Erasmus University Rotterdam

Erasmus School of Economics

Bachelor thesis Econometrie en Operationele Research

Volatility analysis of Bitcoin Price

Name student
Johan (J.C.) Verschoor
Student ID number
535164

Supervisor
Dr. Rutger-Jan (R.) Lange
Second assessor
Prof. dr. Dick (D.J.C.) van Dijk

Date final version: July 1, 2022

Contents

Abstract

In this paper, I use Bitcoin price data from January 2015 till April 2022 to assess whether the Bitcoin volatility predictions by the heterogeneous autoregressive model for realized volatility including jumps (HARRVJ) can be improved by adding variables or translating the model to a neural network. Better predictions of the Bitcoin volatility enable participants to make better risk assessments in the highly volatile cryptomarket. To my knowledge, Bitcoin volatility has not yet been predicted with a neural network fed by HARRVJ-components, and newer data is used. The out-of-sample predictions are compared using six loss functions, and Diebold Mariano and Clark West test statistics. I find that including a leverage effect by means of the negative realized semivariance improves the in- and out-of-sample performance when forecasting Bitcoin volatility, but that the comparison with the neural network did not give significant results. This indicates that negative and positive returns influence the volatility of Bitcoin differently.

1 Introduction

Online payments have been dependent on financial institutions as trusted parties for handling transactions. Nakamoto (2008) proposed Bitcoin as a network where two parties can exchange electronic cash without needing a financial institution. Since then, other cryptocurrencies like Ethereum, XRP and Ripple have been set up. Most research on cryptocurrencies focuses on Bitcoin however, since it has the biggest market capitalization of all cryptocurrencies.[1] The last years, the cryptomarket has been rapidly growing. A significant aspect of the cryptomarket, when comparing it with the stock market, is its high volatility.

Especially in this highly volatile market, proper risk assessments are of interest to the participants. To construct these risk assessments, good predictions of the volatility are crucial. In traditional research, the benefit of working with realized volatility (RV) when modeling volatility is recognized (Andersen et al.,2000). These can be implemented by using a heterogeneous autoregressive (HAR) model. Expansions, like the heterogeneous autoregressive model for realized volatility including jumps (HARRVJ), as developed by Andersen et al. (2007) implement the possibility of structural breaks. Pichl and Kaizoji (2017) use the HARRVJ model on Bitcoin volatility, and find that it applies reasonably well. Shen et al. (2020) implement even more heterogeneous autoregressive realized volatility (HARRV) models to study the volatility of Bitcoin and find that modelling jumps 'significantly improves the accuracy of volatility forecasts of Bitcoin through popular HARRV models'.

[1] On the 30^{th} of June 2022, the market capitalization of Bitcoin is 363 billion dollars, whereas the total cryptocurrency market capitalization is 863 billion dollar.

I am interested in the in-sample and out-of-sample performance of the HARRVJ model on more recent Bitcoin data. Therefore, I want to consider the following research question: *Can the (predictive) performance of the heterogeneous autoregressive model for realized volatility including jumps (HARRVJ) on Bitcoin volatility be improved?* I assess the results of adding other linear components to the HARRVJ model, like the realised semivariance (Patton and Sheppard,2015) or realized quarticity (Bollerslev et al.,2016). This results in 8 HARRV-models, of which I compare the in-sample and predictive performance. Bitcoin volatility forecasts have also been constructed combining GARCH-models with neural networks (Seo and Kim,2020). I implement a machine learning approach, using the HARRVJ model, and verify whether this improves the linear HARRVJ model.

The data used is historical price data from the Bitstamp exchange market, ranging from 1 January 2015 till 30 April 2022, thus consisting of 2677 days. Following Andersen and Bollerslev (1998), realized measures are calculated using 5-minute frequency logarithmic returns. To construct these over the full sample, 770976 price observations were used.

To compare linear HARRV models, a general model is introduced that includes realized volatility variables, negative semivariance variables, jump components and realized quarticity components. Leaving out combinations of these components generates 8 nested models. These models are named HARRV, HARRSV, HARRVJ, HARRSVJ, HARRVQ, HARRSVQ, HARRVJQ and HARRSVJQ. Their in-sample performance is measured by AIC, BIC and adjusted R^2 measures, and their predictive performance by loss functions proposed by Hansen and Lunde (2005). To determine whether combinations of non-nested model outperform each other significantly, the Diebold Mariano (DM) test is used (Diebold and Mariano,1995), whereas the adjustment of Clark and West (2007) to the DM test is used for nested models.

To test a non-linear approach, a neural network containing two hidden layers is introduced. It takes the same input as a HARRVJ model, and is trained with the first two-third of the sample to predict the realized volatility. Its predictive performance is compared with the linear HARRVJ model, using the loss functions of Hansen and Lunde (2005) and the mentioned DB and CW tests.

