Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis MSc Econometrics and Management Science

Exploring the Effectiveness of Wavelet Neural Networks in Forecasting Daily Realized Volatility: A Comparative Analysis.

Kylym Meirazhdinov (467160km)

Supervisor:	dr. M. Grith
Second assessor:	dr. A. Pick
Date final version:	14th August 2023

Abstract

This thesis provides a methodology on the use of Discrete Wavelet Transform to decompose the time series into low and high frequency components that can be used as regressors for time series models. Two different approaches are provided where the realized volatility time series is modelled directly, and the other allows to model the dynamics of the individual components first to then assemble them into a time series forecast. Linear regression analysis as well as deep learning models are employed. Multiple time horizons forecasts of the models introduced in this thesis, are compared with the popular models such as HAR model (Corsi, 2009) and Extended Wald Decomposition model (Ortu, Severino, Tamoni & Tebaldi, 2020).

Contents

Chapter 1

Introduction

Volatility is a fundamental concept in finance that measures the degree of variation or dispersion of a financial instrument's price or return over time. It is a critical parameter used by financial analysts, investors, and risk managers to assess and manage investment risk. In financial markets, volatility is a key factor that can impact the performance of investment portfolios, influence trading decisions, and affect the pricing of options and other derivative securities. Volatility describes the uncertainty and unpredictability of financial markets or specific assets. Higher levels of volatility indicate bigger fluctuations in the data, vice versa. It is a vital input in various financial models, such as option pricing models, value-at-risk (VaR) models, and portfolio optimization models. Many models rely on volatility estimates to calculate option prices, estimate potential losses, and optimize portfolio allocations.

Financial returns often exhibit distinct volatility clustering, which is a phenomenon where periods of high or low volatility tend to persist over time. Volatility, however, cannot be directly observed. This creates complications for accurate predictions as direct forecasting is not possible.

One branch of the research field is built upon the construction of estimators for volatility by utilizing high-frequency data. Andersen & Bollerslev (1998) show that the use of high-frequency data enables the creation of significantly better measurements of volatility through the accumulation of squared intraday returns, with the square root of the sum of squared returns (realized volatility) as an unbiased estimator of latent volatility.

Corsi (2009) proposed a heterogeneous autoregressive (HAR) model which is an AR-type model for realized volatility. The model utilizes the realized volatility over different time horizon. More specifically, the next day realized volatility is predicted using the information on the

previous day realized volatility, an average of the past 5 days (working days) corresponding to a week and an average of the past 22 days (working days) corresponding to a month. Despite the simplicity, the authors show that the model is able to outperform time series models such as AR and ARFIMA. With the popularity of machine learning models, Reisenhofer, Bayer & Hautsch (2022) successfully attempted to close the conceptual divide between traditional time series methods, like the Heterogeneous Autoregressive (HAR) model introduced by Corsi (2009), and deep neural network models. The authors present the model (HARNet) that is based on a hierarchy of dilated convolutional layers, which can significantly enhance the predictive precision of HAR baseline models.

With the rising popularity of wavelet transformations in image processing, due to its ability in signal processing such as signal denoising, signal compression, and feature extraction, the utilization of wavelet transforms in modeling financial time series data becomes more interesting.

Ortu et al. (2020) proposed a method for decomposing stationary time series into orthogonal components based on their persistence properties. The paper demonstrate the usefulness of the proposed method by applying it to a realized volatility time series. They show that the persistence-based decomposition provides insights into the underlying dynamics of the time series that are not captured by the traditional Wold decomposition. More specifically, the authors show the method to analyze a time series in terms of its persistence components.

Given the classical Wold decomposition of the time series $\boldsymbol{x}$,

$$x_t = \sum_{h=0}^{+\infty} \alpha_h \varepsilon_{t-h} \tag{1.1}$$

Theorem 1 (Ortu et al., 2020) states that if $\boldsymbol{x}$ is a zero-mean, weakly stationary purely non-deterministic stochastic process. Then $\boldsymbol{x}$ decomposes as

$$x_t = \sum_{j=1}^{+\infty} \sum_{k=0}^{+\infty} \beta_k^{(j)} \varepsilon_{t-k2^j}^{(j)} \tag{1.2}$$

where for $j \in \mathbb{N}$, the process $\boldsymbol{\varepsilon^{(j)}} = \{\epsilon_t^{(j)}\}_{t\in\mathbb{Z}}$ with

$$\varepsilon_t^{(j)} = \frac{1}{\sqrt{2^j}} \left(\sum_{i=0}^{2^{j-1}-1} \varepsilon_{t-i} - \sum_{i=0}^{2^{j-1}-1} \varepsilon_{t-2^{j-1}-i} \right) \tag{1.3}$$

is a MA($2^j - 1$) with respect to the innovations of $\boldsymbol{x}$ in Eq. (1.1).

Next, for any $j \in \mathbb{N}$, $k \in \mathbb{N}_0$, the coefficients $\beta_k^{(j)}$ are uniquely determined via

$$\beta_k^{(j)} = \frac{1}{\sqrt{2^j}} \left(\sum_{i=0}^{2^{j-1}-1} \alpha_{k2^j+i} - \sum_{i=0}^{2^{j-1}-1} \alpha_{k2j+2^{j-1}+i} \right) \tag{1.4}$$

making the coefficients time invariant.

Combining the previous results, let

$$g_t^{(j)} = \sum_{k=0}^{+\infty} \beta_k^{(j)} \varepsilon_{t-k2^j}^{(j)} \tag{1.5}$$

it is important to notice that many of the assumptions and properties have not been mentioned above as the purpose is to provide a rough idea of how the Extended Wold Decomposition works.

Simplifying the results further by substituting the results in Eq. (1.5) into Eq. (1.2), we obtain another representation of the extended Wold decomposition,

$$x_t = \sum_{j=1}^{+\infty} g_t^{(j)} \tag{1.6}$$

The main object of interest is $g_t^{(j)}$, which authors refer to as persistent component at scale j, where the scale j involves the stocks that last 2^j working days. In the empirical analysis section (Ortu et al., 2020), the authors in the linear setting, use the persistent components as regressors and by iteratively adding components by their explained variance, noticed that after including three scales j = 7,8,9 corresponding to 128, 256 and 512 working days, there is no significant improvement in the forecasting performance. The authors also linked their findings to the HAR model, mainly stating that the daily, weekly and monthly components used in HAR effectively serve as indicators/estimates for events that take place on timescales ranging from six months to two years.

Bandi, Perron, Tamoni & Tebaldi (2019) as a part of their third contribution mention that the predictability is not modeled directly on the raw time series, but rather, the dependency is defined as a linear relation between the regressor and the regressand at individual scales, property to which the authors refer to as scale-specific predictability. An example presented in

the paper is the regressions of the components of excess stock market returns on the components of market variance.

$$r^{(j)}_{k2^j+2^j} = \beta_j v^{(j)}_{k2^j} + u^{(j)}_{k2^j+2^j}$$

with j indicating a specific scale. Following section 5 (Filtering the scale-specific components) of the paper (Bandi et al., 2019), the authors present the methodology to decompose the original time series into "transitory" and "persistent" components. Let $\boldsymbol{x}$ be the time series, then for $J = 1$, the following decomposition can be achieved,

$$x_t = \frac{x_t - x_{t-1}}{2} + \frac{x_t + x_{t-1}}{2}$$

Where on the right-hand side, the first part is the "transitory" component that we denote as $\hat{x}^{(1)}_t$ and the second part is the "persistent" component $\hat{\pi}^{(1)}_t$. By applying the same procedure to the "transitory" component we can further break down the original time series into components corresponding to lower frequencies. Another representation of the filter uses a projection operator, for $J = 2$

$$\begin{pmatrix} \hat{\pi}^{(2)}_t \\ \widehat{x}^{(2)}_t \\ \widehat{x}^{(1)}_t \\ \widehat{x}^{(1)}_{t-2} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} x_t \\ x_{t-1} \\ x_{t-2} \\ x_{t-3} \end{pmatrix} \tag{1.7}$$

Where in the paper the matrix in Eq. (2.2) is denoted by $\mathcal{T}^{(2)}$, being orthogonal and invertible. The invertibility allows for signal reconstruction given the filtered components. The key difference with the Haar transforms (matrix) lies in the basis (matrix) that is used.

$$H_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{pmatrix}$$

with the H_4 subscript indicating the size of the matrix. H_4 also being orthonormal, meaning $H_4 H_4^T = I$. As a possible extension, it might be valuable to investigate whether the predictive power of the components can be unfolded in the non-linear setting while not being limited to

the individual scale.

In this thesis HAR model (Corsi, 2009) and Extended Wold Decomposition model (Ortu et al., 2020) are the benchmark models. Research on using deep neural networks with wavelets for realized volatility forecasting can contribute to the development of more sophisticated and accurate models for financial forecasting and analysis. By combining deep learning techniques with signal processing methods, such as wavelet analysis, researchers can capture nonlinear and multiscale interactions in financial data, leading to more robust and accurate models.

Chapter 2

Methodology

2.1 Data & Realized Volatility

The data used in this proposal and that will further be used in the main text, is the daily time series of USD/CHF exchange rates realized volatility built by Corsi (2009). The original time series is the tick-by-tick series of USD/CHF exchange rates ranging from December 1989 to December 2003. The construction of the daily realized volatility and the notation follows the same paper (Corsi, 2009),

$$RV_t^{(d)} = \sqrt{\sum_{j=0}^{M-1} r_{t-j\cdot\Delta}^2} \tag{2.1}$$

with $\Delta = 1d/M$ and $r_{t-j\cdot\Delta} = p(t - j\cdot\Delta) - p(t - (j+1)\cdot\Delta)$, where $p(t)$ the logarithm of instantaneous price and $M = 12$ denoting the time intervals of two hours in a 24 hours trading day. In the dataset, the logarithmic middle price is used as a proxy for the true price.

The sample size of daily realized volatility is 3599 observations and the wavelet decomposition is only performed up to scale $J = 3$, due to the machine learning models requiring bigger samples to train. Furthermore, the sample size has been extended by a single value 0, which is placed at the end of the sample, the process is referred to as "Zero-padding". It's done to make the sample size a power of two or a multiple of the decomposition factor.

