encoder's, ranging from 6.16E-05 to 7.49E-05 compared to 7.35E-05 to 7.83E-05, indicating that with the right hyperparameters, LSTM can outperform more complex models.

This statement is reinforced by the fact that the Test results from the best models of LSTM and Auto-Encoder showed that the first outperforms the other with a MSE result of 7.24E-09 compared to the result of 8.41E-09. Both models benefit from MAE as the loss function, balancing prediction errors and aiding convergence, while Adam and Nadam optimizers are preferred for their fast convergence and adaptive learning rates. SGD and Adadelta may struggle to converge and require more epochs, but generally, epochs beyond 150 are unnecessary for convergence.

Regarding the Ablation Study, it shows that disabling early stopping leads to a reduction in almost all performance metrics, both on training and validation sets for both models. We can interpret this decreased performance as a sign of the positive effect on adding early stopping criteria, which had 30 epochs of patience until stale results in the cost function triggers a stop-and-reload of the previous weights, and more importantly, the benefit of an information-based criteria to select the training datasets in order to perform Out-of-distribution generalization.

Nevertheless, to not add early stopping criteria did not strongly diminished the benefits of all the previous considerations on this modeling framework, because, on average, the difference in the performance metrics was negligible, as shown in tables 14 and 15.

6 Conclusions and Future Work

On this work it was first analyzed if a baseline model such as GARCH was able to forecast volatility and obtain significant differences compared to average the past volatility and use that mean value for future volatility timesteps and the results on both One Window Training and Walk-Forward experiments showed that GARCH indeed is able to outperform the Mean Exercise, but not significant, which indicates that more sophisticated models such as Deep Learning Models are required and therefore, LSTM and Auto-Encoder are proposed.

This research presented the obstacle that both face regarding their consumption of RAM and GPU RAM memory, which makes them hard to implement on large datasets and also hard to optimize. In order to deal with that issue, the T-Fold SV proposal to split the data, select unique features and then calculate Kullback-Leibler Divergence to select the sets that may provide unique information was applied on the dataset. This process helped to reduce significantly the consumption of RAM and GPU RAM memory and was also able to get similar results in comparison to training the whole dataset.

For hyperparameter optimization, the Genetic Algorithm focused on the values of Epochs, Batch Size, Learning Rate, Loss Function, Optimizer, Layers and Number of Units. After three generations it was clear for most of the values of the LSTM Model which hyperparameter values to use, but for Auto-Decoder was not clear enough for some of its values, which may be an indication that for more complex models, more generations are required. With the optimized hyperparameter values, LSTM managed to outperformed the Auto-Encoder and showed a significant improvement compared to its previous experiments.

Both models have also the property that performed better with MAE loss function, which is an indication that MSE may overpenalize volatility spikes. As for the optimizers, both models showed good performance with Adam and Nadam, which leads to conclude that a strong capability for adaptive learning and momentum are required for a better performance. Lastly, an Ablation study was perform removing the early stopping criteria on both best models for LSTM and Auto-Encoder and the results showed that there indeed a negative impact on the results, but not significant.

Future Works may incorporate new models and new input variables as well as compare results on different time frequencies, different cryptocurrencies and different financial markets. To deep dive on time frequencies, it has been proved on this work it is possible to forecast intraday volatility, therefore it is necessary to research more on what works on high frequency volatility. Regarding the input variables, Onchain Data, Twitter or News may also bring relevant information that strengthen the models capabilities. On cryptocurrencies, most of the researches have focused on BTC and there are other major cryptocurrencies that are under-researched including Ether.

With more resources, it could be possible to try running more generations of models and also including other hyperparameters. It will also be interesting to test Adam and Nadams own hyperparameters because they were the best performing optimizers for both models and were only tested with their default hyperparameters. A more in-depth analysis of the impact of different hyperparameters and their interactions could provide a comprehensive understanding of the models robustness and more ablation studies could be conducted. Lastly, regarding the T-Folds SV, more exploration is required to exploit its potential to Time Series Problems Applications. One of them could be to label distributions and train specific models for each distribution instead of one model for all the distributions and for an unseen distribution use one of the models that were trained that fits the most.

References

1. Aidov, A., Lobanova, O.: The relation between intraday limit order book depth and spread. International Journal of Financial Studies 9(60), 1–13 (2021)
2. Charandabi, S., Kamyar, K.: Survey of cryptocurrency volatility prediction literature using artificial neural networks. Business and Economic Research 12(1), 17–27 (2022)
3. Guo, T., Bifet, A., Antulov-Fantulin, N.: Bitcoin volatility forecasting with a glimpse into buy and sell orders. IEEE International Conference on Data Mining pp. 989–994 (2018)
4. Jang, H., Jaewook, L.: An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. IEEE Access 6, 5427–5437 (2018)
5. Jung, G., Choi, S.Y.: Forecasting foreign exchange volatility using deep learning autoencoder-LSTM techniques. Complexity 2021, 16 (2021)
6. Lopez-de-Prado, M.: Advances in financial machine learning. Wiley (2018)
7. Miura, R., Pichl, L., Kaizoji, T.: Artificial neural networks for realized volatility prediction in cryptocurrency time series. Advances in Neural Networks 11554, 165–172 (2019)
8. Muñoz-Elguezábal, J., Sánchez Torres, J.D.: T-fold sequential validation technique for out-of-distribution generalization with financial time series data. In: Proceedings of the 4th International Conference on Econometrics and Statistics (2021)
9. Naimy, V.Y., Hayek, M.R.: Modelling and predicting the bitcoin volatility using garch models. International Journal of Mathematical Modelling and Numerical Optimisation 8(3), 197–215 (2018)
10. Rodikov, G., Antulov-Fantulin, N.: Can LSTM outperform volatility-econometric models? (2022)