Regressing the linear HAR-models over the full sample, I find that including realized semivariance and realized quarticity increased the in-sample fit. The HARRSVQ had the lowest BIC value, whereas the HARRSVJQ model had the lowest AIC and highest adjusted R^2 value. I also compare one-day-ahead predictions using a moving window of 1 year, and find that both the HARRSV and HARRSVJ, which both include negative semivariance, have the lowest loss function value for three of the six loss functions. The results of the CW and DB test also show that

models including negative semivariance often outperform other models. Out of the models that do not include negative semivariance, the HARRVJQ model which includes jump components and realized quarticity performs significantly better.

For the neural network, I find that it had lower loss function values for 5 of the 6 loss function values when compared to the HARRVJ model. Although a negative DM test statistic indicates better predictive performance of the neural network, it is not significant.

In sum, including a leverage effect by means of the negative realized semivariance to a HARRVJ model improves its in-sample and out-of-sample performance when forecasting Bitcoin volatility. Feeding a neural network with the variables of a HARRVJ model seemed to give better predictive performance, but this improvement was not significant.

Section 2 discusses the existing literature on linear and non-linear volatility forecasts for Bitcoin, and section 3 describes the used data. Section 4 describes the used HAR-models and the used neural network, and how their in- and out-of-sample performance is measured. Section 5 5 presents the results of both the in- and out-of-sample setting. Section 6 interprets the findings and concludes with some suggestions for further research.

2 Literature

This section discusses the existing literature on volatility forecasts, specifically for Bitcoin. Volatility forecasts based on linear models are discussed in section 2.1. The use of non-linear forecasts like neural networks for volatility is discussed in section 2.2.

2.1 Linear models for volatility forecasting

It is a well-known concept in the financial world that returns of assets are hard to predict, whereas their volatility is more predictable. The problem is however that the conditional variance can only be estimated, but is not observed. A well-known model for the variance of time series is the (generalized) autoregressive conditional heteroskedasticity (GARCH) model (Bollerslev, 1986).

Empirically, it has been shown that most latent volatility models do not manage to replicate all stylized facts of financial returns (McAleer and Medeiros,2008). Squared returns have for example low, slowly decreasing autocorrelations, which are not described satisfactory by most standard models. As a result, standardized residuals are often non-normal (Carnero et al., 2004). This decreases the accuracy of the volatility forecasts. Using realized volatility could reduce this problem. Andersen et al. (2000) mention the benefits of using realized volatility when working with exchange data. They conclude that a return distribution standardized by realized volatilites approaches the normal distribution better then using GARCH-estimated volatilities

to standardize.

This lead to the introduction of the heterogeneous autoregressive model for realized volatility (HARRV) by Corsi (2009). This simple AR-type model only uses realized volatilities, but reproduces many of the main stylized facts of financial data. About the construction of realized volatility variables, Andersen and Bollerslev (1998) point out that increasing the price observation frequency to an infinitely small interval theoretically makes the latent volatility observable. However, this is not possible because of micro-structure features, like bid-ask spreads and infrequent trading. Therefore, they conclude that the ex post volatility is best measured using a 5 minute return data frequency.

There has also been growing interest in the importance of jumps in financial time series. Barndorff-Nielsen and Shephard (2004) developed tests for detecting jumps in high-frequency data. They measure variance with two components: One captures the effect of the jumps, while the other component is robust to the effect of the jumps. Andersen et al. (2007) builds further upon this, by introducing the heterogeneous autoregressive model for realized volatility including jumps (HARRVJ). It estimates the return volatility by using a number of past realized volatilities and 'jump components'. These jumps are found to be highly important as they give out-of-sample volatility forecast improvement.

Bergsli et al. (2022) stress that HAR models in general outperform GARCH models when it comes to modelling Bitcoin volatility. They assign this to the fact that realized variance has a higher autocorrelation than squared returns. The HARRVJ model is implemented on Bitcoin price data by Pichl and Kaizoji (2017), who find significant lag and jump coefficients. Shen et al. (2020) research 18 HAR-models on high-frequency Bitcoin data. They find that including jumps gives more in-sample explanatory power, and improves the predictive value as well.

I assess whether jump components are still improving the in- and out-of-sample performance when forecasting Bitcoin volatility, using new Bitcoin price data. Furthermore, I add other variables to the HARRV model to assess whether these result in a better in-sample fit and predictive performance as well.

2.2 Non-linear models for volatility forecasting

Apart from the linear models mentioned in section 2.1, plenty of non-linear models have been proposed to predict volatility. The advantage is that these non-linear models are more flexible, and might have greater forecast predictability. The drawbacks of these non-linear models however is that they might be hard to compute, are not easily interpretable and can overfit the data (White,2006). A neural network is an example of a model formulation that supports highly

nonlinear approximation. These models have found plenty of applications in finance, for example in predicting bankruptcy of firms, trading of bonds and the construction of stock market volatility forecasts (Wong and Selvi,1998).