	Mean	**Median**	**Std. Dev.**	**Min**	**Max**	**Skewness**	**Kurtosis**
RV	12.397	11.593	3.900	3.828	41.947	1.594	5.071

Table 2.1: Descriptive statistics for the series of daily realized volatility.

2.2 Discrete Wavelet Decomposition

Let $\{x_{t-i}\}_{i\in\mathbb{N}}$ be the realized volatility time series. We apply the discrete wavelet transform (DWT), using the modified Haar matrix (Eq. 2.2) proposed by Bandi et al. (2019). For convenience, we start from scale $j = 1$. Sliding the wavelet matrix down the time series, we split the original series into a persistent component $\hat{\pi}_t^{(1)}$ and a transitory component $\hat{x}_t^{(1)}$.

$$\begin{pmatrix} \hat{\pi}_t^{(1)} \\ \hat{x}_t^{(1)} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix} \tag{2.2}$$

To further obtain components at higher scales, one can further decompose the persistent component ($\hat{\pi}_t^{(1)}$) by repeating the same procedure or directly using the Haar-like matrix of higher order on the original time series. The intuition behind the DWT is depicted in Fig. (2.1). The persistent component represents the slowly changing or trend-like behavior of a time series. When a low-pass filter is applied to a time series, it removes the high-frequency variations and noise, emphasizing the lower-frequency components that change slowly over time. The transitory component can be seen as innovations, it highlights the short-term fluctuations enabling the analysis of transient effects.

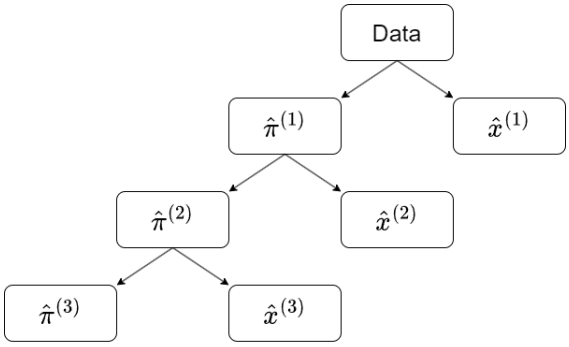


Figure 2.1: The graphical representation of the wavelet transforms with $\hat{\pi}$ the slow changing components and the $\hat{x}$ the high-frequency variations.

It is important to mention that with each scale the length of the time series shrinks by a factor of 2, meaning that for scale j if n is the length of the original series, the amount of observations available is $\lfloor \frac{n}{2^j} \rfloor$. In this research, the DWT is used due to its energy preservation and perfect reconstruction properties.

Given the decomposition level J, applying reconstruction filters, that are specific to the wavelet used in the DWT, to the upsampled details coefficients and upsampled level J approximation coefficients, and then adding them together yields the reconstructed signal. In this thesis,

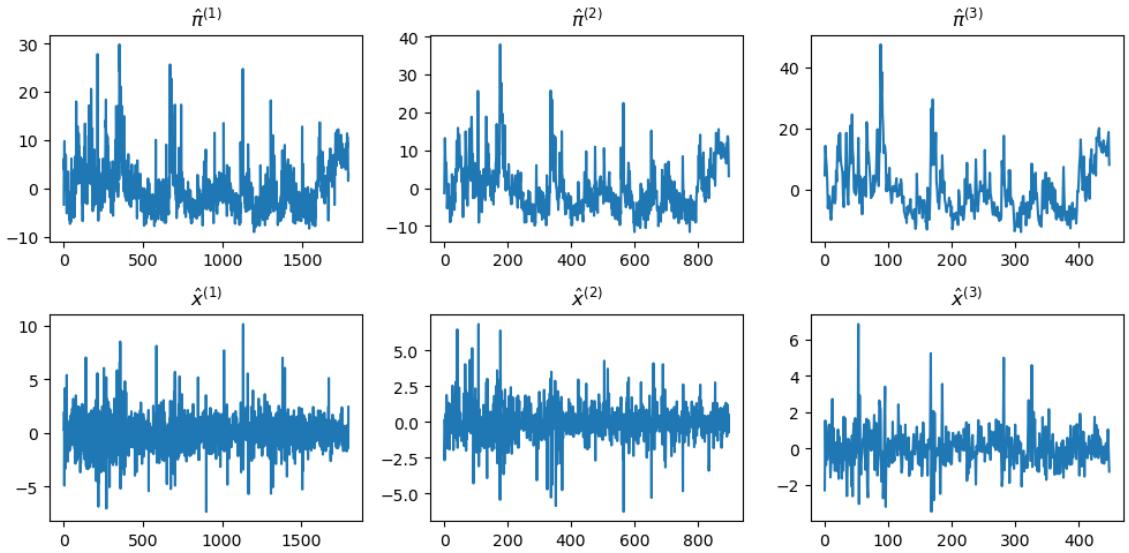


Figure 2.2: Persistent and transitory components of the realized volatility time series up to scale $J = 3$. The original time series plot can be found in the appendix (Fig. 6)

discrete wavelet decomposition is taken up to level $J = 2$.

2.3 Direct Forecasting

The next day realized volatility will be directly modeled as a function of the lagged coefficients and lags of the realized volatility itself. The structure of the wavelet coefficients can be seen in Fig. 2.3, for instance, if the current day is 4 and the prediction for day 3 has to be made then the latest wavelet coefficients available are $\hat{\pi}_4^{(1)}$, $\hat{x}_4^{(1)}$, $\hat{\pi}_4^{(2)}$ and $\hat{x}_4^{(2)}$.

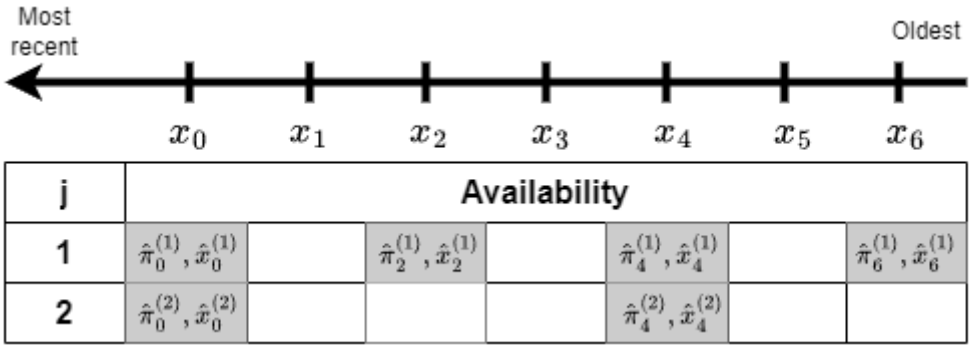


Figure 2.3: The axis represents the time in days, where the most right point represents older observations and the most left point is the most recent. The table shows the time correspondence between the original time series and wavelet coefficients. It's important to note that the coefficients in empty cells are not defined due to downsampling.

2.4 Multistep Forecasting

2.4.1 Conditional Multistep Forecasting

The following method relies on the orthogonality of the wavelet matrix. Due to the perfect reconstruction properties, wavelet components can be estimated individually and then assembled to obtain the forecast for the original time series.

The prediction of each scale coefficient requires the use of its own lags as well as the lags of the other components. A visual representation of this concept is depicted in Figure 2.4, which illustrates the time axis, and a table below it indicates the availability and order of the wavelet coefficients.

To illustrate the forecasting process, let's consider predicting the value of x_3. The forecasted value, denoted as $\hat{x}_3$, is obtained using the conditional expectation $\mathbb{E}[x_3 \mid \mathcal{F}_3]$, which is the expectation of x_3 given the information available at time 3, represented by $x_4, x_5, x_6, \ldots$. In the table, this corresponds to the position labeled as 4.

Following the multistep approach, in order to predict x_3, we require the coefficients $\hat{\pi}_0^{(2)}$, $\hat{x}_0^{(2)}$ and $\hat{x}_2^{(1)}$, for the reconstruction. To obtain $\hat{x}_0^{(2)}$, we need to consider the most recent scale coefficients available, which are $\hat{\pi}_4^{(1)}$, $\hat{x}_4^{(1)}$, $\hat{\pi}_4^{(2)}$ and $\hat{x}_4^{(2)}$.

We see that the most recent lag available for scale $j = 1$ is four days away. However, if we attempt to predict x_1 given information up to and including $t = 2$, once again requiring $\hat{x}_0^{(2)}$, we now have $\hat{\pi}_2^{(1)}$ and $\hat{x}_2^{(1)}$ for scale $j = 1$, which are only two days away. Therefore, the temporal structure changes, and the same model can't be used to predict $\hat{x}_0^{(2)}$ in these two cases.

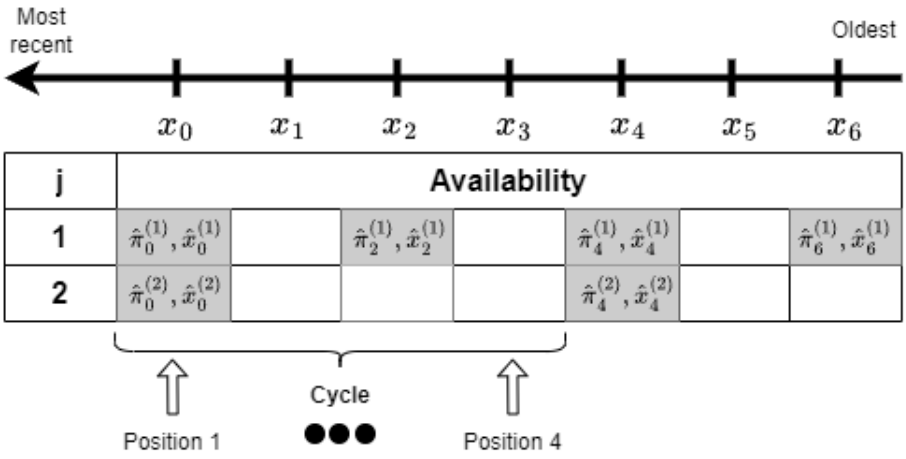


Figure 2.4: The axis represents the time in days, where the most right point represents older observations and the most left point is the most recent. The table is the matrix representation of the chronological order of wavelet coefficients with timepoint correspondence. The positions depicted on the figure refer to the positions within the cycle, in this case the maximum scale is 2, meaning that the cycle is 4 days. Depending on the positions, different lags are available. It's important to note that the coefficients in empty cells are not defined due to downsampling.

Hence, we can assign a specific modelling structure for positions 1,2 and position 3,4 within the cycle. In order to make forecasts for the wavelet coefficients corresponding to positions 3 and 4 the following linear equations are required,

$$\begin{bmatrix}\hat{\pi}_t^{(2)}\\ \hat{x}_t^{(2)}\end{bmatrix} = \sum_{i=1}^{n}\begin{bmatrix}\beta_{i,1}^{(2)}\gamma_{i,1}^{(2)}\\ \beta_{i,2}^{(2)}\gamma_{i,2}^{(2)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+4i}^{(2)}\\ \hat{x}_{t+4i}^{(2)}\end{bmatrix} + \sum_{i=1}^{n}\begin{bmatrix}\beta_{i,1}^{(1)}\gamma_{i,1}^{(1)}\\ \beta_{i,2}^{(1)}\gamma_{i,2}^{(1)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+2+2i}^{(1)}\\ \hat{x}_{t+2+2i}^{(1)}\end{bmatrix} + \begin{bmatrix}\varepsilon_t\\ \eta_t\end{bmatrix} \tag{2.3}$$

$$\begin{bmatrix}\hat{\pi}_t^{(1)}\\ \hat{x}_t^{(1)}\end{bmatrix} = \begin{bmatrix}\beta_{1,1}^{(2)}\gamma_{1,1}^{(2)}\\ \beta_{1,2}^{(2)}\gamma_{1,2}^{(2)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+2}^{(2)}\\ \hat{x}_{t+2}^{(2)}\end{bmatrix} + \sum_{i=2}^{n}\begin{bmatrix}\beta_{i,1}^{(2)}\gamma_{i,1}^{(2)}\\ \beta_{i,2}^{(2)}\gamma_{i,2}^{(2)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{(t+2)+4(i-1)}^{(2)}\\ \hat{x}_{(t+2)+4(i-1)}^{(2)}\end{bmatrix} + \sum_{i=1}^{n}\begin{bmatrix}\beta_{i,1}^{(1)}\gamma_{i,1}^{(1)}\\ \beta_{i,2}^{(1)}\gamma_{i,2}^{(1)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+2i}^{(1)}\\ \hat{x}_{t+2i}^{(1)}\end{bmatrix} + \begin{bmatrix}\varepsilon_t\\ \eta_t\end{bmatrix} \tag{2.4}$$

If the position is 1 or 2 then,

$$\begin{bmatrix}\hat{\pi}_t^{(2)}\\ \hat{x}_t^{(2)}\end{bmatrix} = \sum_{i=1}^{n}\begin{bmatrix}\beta_{i,1}^{(2)}\gamma_{i,1}^{(2)}\\ \beta_{i,2}^{(2)}\gamma_{i,2}^{(2)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+4i}^{(2)}\\ \hat{x}_{t+4i}^{(2)}\end{bmatrix} + \sum_{i=1}^{n}\begin{bmatrix}\beta_{i,1}^{(1)}\gamma_{i,1}^{(1)}\\ \beta_{i,2}^{(1)}\gamma_{i,2}^{(1)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+2i}^{(1)}\\ \hat{x}_{t+2i}^{(1)}\end{bmatrix} + \begin{bmatrix}\varepsilon_t\\ \eta_t\end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix}\hat{\pi}_t^{(1)}\\ \hat{x}_t^{(1)}\end{bmatrix} = \sum_{i=1}^{n}\begin{bmatrix}\beta_{i,1}^{(2)}\gamma_{i,1}^{(2)}\\ \beta_{i,2}^{(2)}\gamma_{i,2}^{(2)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+4i}^{(2)}\\ \hat{x}_{t+4i}^{(2)}\end{bmatrix} + \sum_{i=1}^{n}\begin{bmatrix}\beta_{i,1}^{(1)}\gamma_{i,1}^{(1)}\\ \beta_{i,2}^{(1)}\gamma_{i,2}^{(1)}\end{bmatrix}\begin{bmatrix}\hat{\pi}_{t+2i}^{(1)}\\ \hat{x}_{t+2i}^{(1)}\end{bmatrix} + \begin{bmatrix}\varepsilon_t\\ \eta_t\end{bmatrix} \tag{2.6}$$

Ignoring the fact that equations for $\hat{\pi}_t^{(1)}$, $\hat{x}_t^{(1)}$ don't look the same, the lags used in forecast are exactly the same, implying that the model for $j = 1$ coefficients is independent of the positions in the cycle. Whereas, the lags for $j = 2$ coefficients differ when switching from position 3 to position 2, as the for the same wavelet coefficient $j = 2$ a new observation of $j = 1$ coefficients becomes available as regressors. Thus, coefficients differ according to the positions.

After predicting the required wavelet coefficients, the assembly of the final forecast of the original time series is done via,

- Position 4, $\hat{x}_t = \hat{\pi}_t^{(2)} - \hat{x}_t^{(2)} - \hat{x}_t^{(1)}$
- Position 3, $\hat{x}_t = \hat{\pi}_t^{(2)} - \hat{x}_t^{(2)} + \hat{x}_t^{(1)}$
- Position 2, $\hat{x}_t = \hat{\pi}_t^{(2)} + \hat{x}_t^{(2)} - \hat{x}_t^{(1)}$
- Position 1, $\hat{x}_t = \hat{\pi}_t^{(2)} + \hat{x}_t^{(2)} + \hat{x}_t^{(1)}$

The advantage of this method is that a clear model structure is defined when making a prediction. However, the cost is that the number of models that need to be calibrated increases.

Algorithm 1 Conditional Multistep Forecasting Process

model_1 - the model that forecasts the $j = 1$ coefficients

model_2 - the model that forecasts the $j = 2$ coefficients for positions 1 and 2 (Eq. 2.5)

model_3 - the model that forecasts the $j = 2$ coefficients for position 3 and 4 (Eq. 2.3)

Predict x_t given information up to and not including t.

Let $r1$ and $r2$ be the $j = 1$ and $j = 2$ indices of the coefficients required for the reconstruction.

$position = t$ - $r2$

X_1 coefficients that are used for forecasting $j = 1$ coefficients

$\hat{\pi}_{r1}^{(1)}, \hat{x}_{r1}^{(1)} = model\_1.predict(X_1)$

if $position == 1$ *or* 2 **then**

    X_2 observations that are used for forecasting $j = 2$ coefficients (Eq. 2.5).

    $\hat{\pi}_{r2}^{(2)}, \hat{x}_{r2}^{(2)} = model\_2.predict(X_2)$

else

    X_3 observations that are used for forecasting $j = 2$ coefficients (Eq. 2.3).

    $\hat{\pi}_{r2}^{(2)}, \hat{x}_{r2}^{(2)} = model\_3.predict(X_3)$

end if

if $position == 4$ **then**

    $x_t = \hat{\pi}_{r2}^{(2)} - \hat{x}_{r2}^{(2)} - \hat{x}_{r1}^{(1)}$

else if $position == 3$ **then**

    $x_t = \hat{\pi}_{r2}^{(2)} - \hat{x}_{r2}^{(2)} + \hat{x}_{r1}^{(1)}$

else if $position == 2$ **then**

    $x_t = \hat{\pi}_{r2}^{(2)} + \hat{x}_{r2}^{(2)} - \hat{x}_{r1}^{(1)}$

else if $position == 1$ **then**

    $x_t = \hat{\pi}_{r2}^{(2)} + \hat{x}_{r2}^{(2)} + \hat{x}_{r1}^{(1)}$

end if

2.4.2 Unconditional Multistep Forecasting

Another approach is to ignore the modelling difference between Eq. 2.5 and Eq. 2.3, and define a linear model that uses the same coefficients unconditional of the position within the cycle (Fig. 2.4).

Algorithm 2 Unconditional Multistep Forecasting Process

1: $model\_1$ - the model that forecasts the $j = 1$ coefficients
2: $model\_2$ - the model that forecasts the $j = 2$ coefficients
3: Predict x_t given information up to and not including t
4: Let $r1$ and $r2$ be the $j = 1$ and $j = 2$ indices of the coefficients required for the reconstruction
5: $position = t$ - $r2$
6: X_1 coefficients that are used for forecasting $j = 1$ coefficients
7: $\hat{\pi}_{r1}^{(1)}, \hat{x}_{r1}^{(1)} = model\_1.predict(X_1)$
8: X_2 observations that are used for forecasting $j = 2$ coefficients
9: $\hat{\pi}_{r2}^{(2)}, \hat{x}_{r2}^{(2)} = model\_2.predict(X_2)$
10: **if** $position == 4$ **then**
11: $\quad x_t = \hat{\pi}_{r2}^{(2)} - \hat{x}_{r2}^{(2)} - \hat{x}_{r1}^{(1)}$
12: **else if** $position == 3$ **then**
13: $\quad x_t = \hat{\pi}_{r2}^{(2)} - \hat{x}_{r2}^{(2)} + \hat{x}_{r1}^{(1)}$
14: **else if** $position == 2$ **then**
15: $\quad x_t = \hat{\pi}_{r2}^{(2)} + \hat{x}_{r2}^{(2)} - \hat{x}_{r1}^{(1)}$
16: **else if** $position == 1$ **then**
17: $\quad x_t = \hat{\pi}_{r2}^{(2)} + \hat{x}_{r2}^{(2)} + \hat{x}_{r1}^{(1)}$
18: **end if**

2.4.3 Regularization

Regularization techniques play an important role in addressing issues such as multicollinearity and overfitting. Three widely employed regularization methods are Ridge, Lasso, and ElasticNet. Ridge regularization, also known as L2 regularization, introduces a penalty term to the model's objective function that constrains the magnitude of regression coefficients. This helps prevent excessive coefficient inflation and mitigates multicollinearity effects. Lasso regularization, or L1 regularization, enforces sparsity by adding the absolute values of coefficients to the loss function. Lasso regularization set certain coefficients to be exactly zero, leading to feature selection. ElasticNet combines both L2 and L1 penalties, providing a balance between the strengths of Ridge and Lasso regularization. In regards to the two approaches, direct and

multistep, the former exhibited superior performance with Lasso regularization. Conversely, for the multistep approach, Ridge regularization yielded the most favorable results, suggesting that it effectively countered multicollinearity issues.