Kristjanpoller et al. (2014) use a neural network to forecast volatility of three Latin-American stock exchange indexes. Amongst other variables, they feed this neural network with realized volatility and GARCH(1,1) output. They find that this neural network has better predictive performance than a GARCH(1,1) model. Donaldson and Kamstra (1997) use neural networks to forecast the stock return volatility in New York and London. They find that the neural network outperforms GARCH, EGARCH and GJR models, since the neural network is better able to capture asymmetric effects of returns. This shows that predicting volatility with neural networks has shown satisfactory results.

When it comes to neural network implementations on Bitcoin, the focus is mostly on forecasting returns rather than volatility. Pichl and Kaizoji (2017) implement a two-layer feed-forward neural network on Bitcoin return data, and find that extreme event clustering is approximated quite well. McNally et al. (2018) compare even more advanced deep learning methods using Bitcoin price data, and find that the autoregressive integrated moving average (ARIMA) model is clearly outperformed by the non-linear deep learning methods.

Seo and Kim (2020) are one of the few researchers that use a neural network to analyse Bitcoin's volatility. They use advanced neural networks, fed by GARCH output and some other variables, like the volatility index (VIX). They find that these models are appropriate for forecasting Bitcoin volatility, and show good predictive results.

I use the HARRVJ input to train a neural network, and assess whether the predictive results of the neural network are better than the HARRVJ results. This has not yet been done, and sheds further light on the use of neural networks on Bitcoin volatility.

3 Data

The data are retrieved from the crypto-datadownload web[2]. I use historical price data from the Bitstamp exchange market, expressing the value of Bitcoin in US Dollars (BTCUSD). The sample ranges from 1 January 2015 till 30 April 2022, therefore consisting of 2677 days. The data is exported daily, which generates 2677 observations.

This data is also needed at a 5 minute frequency, to construct observed realized volatilities, which generates 770976 observations. The data originally is provided at a 1 minute frequency, the sets of observations are created by leaving out the unnecessary observations. The original

[2]This data can be assessed via https://www.cryptodatadownload.com/data/bitstamp/

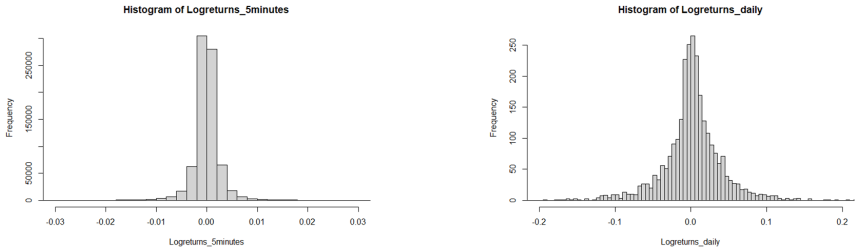
data had 9 missing observations, namely the price observation of the first minute of every year, and the price level at 23-6-2016, 12:36:00. To still get 5 minute-frequency data, only the closing prices at 00:04, 00:09, etc. are taken.

The logarithmic returns are then defined as $R_t = \log\left(\frac{B_t}{B_{t-1}}\right)$. Here, B_t stands for the Bitcoin price level at time step t.

Table 1 and figure 1 show that the logarithmic returns of both the 5 minute and the daily returns are not normally distributed. The skewness is negative for both return frequencies, meaning that large negative outliers are more common than large positive outliers. The low minima and high maxima, together with the high kurtosis, show that the observations contain outliers. It is noticeable that the median of the 5-minute log returns is exactly 0. This is because quite some 5 minute intraperiods showed no change in price level, especially in the earlier observations in 2015 and 2016. Both return observations have a positive mean, meaning that Bitcoin has increased in price level over the past 7 years.

	# observations	**Minimum**	**Maximum**	**Median**	**Mean**	**Std. dev.**	**Kurtosis**	**Skewness**
5-minute log-returns	770976	-0.208	0.085	0.000	$5.910 * 10^{-6}$	0.0027	122.4	-1.274
Daily log returns	2676	-0.491	0.218	0.0019	0.0017	0.040	15.00	-0.856

Table 1: Summary statistics of the logarithmic returns of Bitcoin, from 1 January 2015 till 30 April 2022



(a) 5 minute log returns

(b) Daily log returns

Figure 1: Histograms of the 5 minute log returns, and the daily log returns for Bitcoin from 1 January 2015 till 30 April 2022

4 Methodology

Section 4.1 describes how the realized volatility is constructed and forecast with linear HARRV models, and section 4.2 works out the used neural network. Section 4.3 describes how the in-sample performance of models is compared, and section 4.4 describes how the predictive performance is compared.

4.1 Modelling and forecasting realized volatility

The heterogeneous autoregressive model for realized volatility including jumps is worked out in section 4.1.1. Section 4.1.2 provides a more general HAR-model to forecast the realized variance.