2.4.4 Adaptive Forecasting

Due to changing economic conditions, coefficients re-estimation would let models adapt accordingly. Hence, with a specific pereodicity regularization coefficient is re-tuned and weights are recalibrated.

- In the direct forecasting methodology (section 2.3) the models are re-tuned and updated everyday.
- In the multistep forecasting methodology (section 2.4) the models are re-tuned and updated every 22 days (one month).

2.5 Deep Learning Model

The next natural step is to introduce nonlinearity, a powerful approach to capturing complex nonlinear relationships are deep learning models. The advantages of using NNs lie in their flexibility and capacity to handle complex data. Their ability to learn from data allows them to uncover underlying patterns and relationships that may not be apparent using traditional linear models.

2.5.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are the fundamental building blocks of deep learning models. They are computational models inspired by the structure and functioning of biological neurons. ANNs consist of interconnected layers of artificial neurons called "nodes" or "hidden units". Each node receives input signals, processes them through activation functions, and produces output signals. ANNs can learn and make predictions by adjusting the weights and biases associated with the connections between nodes.

The output of a neuron in an ANN is computed using an activation function. Commonly used activation functions include the sigmoid function, the rectified linear unit (ReLU), and/or the hyperbolic tangent function. The choice of activation function depends on the nature of the problem and the desired properties of the model.

Mathematically, the output of a neuron can be represented as follows:

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.7}$$

where y is the output, f is the activation function, w_i and x_i are the weights and inputs respectively, and b is the bias.

Neural networks can become more complex by adding more hyperparameters, which are adjustable settings that affect the behavior and performance of the model during training. By increasing the number of hyperparameters, such as the number of hidden layers, the number of neurons in each layer, the learning rate, the regularization strength, and the activation functions, neural networks gain additional flexibility and capacity to capture intricate patterns in the data.

However, with increased complexity, there are challenges in finding the optimal combination of hyperparameters that strike the right balance between model expressiveness and generalization ability. Proper hyperparameter tuning techniques, such as grid search, random search, or more advanced optimization algorithms, are necessary to explore the vast space of possibilities and maximize the neural network's performance.

Additionally, as neural networks become more complex, they may require larger amounts of computational resources, longer training times, and careful handling to prevent overfitting or underfitting. Careful regularization techniques, such as dropout or L1/L2 regularization, can help mitigate overfitting by adding constraints to the model's parameters.

Neural networks have shown remarkable success in various domains, including image recognition, natural language processing, and time series forecasting. Their ability to learn complex patterns and make accurate predictions has made them a powerful tool in the field of deep learning.

2.5.2 Deep Learning Training

Training deep learning models involves an iterative process of forward propagation, loss computation, and backpropagation. During forward propagation, inputs are passed through the network, and predictions are generated. The loss function measures the dissimilarity between the predicted outputs and the true targets. Backpropagation calculates the gradients of the loss with respect to the model's parameters, allowing the optimization algorithm (e.g., stochastic gradient descent) to update the weights and biases.

The choice of loss function depends on the task at hand. The most popular and standard loss

function used for regression problems is a mean squared error (MSE), which is used in this thesis.

2.5.3 Hyperparameter Tuning

Finding the right set of hyperparameters is an important task that differentiates a good-performing model from a bad-performing model. Multiple optimization algorithms exist and the choice can heavily depend on the circumstance. Hyperparameter tuning is a computationally expensive task, there are tuning algorithms that do a thorough check of the search space such as grid search, which goes through every combination of hyperparameters inside the search space, however, it also leads to a combinatorial explosion in the number of model trainings. Two more tuning algorithms that attracted attention and have proven themselves are,

- Random Search, is a hyperparameter optimization technique commonly used in machine learning to find the optimal combination of hyperparameters for a model. Unlike grid search, which exhaustively explores all possible hyperparameter combinations in a predefined grid, random search randomly selects hyperparameter values from a given search space. The main idea behind random search is that the performance of a model is not solely determined by the values of its hyperparameters but also by the interactions between them. By randomly sampling hyperparameter values, random search allows for a broader exploration of the hyperparameter space, potentially uncovering superior combinations that may not be discovered through a grid search approach.

- Bayesian Optimization, is a powerful technique for hyperparameter optimization that uses probabilistic models to intelligently search the hyperparameter space. It combines the strengths of both random search and model-based optimization, allowing for efficient exploration and exploitation of the search space. The key idea behind Bayesian optimization is to construct a surrogate model, often a Gaussian process (GP), that approximates the unknown performance function mapping hyperparameters to model performance. The GP model provides a probabilistic representation of the objective function, allowing for uncertainty quantification.

2.5.4 Model-Validation

The forecasting methodology incorporates an iterative procedure to tune hyperparameters while efficiently utilizing available data. Initially, the time series is partitioned into a training set and a test set. The test set is never used in the optimization of the neural networks. In attempt to

ensure enough generalization of the models, 30% of the training data is used for hyperparameter validation. Following this, the tuned and trained neural network is used to make n number of predictions in the test set. The true data that is obtained after the predictions for them have been made are subsequently integrated into the training data to create an augmented training dataset. The process is repeated to make the next set of predictions for the test set. This iterative procedure of dataset expansion enables the model to assimilate and adapt to previously unaccounted patterns in the time series data.

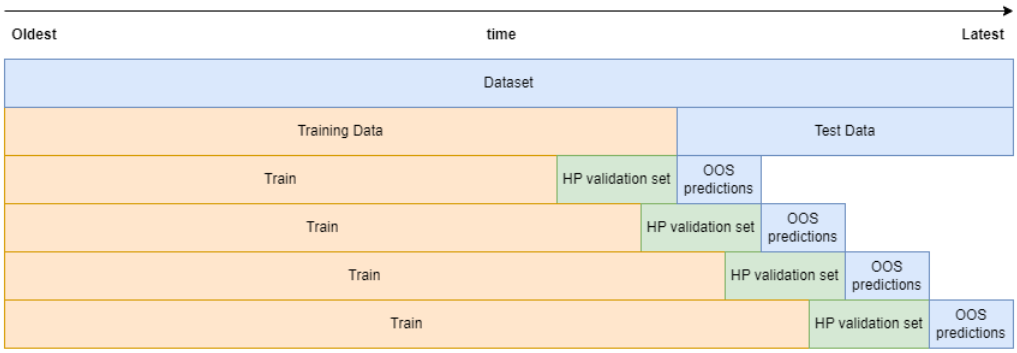


Figure 2.5: Iterative model adjusting procedure.

2.6 The Diebold-Mariano Test

The Diebold-Mariano (DM) test is used to compare the forecast accuracy of two sets of predictions. In this context, the hypothesis being tested is about whether one forecasting model is significantly more accurate than the other.

Let $e_t^{(A)} = y_t - \hat{y}_t^{(A)}$ and $e_t^{(B)} = y_t - \hat{y}_t^{(B)}$ be the residuals of the models A and B. We define a loss function $g(.)$ that can be a square, an absolute value of the residual or of any other more sophisticated form. Define a loss differential, $d_t^{(AB)} = g(e_t^{(A)}) - g(e_t^{(B)})$.

The null hypothesis $H_0 : E[d_t^{(AB)}] = 0$, the alternative depends on whether one sided or two sided test is performed. Given the forecasted values of model A and model B.

- Two sided - $H_a : E[d_t^{(AB)}] \neq 0$.
- One sided - $H_a : E[d_t^{(AB)}] > 0$.

In this manner, the Diebold-Mariano test aids in decision-making, confidently adopting predictive methodologies.

Chapter 3

Forecasting Results

This section provides a comparative analysis of several methodologies discussed above. The NNs have been tuned using the Bayesian Optimization method and sliding window CV. The following models are mentioned in the tables,

- HAR - Heterogeneous Autoregressive model by Corsi (2009).
- EW(9) - Extended Wald Decomposition with 9 components by Ortu et al. (2020).
- CMWLM - Conditional Multistep Wavelet Linear Model (Section 2.4.1).
- UMWLM - Unconditional Multistep Wavelet Linear Model (Section 2.4.2).
- DWLM - Direct Wavelet Linear Model (Section 2.3).
- DWNN - Direct Wavelet Neural Network (Section 2.3).
- UMWNN - Unconditional Multistep Wavelet Neural Network (Section 2.4.2).

The selection of the number of lags in the linear models for approximation and detail coefficients for both scales, as well as the number of lags of realized volatility is done via Bayesian Information Criterion (BIC). The model that has the lowest BIC is selected. Using random search, the selection of the variables can be significantly sped up.

For the deep learning models, Table 3.1 contains the possible values for the hyperparameters.

Table 3.1: Hyperparameter Ranges. Due to data limitation, via trial and error having more than a single hidden layer only worsens the performance as the model becomes more complex, same applies to the number of hidden neurons, which increases the performance within each CV set, however, the final model struggles to generalize. Furthermore, the regularization type used is the kernel regularizer with l1 and l2 regularizations combined.

Hyperparameter	**Value Range**
Number of Hidden Layers	1
Number of Neurons per Layer	[10, 200]
Optimizer	Adam
Activation Function	{ReLU, tanh, sigmoid}
Learning Rate	[0.1, 0.01, 0.001, 0.00001, 0.05, 0.005, 0.00005]
Weight Decay	[0.005, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]
L1 Regularization	(0, 0.3)
L2 Regularization	(0, 0.3)
Normalization	Batch

Multiple forecasting horizons will be presented, as some performances of the models can differ depending on how far into the future we are predicting. The first and potentially most interesting is how good are the models in terms of predicting the next day realized volatility.

For the results section, two different datasets are considered

- SPDR SP 500 Trust ETF (SPY)
- USD/CHF Exchange Rate

3.1 Data: SPDR SP 500 Trust ETF

The sample size of daily realized volatility is 5247 observations and the wavelet decomposition is only performed up to scale $J = 2$, due to the machine learning models requiring bigger samples to train.

	Mean	**Median**	**Std. Dev.**	**Min**	**Max**	**Skewness**	**Kurtosis**
RV	4.69	4.01	3.05	1e-21	32.23	2.91	18.28

Table 3.2: Descriptive statistics. The original dataset has been scaled by 1000 for both visual and methodological convenience.

Model	MSE	MAE	R^2(%)
HAR	4.27	1.22	71.29
EW(9)	6.71	1.8	54.88
CMWLM	4.8	1.3	67.68
UMWLM	4.86	1.33	67.31
DWLM	4.19	1.22	71.79
DWNN	5.17	1.32	65.22
UMWNN	5.07	1.35	65.91

Table 3.3: The next day out-of-sample forecasting performance. MSE, MAE, and R^2 denote root mean square error, mean absolute error, and the coefficient of determination. The size of the testing set is 1500, which roughly corresponds to 5-6 years of daily observations. Lower MSE values indicate better predictive accuracy.

The updating frequency of the neural networks and the lags of the models can be found in appendix, Table 1.

From the models that were introduced in this study, the Direct Wavelet Linear Model is performing best in terms of predicting next-day realized volatility. In comparison to the HAR model, a slight improvement can be observed in terms of mean squared error (MSE). However, both models perform on par in terms of mean absolute error (MAE). Between the conditional and unconditional multistep methods, both models' performances are relatively close, both having the coefficient of determination (R^2) coefficient roughly 67% in the linear context and about 65% in the non-linear context.

Judging purely based on the results presented in Table 3.3, one can conclude that the DWT approaches, better unfold the dynamics behind the time series than the EW(9) (Extended Wald Decomposition model with 9 components) (Ortu et al., 2020).

As for the deep learning models, a Conditional Multistep Wavelet Neural Network wasn't added in the analysis, due to the failed convergence of the third model, which corresponds to modeling $j = 2$ wavelet coefficients at positions 3 and 4 (Fig. 2.4). The rate and overall trend of the training loss curve and validation loss curve can tell a lot about the generalization ability and stability of the model. Fig. 3.1 shows the convergence plots for 3 models that had to be tuned and trained. We observe a stable convergence in the first and the second graphs, but not the third (most right figure). Therefore, only the UMWNN (Unconditional Multistep Wavelet Neural Network) is evaluated. The introduction of the non-linearity via neural networks only outperforms the EW(9) and not the other benchmark HAR model.

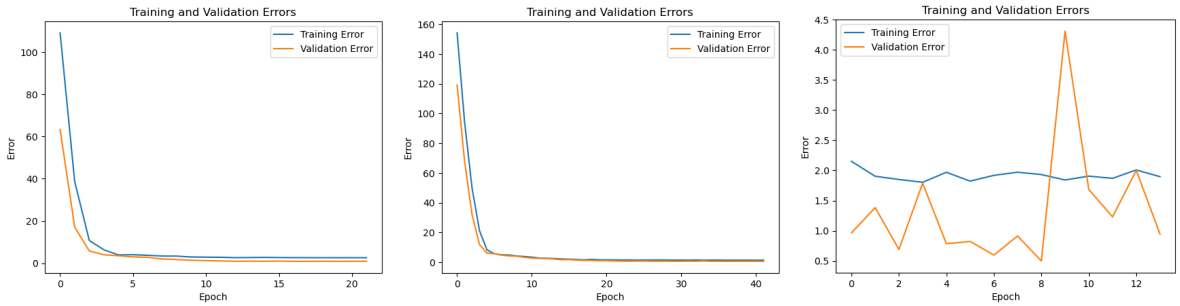


Figure 3.1: Convergence plots of the Conditional Multistep Wavelet Neural Network. The left plot is for the first regression that models the $j = 1$ coefficients, and the plot in the middle is for the second regression that models $j = 2$ coefficients corresponding to positions 1 and 2. The most right graph models $j = 2$ coefficients for positions 3 and 4.

The forecasting results are also evaluated across three distinct time horizons: 5 days, 20 days, and 60 days. Table 3.4 and Table 3.5 present the performance metrics, specifically MSE, MAE, and R^2. Table 3.4 compares the h-step ahead forecast with the actual value of the time series at that future time point. While Table 3.5 instead compares a forecast at a specific future time point with the average of the past few days leading up to that time point. The direct forecast is important as it measures the accuracy of the model in capturing the exact future values of the time series. Whereas rolling average focuses on assessing the trend and overall movement of the time series. It helps with understanding how well the forecasted values of the model align with the general behavior of the data over a given time horizon.

Metric	**Horizon**	**HAR**	**EW(9)**	**CMWLM**	**UMWLM**	**DWLM**	**DWNN**	**UMWNN**
	5-day	6.95*	9.06	6.99	7.23	7.59	8.37	7.71
MSE	20-day	10.15	11.7	9.91*	10.17	15.28	12.26	12.17
	60-day	14.70	15.99	13.67*	13.8	15.51	15.02	16.62
	5-day	1.53*	2.01	1.53*	1.57	1.59	1.67	1.6
MAE	20-day	1.84	2.17	1.76*	1.83	2.29	1.96	1.98
	60-day	2.30	2.43	2.02*	2.13	2.3	2.19	2.4
	5-day	53.35*	39.25	53.05	51.48	49.03	43.79	48.27
R2	20-day	32.37	22.08	33.92*	32.22	-1.85	18.31	18.87
	60-day	3.32	-5.2	10.09*	9.27	-2.02	1.23	-9.34

Table 3.4: The table displays the performance metrics values obtained from forecasting [5/20/60]-days ahead. A direct comparison to the true values is performed. The performance metrics are MSE, MAE, and R^2. The values with an asterisk next to them indicate the model with the best performance per each metric per horizon.

When forecasting longer horizons, we observe a decrease in the performance of the DWLM

(Direct Wavelet Linear Model) which performed well in forecasting next-day realized volatility. An interesting observation is that the predictive ability of CMWLM (Conditional Multistep Wavelet Linear Model) increases the longer the forecasting horizon is, with CMWLM taking over the place of the best-performing model as soon as the horizon is longer than 5 days when comparing MSEs. The other metrics also follow the previous statement.

The performances of DWNN and UMWNN when compared to the EW(9) vary a lot when comparing MSEs and R^2s, but not MAE, where we see that both models that use neural networks outperform EW(9). Therefore, no concise conclusion can be drawn.

Metric	Horizon	HAR	EW(9)	CMWLM	UMWLM	DWLM	DWNN	UMWNN
	5-day	3.29*	5.59	3.69	3.81	4.4	5.26	4.33
MSE	20-day	3.46*	5.44	3.64	3.65	10.71	6.91	5.24
	60-day	5.05	7.09	5.51	5.02*	8.93	7.48	7.01
	5-day	1.02*	1.63	1.07	1.11	1.18	1.28	1.15
MAE	20-day	1.07	1.63	1.05*	1.11	1.96	1.45	1.28
	60-day	1.32	1.79	1.28*	1.29	1.84	1.63	1.47
	5-day	73.23*	54.6	70.02	69.02	64.22	57.21	64.81
R2	20-day	66.87*	47.87	65.13	65.0	-2.64	33.71	49.77
	60-day	40.98	17.17	35.67	41.31*	-4.31	12.58	18.13

Table 3.5: The table displays the performance metrics values obtained from forecasting [5/20/60]-days ahead. The rolling average comparison is performed. The performance metrics are MSE, MAE and R^2. The values with an asterisk next to them indicate the model with the best performance per each metric per horizon.

As for the rolling average comparison (Table 3.5), the picture changes. With HAR model continues to dominate in terms of MSE for 5 days and 20 days ahead forecast and is slightly outperformed by the UMWLM (Unconditional Multistep Wavelet Linear Model) when forecasting 60 days ahead. Another observation is the improving performance of the UMWLM, being almost as good as CMWLM when forecasting 20 days ahead, and better when forecasting 60 days ahead in terms of MSE and R^2. This could be explained by the model's complexity, the conditional model involves more complexity with three linear models, and the unconditional model, with only two linear models, may be more robust and better at generalizing, leading to better performance in the rolling average comparison. We also observe that UMWNN outperforms the EW(9) in terms of all metrics.

It's important to test if the forecasted values provided by UMWNN model are significantly

better than the other predictions. The results of the two-sided Diebold-Mariano test show no significant difference between the predictions of DWLM being more accurate than HAR when forecasting next day realized volatility. With regards to the CMWLM which outperformed the HAR in longer horizons (direct comparison), the test only supports the claim that CMWLM beats HAR when horizon is 60 days. As for the rolling average comparison, there's not enough evidence to state UMWLM outperforms HAR when forecasting 60 days ahead.

	(a) Direct comparison				(b) Rolling average comparison			
Models / Horizon	1	5	20	60	1	5	20	60
HAR/EW(9)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
HAR/CMWLM	0.000	0.806	0.240	0.000	0.000	0.004	0.126	0.006
HAR/UMWLM	0.003	0.074	0.875	0.000	0.003	0.000	0.029	0.807
HAR/DWLM	0.271	0.074	0.000	0.011	0.271	0.000	0.000	0.000
HAR/DWNN	0.001	0.002	0.001	0.271	0.001	0.000	0.000	0.000
HAR/UMWNN	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.000
EW(9)/CMWLM	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EW(9)/UMWLM	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EW(9)/DWLM	0.000	0.000	0.000	0.150	0.000	0.000	0.000	0.000
EW(9)/DWNN	0.000	0.140	0.359	0.001	0.000	0.388	0.002	0.182
EW(9)/UMWNN	0.000	0.000	0.252	0.145	0.000	0.000	0.471	0.771
CMWLM/UMWLM	0.777	0.041	0.087	0.156	0.777	0.230	0.871	0.000
CMWLM/DWLM	0.000	0.011	0.000	0.000	0.000	0.000	0.000	0.000
CMWLM/DWNN	0.166	0.000	0.000	0.000	0.166	0.000	0.000	0.000
CMWLM/UMWNN	0.204	0.000	0.000	0.000	0.204	0.000	0.000	0.000
UMWLM/DWLM	0.001	0.238	0.000	0.000	0.001	0.024	0.000	0.000
UMWLM/DWNN	0.337	0.006	0.000	0.000	0.337	0.000	0.000	0.000
UMWLM/UMWNN	0.087	0.022	0.000	0.000	0.087	0.003	0.000	0.000
DWLM/DWNN	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DWLM/UMWNN	0.000	0.718	0.000	0.004	0.000	0.758	0.000	0.000
DWNN/UMWNN	0.724	0.077	0.869	0.000	0.724	0.001	0.000	0.028

Table 3.6: The results of the Diebold-Mariano Test. For a more convenient visualisation of the results, the values presented in the table are the probability values (p-value). Two-sided Diebold-Mariano Test has been performed to check if every combination of the models is significantly different from one another. The underlined values in the table are the tests with no rejection of the H_0 targeting the benchmarks as one of the models that is compared.