4.1.1 The HARRVJ model

For each day t in the sample, realized volatility can be constructed by using high-frequency return data:

$$RV_t = \sum_{i=1}^{n_t} r_{t,i}^2, \tag{4.1}$$

where in this case and from here forward, n_t denotes the amount of observations on day t, and $r_{t,i}$ denotes the i^{th} intraperiod return of day t.

To differentiate between a continuous component of the realized variance and jumps, the realized bi-power variation (RBPV) measure proposed by Barndorff-Nielsen and Shephard (2004), is necessary as well. It is given by

$$RBPV_t = \delta_1 \sum_{i=3}^{n_t} |r_{t,i}||r_{t,i-1}|. \tag{4.2}$$

As mentioned, n_t denotes the amount of observations on day t, and $r_{t,i}$ denotes the ith intraperiod return of day t from here forward. δ_1 denotes the mean of the absolute value of the standard normal random variable Z, which is given by $\pi/2$.

To test whether certain jumps are significant, the realized tripower quarticity (RTQ) is used, which is studied by Barndorff-Nielsen and Shephard (2004). It is denoted by:

$$RTQ_t = n_t \delta_{4/3}^3 \sum_{i=3}^{n_t} |r_{t,i}|^{\frac{4}{3}} |r_{t,i-1}|^{\frac{4}{3}} |r_{t,i-2}|^{\frac{4}{3}}, \tag{4.3}$$

where $\delta_{\frac{4}{3}}$ denotes the mean of the absolute value of the standard normal random variable $Z^{4/3}$.

The jumps at day t with significance level α are defined by Andersen et al. (2007), as

$$J_{t,\alpha} = (RV_t - RBPV_t) * \mathbb{1}(ZJ_{RBPV}(t) > \Phi_\alpha), \tag{4.4}$$

where RV_t and $RBPV_t$ are respectively given by functions 4.1 and 4.2. The significance level α is taken as 0.05, and Φ_α is its critical value from the normal distribution. $ZJ_{BPV}(t)$ is the adjusted jump ratio statistic at day t from Huang and Tauchen (2005), which has test statistic

$$ZJ_{RBPV}(t) = \frac{(RV_t - RBPV_t)RV_t^{-1}}{\left(\theta \max\{1, RTQ_t RBPV_t^{-2}\}\right)^{\frac{1}{2}}}, \tag{4.5}$$

where RV_t, $RBPV_t$ and RTQ are given by functions 4.1, 4.2 and 4.3 respectively, and $\theta = \frac{\pi^2}{4} + \pi - 5$.

Replicating Pichl and Kaizoji (2017), the heterogeneous autoregressive model for realized volatility including jumps (HARRVJ) constructed by Andersen et al. (2007) is used to forecast Realized Variance. It is defined as

$$\sqrt{RV_{t+1}} = \beta_0 + \beta_1\sqrt{RV_t} + \beta_2\sqrt{RV_{t-4}} + \beta_3\sqrt{RV_{t-9}} + \beta_4\sqrt{J_t} + \beta_5\sqrt{J_{t-4}} + \beta_6\sqrt{J_{t-9}} + \epsilon_{t+1}, \quad (4.6)$$

where the sample horizons are chosen to be daily, 5 days back and 10 days back. The (1,5,10)-day parameter selection, which deviates from the suggestion of Andersen et al. (2007), is motivated by the finding of Pichl and Kaizoji (2017) that β_3 becomes significant when choosing 10 days rather than 22 for Bitcoin data. Here, RV_{t-i} stands for the average realized variance between day $t-i$ and t and is therefore defined by

$$RV_{t-i} = \frac{\sum_{j=0}^{i} RV_{t-j}}{i+1}. \quad (4.7)$$

J_{t-i} stands for the average jump between day $t-i$ and t, and is therefore defined by

$$J_{t-i} = \frac{\sum_{j=0}^{i} J_{t-j}}{i+1}, \quad (4.8)$$

where the daily jumps are defined by formula 4.4. ϵ_{t-1} denotes the error term for the square root of the realized variance at day $t+1$. This regression is implemented by using the highfrequency R-package from Boudt et al. (2021). It is regressed by using ordinary least squares, and makes use of Newey-West standard errors.

4.1.2 Alternative RV forecasting models

Pichl and Kaizoji (2017) suggest the use of a square root transformation on the HARRVJ model. Alternatives to the model in equation 4.6 are excluding the jump components, including realized quarticities and including a leverage effect on the realized variance component. The alternatives will be discussed in section 4.1.2.1 till 4.1.2.3. A general model is presented in 4.1.2.4.

4.1.2.1 Leaving out the jump components To verify that the jump components improve the model, the HARRVJ model as defined in formula 4.6 can be compared with the heterogeneous autoregressive realized variance (HARRV), as proposed by Corsi (2009). In spite of the simplicity of the model, it gives good forecasting results for exchange rates and stock market indices.