3.2 Data: USD/CHF Exchange Rate

The sample size of daily realized volatility created by Corsi (2009) contains 3599 observations and the wavelet decomposition is only performed up to scale $J = 2$, due to the machine learning models requiring bigger samples to train.

	Mean	Median	Std. Dev.	Min	Max	Skewness	Kurtosis
RV	12.397	11.593	3.900	3.828	41.947	1.594	5.071

Table 3.7: Descriptive statistics for the USD/CHF time series of daily realized volatility.

Model	MSE	MAE	R^2(%)
HAR	3.23	1.36	74.87
EW(9)	4.91	1.73	61.81
CMWLM	3.61	1.45	71.91
UMWLM	4.11	1.5	68.0
DWLM	3.36	1.39	73.83
DWNN	3.44	1.42	73.21
UMWNN	4.01	1.49	68.79

Table 3.8: The next day out-of-sample forecasting performance. MSE, MAE, and R^2 denote root mean square error, mean absolute error, and R^2. The size of the testing set is 520, which roughly corresponds to 2 years of daily observations. Lower MSE values indicate better predictive accuracy.

The lags of the models can be found in appendix, Table 2.

The first benchmark (HAR) outperforms all the other models with DWLM and DWNN being the closest when comparing all metrics. For the DWLM, similar behavior has been observed previously (SPDR SP 500 Trust ETF), which somewhat points out the stability of the direct forecasting methodology across datasets. The DWNN model demonstrates a notably superior predictive performance when applied to the exchange rate dataset, as compared to its performance on the SP500 dataset. As for the UMWNN model, the performance is relatively consistent. Lastly, based on the next-day forecasting results, both linear and non-linear methods are superior to the EW(9) model.

When looking at longer time horizons, comparing predictions directly to the true values at those time points (Table 3.9), the analysis yields compelling evidence of the UMWNN (Unconditional Multistep Wavelet Neural Network) model's superiority over established benchmarks and other

wavelet-based models. The multistep approach offers a nuanced advantage by affording the ability to capture and model the distinct characteristics of individual wavelet coefficients. By focusing on each coefficient individually, it's possible to capture the specific patterns thereby accommodating a more accurate representation of longer-term realized volatility dynamics.

Metric	Horizon	HAR	EW(9)	CMWLM	UMWLM	DWLM	DWNN	UMWNN
	5-day	4.85	6.54	5.29	4.97	8.7	6.46	4.61*
MSE	20-day	7.11	9.17	8.62	7.78	15.5	10.73	6.77*
	60-day	12.08	14.89	15.77	13.78	17.14	13.77	10.92*
	5-day	1.69	2.04	1.8	1.7	2.44	2.0	1.64*
MAE	20-day	2.14	2.49	2.42	2.26	3.34	2.62	2.04*
	60-day	2.85	3.24	3.37	3.13	3.51	3.13	2.56*
	5-day	62.05	48.82	58.59	61.09	31.87	49.44	63.89*
R2	20-day	43.0	26.48	30.89	37.65	-24.18	13.99	45.73*
	60-day	-11.11	-37.05	-45.13	-26.77	-57.74	-26.73	-0.48*

Table 3.9: The table displays the performance metrics values obtained from forecasting [5/20/60]-day ahead. A direct comparison to the true values is performed. The performance metrics are MSE, MAE, and R^2. The values with an asterisk next to them indicate the model with the best performance per each metric per horizon.

Metric	Horizon	HAR	EW(9)	CMWLM	UMWLM	DWLM	DWNN	UMWNN
	5-day	2.2*	4.01	2.85	2.41	6.41	4.18	2.21
MSE	20-day	2.81	5.0	4.54	3.58	11.91	6.99	2.45*
	60-day	5.1	7.65	9.36	7.43	11.53	8.29	4.01*
	5-day	1.19*	1.64	1.39	1.25	2.13	1.61	1.21
MAE	20-day	1.41	1.91	1.85	1.63	3.02	2.11	1.23*
	60-day	1.95	2.37	2.66	2.38	2.96	2.57	1.55*
	5-day	79.73*	63.09	73.71	77.77	40.91	61.53	79.62
R2	20-day	70.44	47.37	52.23	62.37	-25.22	26.52	74.2*
	60-day	32.38	-1.49	-24.06	1.44	-52.93	-9.95	46.76*

Table 3.10: The table displays the performance metrics values obtained from forecasting [5/20/60]-day ahead. The rolling average comparison is performed. The performance metrics are MSE, MAE and R^2. The values with an asterisk next to them indicate the model with the best performance per each metric per horizon.

The results of hypothesis testing, indicate there's enough statistical evidence to signify the performance of the UMWNN over HAR, in forecasting 60 days ahead when making direct comparison of the forecasts with the true values of that day. As for the rolling average comparison,

there's significant difference when making 20 and 60 days ahead predictions.

The results support the claim that the UMWNN better accomodates a more accurate representation of longer-term realized volatility dynamics.

	(a) Direct comparison				(b) Rolling average comparison			
Models / Horizon	1	5	20	60	1	5	20	60
HAR/EW(9)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
HAR/CMWLM	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000
HAR/UMWLM	0.000	<u>0.188</u>	0.000	0.000	0.000	0.001	0.000	0.000
HAR/DWLM	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000
HAR/DWNN	0.009	0.000	0.000	0.000	0.009	0.000	0.000	0.000
HAR/UMWNN	0.000	<u>0.063</u>	<u>0.064</u>	0.000	0.000	<u>0.884</u>	0.000	0.000
EW(9)/CMWLM	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EW(9)/UMWLM	0.007	0.000	0.000	0.000	0.007	0.000	0.000	0.060
EW(9)/DWLM	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EW(9)/DWNN	0.000	<u>0.756</u>	0.000	0.002	0.000	<u>0.449</u>	0.000	0.012
EW(9)/UMWNN	0.003	0.000	0.000	0.000	0.003	0.000	0.000	0.000
CMWLM/UMWLM	0.012	0.000	0.000	0.000	0.012	0.000	0.000	0.000
CMWLM/DWLM	0.018	0.000	0.000	0.000	0.018	0.000	0.000	0.000
CMWLM/DWNN	0.196	0.000	0.000	0.000	0.196	0.000	0.000	0.000
CMWLM/UMWNN	0.033	0.000	0.000	0.000	0.033	0.000	0.000	0.000
UMWLM/DWLM	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000
UMWLM/DWNN	0.003	0.000	0.000	0.985	0.003	0.000	0.000	0.000
UMWLM/UMWNN	0.241	0.001	0.000	0.000	0.241	0.010	0.000	0.000
DWLM/DWNN	0.368	0.000	0.000	0.000	0.368	0.000	0.000	0.000
DWLM/UMWNN	0.002	0.000	0.000	0.000	0.002	0.000	0.000	0.000
DWNN/UMWNN	0.008	0.000	0.000	0.000	0.008	0.000	0.000	0.000

Table 3.11: The results of the Diebold-Mariano Test. For a more convenient visualisation of the results, the values presented in the table are the probability values (p-value). Two-sided Diebold-Mariano Test has been performed to check if every combination of the models is significantly different from one another. The underlined values in the table are the tests with no rejection of the H_0 targeting the benchmarks as one of the models that is compared.

Chapter 4

Conclusion

The analysis performed in this thesis, revolves around using Discrete Wavelet Transform, with the choice of the wavelet being inspired by Bandi et al. (2019). The orthogonality property of the wavelet matrix (Eq. 1.7) allows to perfectly reconstruct the time series using the wavelet coefficients. This gives rise to two methodologies introduced in this thesis, being the direct approach (section 2.3) and the multistep approach (section 2.4).

The direct approach regresses the realized volatility time series directly on the lags of the wavelet coefficients and the lags of the realized volatility itself. While the multistep approach allows to model the dynamics of the individual components that can then be assembled to make provide a forecast of the time series. Two branches of the multistep methodology are created with the first being the conditional multistep model and the unconditional multistep model. The former utilizes the cycle structure of the components, using two separate models that estimate $j = 2$ coefficients depending on the positions within the cycles. While the ladder relaxing the cycle structure idea and simply uses the same coefficients/weights to forecast $j = 2$ coefficients independent of the position.

The empirical application favors the multistep approach when forecasting longer time horizons. In the first dataset, exploration with linear regression unveiled the prowess of the multistep approach, as it efficiently estimated wavelet coefficients and adeptly assembled them to produce accurate forecasts. In the exchange rate dataset, the neural network-powered multistep approach outperformed other methods indicating its ability to capture nonlinear dynamics in the realized volatility time series.

This outcome underscores the adaptability of the multistep technique, which, when coupled with different modeling methods, showcases a capability to unravel individual component dynamics

and forecast volatility.

Further research could expand upon the current study by exploring the application of more advanced and complex machine learning models to further enhance the predictive prowess of the proposed multistep and direct approaches for realized volatility forecasting.

References

Andersen, T. G. & Bollerslev, T. (1998). Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International Economic Review*, *39*(4), 885–905. Retrieved 2023-04-14, from http://www.jstor.org/stable/2527343

Bandi, F., Perron, B., Tamoni, A. & Tebaldi, C. (2019). The scale of predictability. *Journal of Econometrics*, *208*(1), 120-140. Retrieved from https://www.sciencedirect.com/science/article/pii/S0304407618301738 (Special Issue on Financial Engineering and Risk Management) doi: https://doi.org/10.1016/j.jeconom.2018.09.008

Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *The Journal of Financial Econometrics*, *7*(2), 174-196. Retrieved from https://EconPapers.repec.org/RePEc:oup:jfinec:v:7:y:2009:i:2:p:174-196

Ortu, F., Severino, F., Tamoni, A. & Tebaldi, C. (2020). A persistence-based wold-type decomposition for stationary time series. *Quantitative Economics*, *11*(1), 203-230. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.3982/QE994 doi: https://doi.org/10.3982/QE994

Reisenhofer, R., Bayer, X. & Hautsch, N. (2022). *Harnet: A convolutional neural network for realized volatility forecasting.*

Appendix

Appendix A: Data: SPDR SP 500 Trust ETF

Model	$\hat{\pi}^{(1)}$	$\hat{x}^{(1)}$	$\hat{\pi}_a^{(2)}$	$\hat{x}_a^{(2)}$	$\hat{\pi}_b^{(2)}$	$\hat{x}_b^{(2)}$	x
CMWLM	1	4	1	5	1	4	
UMWLM	1	4	1	5	1	4	
DWLM	7	3	2	50			2
DWNN	30	30	30	30			30
UMWNN	30	30	30	30			

Table 1: The lag structure of the models. The values in the table are the number of lags per each component and the number of lags of the original time series. The columns $\hat{\pi}_a^{(2)}$ and $\hat{x}_b^{(2)}$ are the number of lags in the conditional multistep approach where two separate models are used to forecast $j = 2$ coefficients. Subscript a is for the wavelet coefficients in position 1 and 2, subscript b is for the wavelet coefficients in position 3 and 4.

Updating frequencies,

- DWNN: 66 days
- UMWNN: 44 days

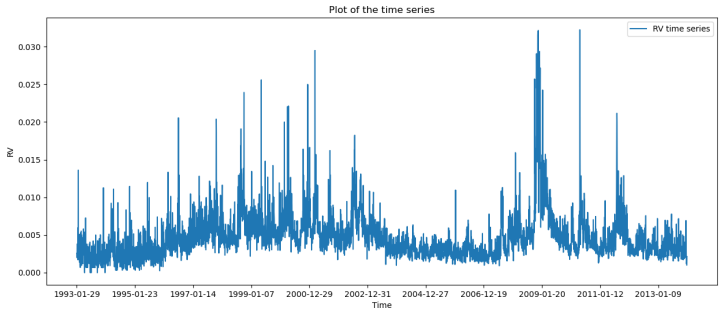
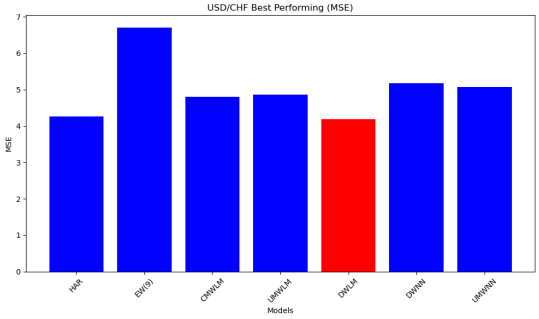
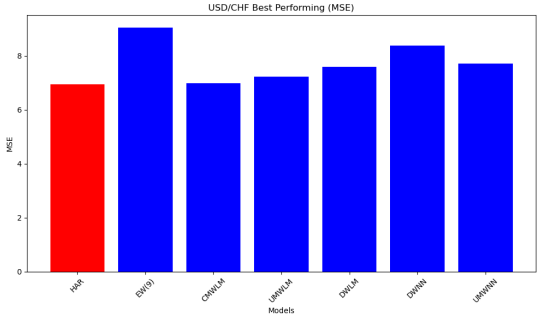


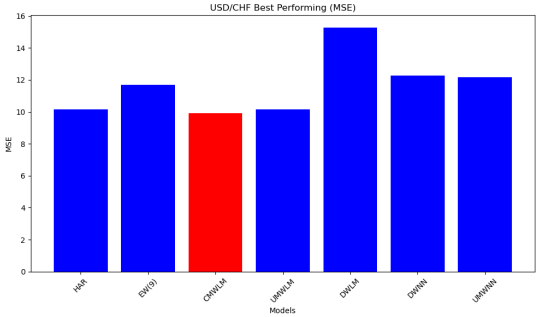
Figure 1: Time series plot.



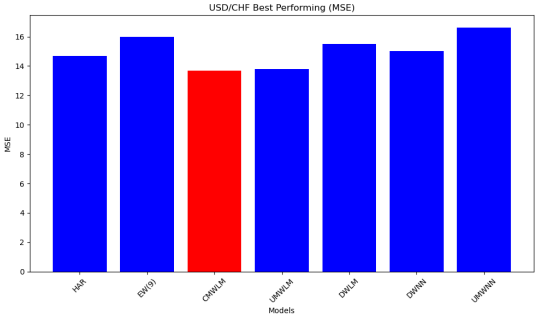
(a) MSE comparison bar plot for horizon = 1



(b) MSE comparison bar plot for horizon = 5



(c) MSE comparison bar plot for horizon = 20



(d) MSE comparison bar plot for horizon = 60

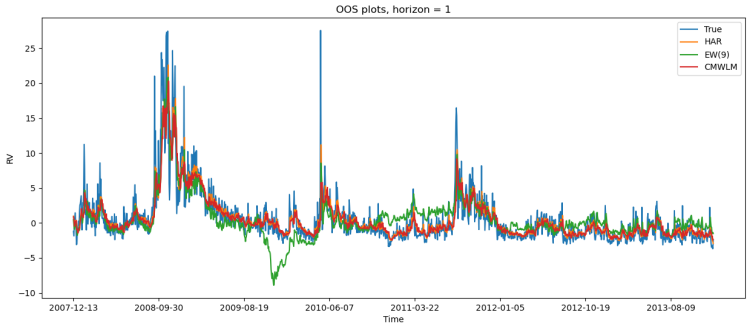
Data: USD/CHF Exchange Rate

Model	$\hat{\pi}^{(1)}$	$\hat{x}^{(1)}$	$\hat{\pi}_a^{(2)}$	$\hat{x}_a^{(2)}$	$\hat{\pi}_b^{(2)}$	$\hat{x}_b^{(2)}$	x
CMWLM	1	3	1	2	1	5	
UMWLM	1	3	1	2	1	5	
DWLM	1	50	9	4			2
DWNN	30	30	30	30			30
UMWNN	30	30	30	30			

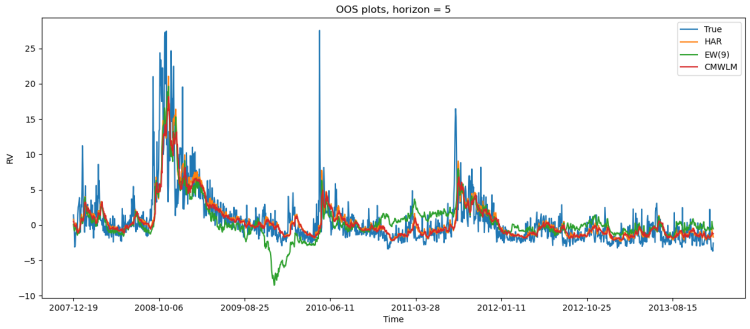
Table 2: The lag structure of the models. The values in the table are the number of lags per each component and the number of lags of the original time series. The columns $\hat{\pi}_a^{(2)}$ and $\hat{x}_b^{(2)}$ are the number of lags in the conditional multistep approach where two separate models are used to forecast $j = 2$ coefficients. Subscript a is for the wavelet coefficients in position 1 and 2, subscript b is for the wavelet coefficients in position 3 and 4.