4.1.2.2 Including realized quarticity Some HAR-model implement the realized quarticity, which is given by

$$RQ_t = \frac{n_t}{3}\sum_{i=1}^{n_t} r_{t,i}^4. \quad (4.9)$$

Again, the average of this measure can be taken over the past i days, resulting in RQ_{t-i} which is given by

$$RQ_{t-i} = \frac{\sum_{j=0}^{i} RQ_{t-j}}{i+1}. \tag{4.10}$$

Adding RQ, RQ_{t-4} and RQ_{t-9} to the model results in a HARQFJ model, proposed by Shen et al. (2020). They compare the performance of 18 HAR-models on Bitcoin, and find that the HARQFJ performs very well. It outperforms almost all other HAR-models significantly for the in-sample analysis. It also has the best out-of-sample performance.

4.1.2.3 Including a leverage effect Another variation on the HARRVJ model, is including a leverage effect by splitting up the realised variance into negative realized variance and positive realized variance, as suggested by Barndorff-Nielsen et al. (2008). The negative daily realized variance and positive daily realized variance are respectively given by

$$RSV_t^- = \sum_{i=1}^{n_t} \left(r_{t,i}^2 * \mathbb{1}_{|r_{t,i}<0|} \right), \tag{4.11}$$

$$RSV_t^+ = \sum_{i=1}^{n_t} \left(r_{t,i}^2 * \mathbb{1}_{|r_{t,i}\geq 0|} \right). \tag{4.12}$$

Note that $RV_t = RSV_t^- + RSV_t^+$. Shen et al. (2020) find that the decomposition of the RV into a positive and negative component for Bitcoin leads to significant components for multiple HAR-models. This indicates that negative and positive Bitcoin returns indeed have different impacts on volatility.

Implementing these semivariances as an extension to the HARRVJ model, results in the heterogeneous autoregressive realized semi-variance model including jumps (HARRSVJ), proposed by Chen and Ghysels (2011). They claim that models featuring asymmetries have better forecasting abilities than models that do not.

4.1.2.4 A general model A general, new model, combining the existing models as given in sections 4.1.2.1 till 4.1.2.3, is now given by:

$$\begin{aligned}\sqrt{RV_{t+1}} =& \beta_0 + \beta_{11}\sqrt{RV_t} + \beta_{12}\sqrt{RSV_t^-} + \beta_{21}\sqrt{RV_{t-4}} + \beta_{22}\sqrt{RSV_{t-4}^-} + \\ & \beta_{31}\sqrt{RV_{t-9}} + \beta_{32}\sqrt{RSV_{t-9}^-} + \beta_4\sqrt{J_t} + \beta_5\sqrt{J_{t-4}} + \beta_6\sqrt{J_{t-9}} + \\ & \beta_7\sqrt{RQ_t} + \beta_8\sqrt{RQ_{t-4}} + \beta_9\sqrt{RQ_{t-9}} + \epsilon_{t+1},\end{aligned} \tag{4.13}$$

where the realized variance component of the past i days RV_{t-i} is given by equation 4.7 and the realized negative semivariance RSV_t^- is given by equation 4.11. RSV_{t-i}^- is then given by the average of the realized negative semivariance of the past i days. The jump components of the past i days is given by equation 4.9, and the quarticity components of the past i days is given by equation 4.10. ϵ_{t-1} denotes the error term for the square root of the realized variance at day $t+1$.

In total, I consider including or excluding jump components, including or excluding realized quarticity, and including or excluding a leverage effect. This generates 8 models, which are specified in table 2. They are presented as restricted forms of the general HARRSVJQ formulation in equation 4.13.

Model	**Restrictions**	**Reference**
HARRV	$\beta_{12} = \beta_{22} = \beta_{32} = \beta_4 = \beta_5 = \beta_6 = \beta_7 = \beta_8 = \beta_9 = 0$	Corsi (2009)
HARRSV	$\beta_4 = \beta_5 = \beta_6 = \beta_7 = \beta_8 = \beta_9 = 0$	Patton and Sheppard (2015)
HARRVJ	$\beta_{12} = \beta_{22} = \beta_{32} = \beta_7 = \beta_8 = \beta_9 = 0$	Andersen et al. (2007)
HARRSVJ	$\beta_7 = \beta_8 = \beta_9 = 0$	Chen and Ghysels (2011)
HARRVQ	$\beta_{12} = \beta_{22} = \beta_{32} = \beta_4 = \beta_5 = \beta_6 = 0$	Bollerslev et al. (2016)
HARRSVQ	$\beta_4 = \beta_5 = \beta_6 = 0$	New
HARRVJQ	$\beta_{12} = \beta_{22} = \beta_{32} = 0$	Shen et al. (2020)
HARRSVJQ	none	New

Table 2: 8 HARRV model specifications, presented as specific cases of the newly introduced HARRSVJQ model, by restricting the parameters of this model.

Restricting $\beta_{12} = \beta_{22} = \beta_{32} = 0$ excludes the leverage effect, restricting $\beta_4 = \beta_5 = \beta_6 = 0$ excludes jump components and restricting $\beta_7 = \beta_8 = \beta_9 = 0$ excludes quarticity components.

4.2 Neural networks

Neural networks are able to use and capture unknown information in data (Svozil et al.,1997). Neural networks consist of 'neurons' grouped in layers. Data enters the network in the input layer, the last layer is the output layer, and between those layers there is at least 1 hidden layer. Figure 2 displays a neural network that consists of 2 hidden layers, containing 4 neurons each.

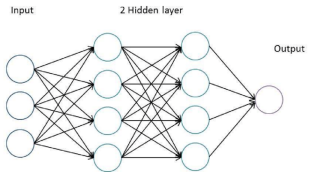


Figure 2: A multi-layered neural network

The output value of neuron i is determined by 4.14 and 4.15. These formulations are taken from Svozil et al. (1997). It holds that neuron i has output value x_i given by

$$x_i = f\left(\upsilon_i + \sum_{j \in \Gamma_i^{-1}} w_{ij} x_j\right), \tag{4.14}$$

with the transfer function f given by

$$f(x) = \frac{1}{1 + exp(-x)}. \tag{4.15}$$

Here, subset $\Gamma_i^{-1} \subseteq V$ is constructed to contain all predecessing neurons of neuron i. Each neuron in a layer is connected with all neurons in the next layer. The connection between neuron i and j is given by the weight coefficient w_{ij}. The i^{th} neuron is characterized by its threshold coefficient υ_i, which is also called the bias of neuron i. The higher the weight coefficient between two neurons, the more importance that connection has in the neural network.

By changing the biases and weight coefficients, the objective function E given by Svozil et al. (1997) is minimized. It is denoted as

$$E = \sum_o \frac{1}{2} (x_o - \hat{x}_o)^2, \tag{4.16}$$

where x_o and $\hat{x}_o$ are vectors with the actual and computed values respectively. The summation runs over all output neurons o.

I implement a two-layerend neural network, using the `neuralnet` package of R. The algorithm used to minimize the objective function in equation 4.16, is the rprop+ algorithm. This algorithm by Igel and Hüsken (2003) is a modification of the resilient backpropagation (RPROP) algorithm, suggested by Riedmiller and Braun (1993). Igel and Hüsken (2003) conclude that this modification gave such good results, that it should be the first choice when training neural networks. The key factor of this algorithm is that it uses different learning rates based on the sign of the partial derivative of the error function on the weights, $\frac{\delta E}{\delta w_{ij}}$, in consecutive steps. In the neural network I use, the step size is increased by a factor 1.2 if the sign stays the same after a step of the algorithm. If the sign changes, the step size is decreased with a factor 0.5.

Initializing the neural network at random, it is trained by the first two third of the HARRVJ variables. This means that the input layer consists of 6 neurons, namely RV_t, RV_{t-4}, RV_{t-9}, J_t, J_{t-4} and J_{t-9}. Following Pichl and Kaizoji (2017), I use a neural network with 2 hidden layers. Sheela and Deepa (2013) find that taking $\frac{4n^2+3}{n^2-8}$ as a rule for the optimal amount of neurons in a hidden layer, where n is the amount of neurons in the input layer, gives the best results. Given that $n = 6$ for my neural network, I choose the hidden layers to consist of 5 neurons. The stopping criteria is a treshold value for $\frac{\delta E}{\delta w_{ij}}$, which is set at 0.01.

Since the model stops running once the stop criterion is reached, it can stop at local minima rather than the global minimum of the error function. Since the model is initialized randomly, each training of the neural network might give different weights and biases. Therefore, I train the model in 30 repetitions, and select the model that has the lowest BIC value. The neural

network belonging to this repetition is used for testing the out-of-sample performance. This is done by forecasting the last third of the sample using this neural network.

4.3 Comparing the in-sample performance

The eight selected models in 4.1.2.4 have different numbers of variables. Including more variables to a model will increase its in-sample fit, but may also result in overfitting. This would decrease the out-of-sample performance of such a model. To examine which model has the best trade-off between model fit and number of variables, I use the Akaike information criterion (AIC), the Bayesian information criterion and the adjusted R^2:

$$\text{AIC} = 2k - 2\ln(\hat{L}) \; ; \tag{4.17}$$

$$\text{BIC} = k\ln(n) - 2\ln(\hat{L}) \; ; \tag{4.18}$$

$$\text{Adjusted } \text{R}^2 = 1 - \frac{(1-R^2)(n-1)}{n-k-1} \; . \tag{4.19}$$

Here, k denotes the number of estimated parameters, n denotes the total sample size and $\hat{L}$ denotes the maximum value of the likelihood function of the model. R^2 is a statistical measure that describes which proportion of the variance is explained by the variables of the regression. Models that fit the same sample can be compared. A lower AIC and BIC value and a higher adjusted R^2 means a better in-sample performance.