Updating frequencies,

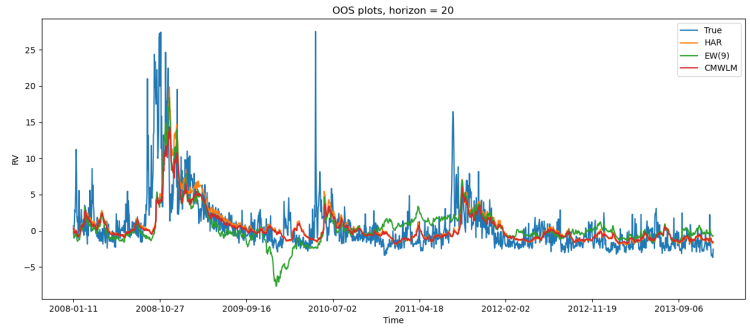
- DWNN: 44 days
- UMWNN: 44 days



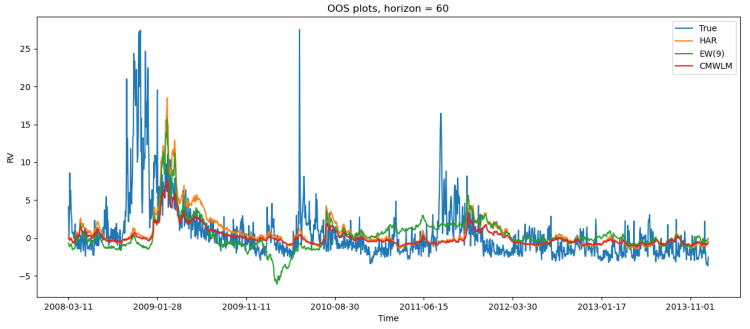
(a) Plot of the forecasts. Horizon = 1



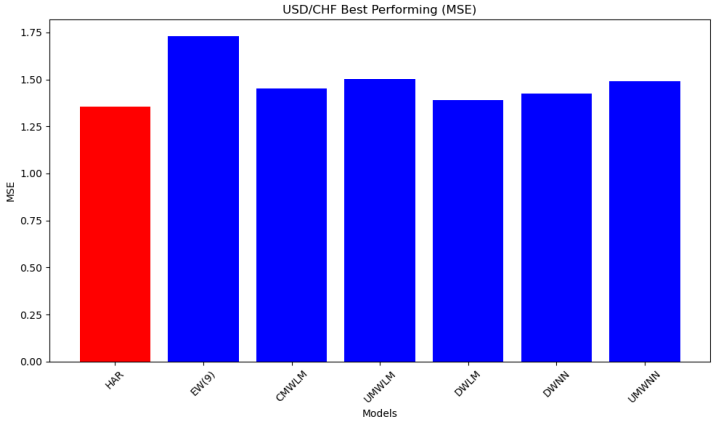
(b) Plot of the forecasts. Horizon = 5



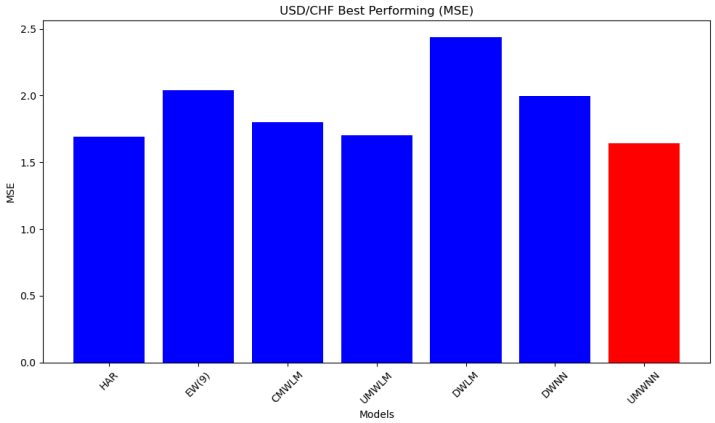
(c) Plot of the forecasts. Horizon = 20



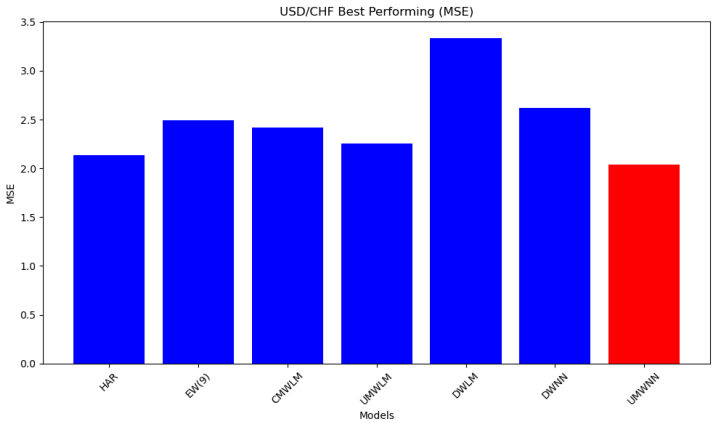
(d) Plot of the forecasts. Horizon = 60



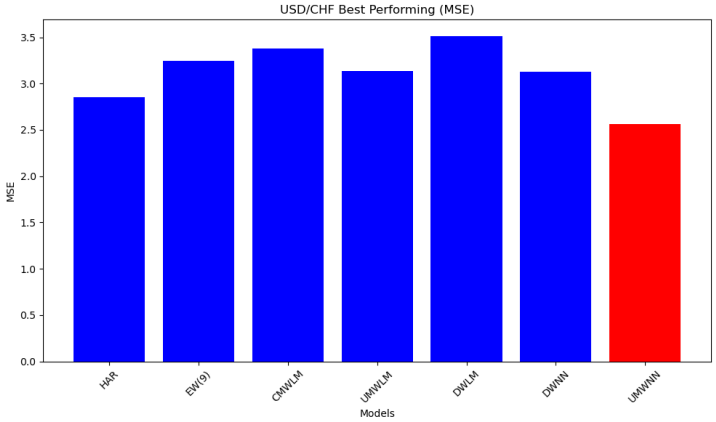
(a) MSE comparison bar plot for horizon = 1



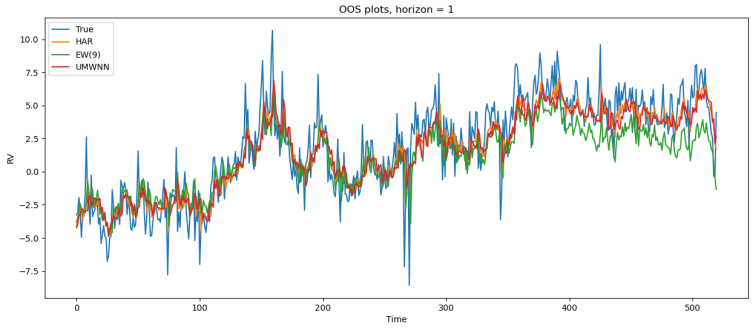
(b) MSE comparison bar plot for horizon = 5



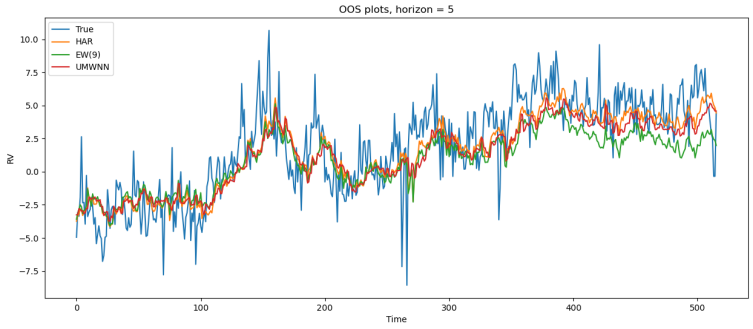
(c) MSE comparison bar plot for horizon = 20



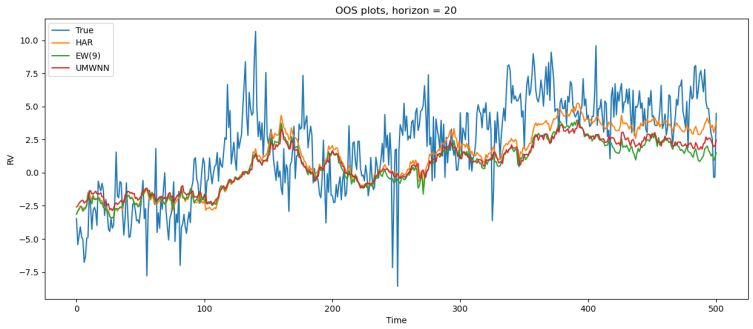
(d) MSE comparison bar plot for horizon = 60



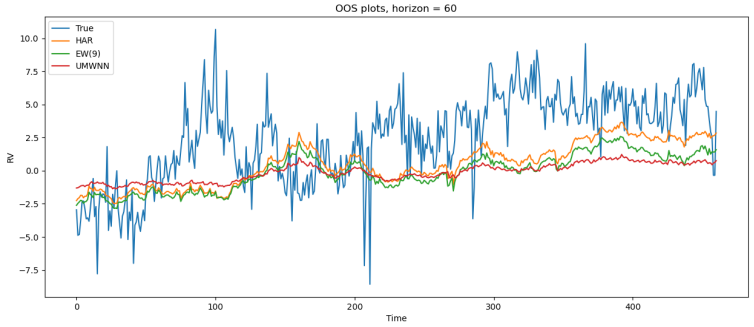
(a) Plot of the forecasts. Horizon = 1



(b) Plot of the forecasts. Horizon = 5



(c) Plot of the forecasts. Horizon = 20



(d) Plot of the forecasts. Horizon = 60

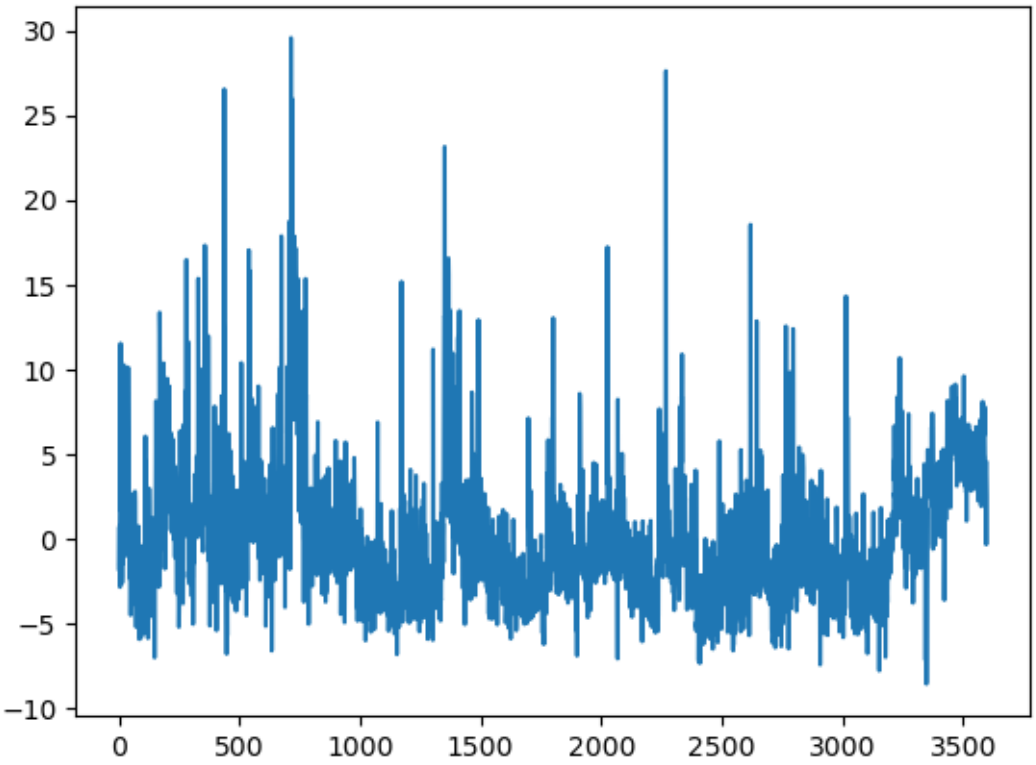


Figure 6: The original time series of daily realized volatility.