Wang and Liu (2006) mention that AIC tends to select a too conservative model with too many variables, such that the model is not parsimonious anymore. BIC gives a higher penalty to additional variables, and is more likely to select a parsimonious model.

4.4 Comparing the out-of-sample performance

This section discusses how the out-of-sample performance is measured and compared. The used loss functions are given in 4.4.1, and the tests for comparison are given in 4.4.2.

4.4.1 Used loss functions

Bollerslev et al. (1994) mentions that there is no obvious loss function for the evaluation of volatility models. Therefore, Hansen and Lunde (2005) suggests using six loss functions rather than one, namely MSE_1, MSE_2, $QLIKE$, R^2LOG, MAE_1 and MAE_2. The loss functions are given in the appendix, section 7.1, equation 7.1 till 7.6.

Hansen and Lunde (2005) notes that the MAE_1 and MAE_2 measures are more robust to outliers then for example the MSE_2 measure. Bollerslev et al. (1994) mentions the $QLIKE$ and R^2LOG measures to be more natural than the MSE^2 loss function.

4.4.2 Test statistics

To test whether a better predictive performance of a model lucky or caused by a difference in population, predictive accuracy tests are necessary (Diebold,2015). Therefore, I use the Diebold Mariano test, introduced by Diebold and Mariano (1995). It tests the null hypothesis of equal predictive performance, H_0: $E(d_t) = 0$, versus H_1: $E(d_t) \neq 0$, Here, d_t is the loss differential of model 1 and 2. The DM-statistic is defined as follows:

$$\text{DM} = \frac{\overline{d}}{\sqrt{\frac{\sigma^2}{T}}} \sim N(0,1), \tag{4.20}$$

where $\overline{d}$ is the average of vector d, σ^2 the variance of the loss differentials and T the number of elements in d. I use a modification of this test, proposed by Harvey et al. (1997). This modified DM-test does not rely on an assumption of forecast unbiasedness, and it rectifies the fact that the original DM-test can be quite oversized for sample of moderate size.

This test cannot be used when comparing nested models. When comparing a parsimonious null model with a larger model that nests the null model, the larger model will introduce noise since the population average of the extra estimated parameters will be zero. This is corrected for by the Clark and West adjustment to the DM-test (Clark and West,2007). This test is also used alongside the DM-test by Corsi and Renò (2012) and Shen et al. (2020), who also compare the predictive performance of HAR-models. The CW-test uses an adjusted mean square prediction for observation $t+1$ given by

$$\hat{f}_{t+1} = (\hat{e}_{1|t+1})^2 - (\hat{e}_{2|t+1})^2 + (\hat{y}_{1|t+1} - \hat{y}_{2|t+1})^2, \tag{4.21}$$

where $\hat{e}_{1|t+1}$ are the 1-day ahead forecast errors of the null model, $\hat{e}_{2|t+1}$ are the 1-day ahead forecast errors of the alternative model, and $\hat{y}_{1|t+1}$ and $\hat{y}_{2|t+1}$ are the null and alternative model forecast values respectively. The test statistic is given by

$$CW = \frac{\sqrt{N}\overline{f_1}}{\sqrt{Var(\hat{f}_{t+1} - \overline{f_1})}} \sim N(0,1), \tag{4.22}$$

where N is the number of forecasts, $\overline{f}_1$ stands for the average of the adjusted mean square prediction errors, and $\hat{f}_{t+1} - \overline{f_1}$ is a vector of all forecasts. Both tests are asymptotically standard normal under the null. Significant positive DM or CW statistics indicate that the alternative model has a better predictive performance, significant negative statistics indicate that the null model is preferred.

5 Results

This section gives the in and out-of-sample results of the linear HARRV models in 5.1, and the neural network forecasts in section 5.2.

5.1 Comparing realized volatility forecast models

Section 5.1.1 gives the in-sample results of the square root transformed HARRVJ model. Section 5.1.2 discusses the coefficient estimates of the 8 HARRV models mentioned in section 4.1.2. Section 5.1.3 compares the in-sample performance of the HARRV models, and section 5.1.4 discusses the out-of-sample results of the HARRV models.

5.1.1 The HARRVJ model

The results of the HARRVJ regression on the full dataset are presented in table 3.

Variable	β	**St. error**	**t-value**	**P-value**
Intercept	0.009*	0.0009	9.969	0.000
$\sqrt{RV_t}$	0.523*	0.052	9.979	0.000
$\sqrt{RV_{t-4}}$	0.137	0.075	1.821	0.069
$\sqrt{RV_{t-9}}$	0.201*	0.048	4.159	0.000
$\sqrt{J_t}$	-0.101	0.092	-1.103	0.270
$\sqrt{J_{t-4}}$	-0.072	0.195	-0.370	0.712
$\sqrt{J_{t-9}}$	-0.150	0.149	-1.006	0.315

Table 3: Full sample (1 January 2015 till 30 April 2022) HARRVJ regression results for Bitcoin price

*: Significant on a 5% level

As table 3 shows, the only variables with significant coefficients are the intercept, $\sqrt{RV_t}$ and $\sqrt{RV_{t-9}}$. This deviates from the findings of Pichl and Kaizoji (2017), who only have an insignificant coefficient for $\sqrt{J_t}$. The coefficient of $\sqrt{RV_t}$ is positive, just as in the results of Pichl and Kaizoji (2017). Their value for β_1 is 0.345 however, which is lower than the reported value of 0.523. This means that the realized variance at day t had a lower impact on the realized variance of day $t+1$ during their period of 2013/2/8 till 2017/4/7. The other significant value, for $\sqrt{RV_{t-9}}$, even has a different sign. Pichl and Kaizoji (2017) report a value of -0.227. This means that they found that a longer period of higher realized volatility tends to be followed by a lower daily realized volatility and vice versa. My findings however show a positive effect of a higher realized variance in the past 10 days on the realized variance of tomorrow. The most likely explanation for the differences is that Pichl and Kaizoji (2017) research a different period.

A possible explanation for the insignificant jumps, might be multicollinearity. Pichl and Kaizoji (2017) report negative coefficients for $\sqrt{J_t}$ and $\sqrt{J_{t-4}}$, and a positive coefficient for $\sqrt{J_{t-9}}$. The three jump components in my regression are all negative however, increasing the risk of multicollinearity and thereby insignificant parameters.

5.1.2 Comparing the HARRV-model coefficients

The coefficients of the 8 models presented in section 4.1.2.4 are given in table 4.

Model	β_0	β_{11}	β_{12}	β_{21}	β_{22}	β_{31}	β_{32}	β_4	β_5	β_6	β_7	β_8	β_9
HARRV	0.008*	0.498*		0.126*		0.165*							
HARRSV	0.008*	0.381*	-0.162*	0.159*	-0.068	0.240*	0.222*						
HARRVJ	0.008*	0.517*		0.144		0.200*		-0.095	-0.081	-0.147			
HARRSVJ	0.009*	0.406*	-0.163*	0.161*	-0.067	0.280*	0.201*	-0.118	-0.030	-0.186			
HARRVQ	0.001	0.590*		0.263*		0.108					-0.703*	-1.204*	-0.195
HARRSVQ	0.002	0.459*	-0.153*	0.306*	-0.087	0.163*	0.219*				-0.596*	-1.353*	0.104
HARRVJQ	0.005*	0.519*		0.197*		0.115*		0.880	3.326	-0.979	-0.449	-1.547	-0.366
HARRSVJQ	0.005*	0.404*	-0.161*	0.221*	-0.107*	0.192*	0.246*	-0.002	5.305	-2.090	-0.353	-1.972	-0.027

Table 4: Full sample (1 January 2015 till 30 April 2022) regression results for 8 HARRV models
*: Significant on a 5% level

The coefficients for $\sqrt{RV_t}$, $\sqrt{RV_{t-4}}$ and $\sqrt{RV_{t-9}}$ are all positive and significant, except for β_{21} in the HARRVJ model and β_{31} in the HARRVQ model. This shows that higher realized variance over both shorter and longer time-spans have a positive effect on the realized variance of tomorrow.

The negative realized semivariance components of the past day, are all significantly negative. This is interesting, since that means that volatility due to negative returns the day before lead to less volatility of Bitcoin prices the next day than volatility due to positive returns. The effect of the negative realized semivariance of the past 5 days is still negative, although only the coefficient of the HARRSVJQ model is significant. The average negative realized semivariances component of the past 10 days however is significantly positive for all models that include the variable.

No model has a significant jump component. The only significant realized quarticity components are reported in models that do not include jump components. All of these significant coefficients, β_7 and β_8 for HARRSVQ and HARRVJQ are negative.

5.1.3 Comparing the in-sample performance of the HARRV-models

The AIC, BIC and adjusted R^2 values, as discussed in section 4.3, are given in table 5. The HARRV-models are fitted over the entire period of 1 January 2015 till 30 April 2022. The HARRSVJQ model has the lowest AIC and adjusted R^2 value, and the HARRSVQ has the

lowest BIC value. It therefore seems that the HARRSVJQ model has the best in-sample fit. However, as mentioned in 4.3, a model selected with AIC often includes too many variables, whereas using BIC is more likely to select a parsimonious model.

It is noticeable that adding the realized negative semivariance improves the in-sample fit of all 4 models that do not have the component included. Adding realized quarticity also improves all 4 models that do not have this component included. Adding jumps results in better AIC and adjusted R^2 values for all the 4 models that do not have these components included, but it increases the BIC values. Compare for example the BIC value of the HARRV model, which is -13790.57, to the HARRVJ BIC value. The BIC value has increased to -13784.88.

Model	**AIC**	**BIC**	**Adjusted R^2**
HARRV	-13820.04	-13790.57	0.492
HARRSV	-13917.61	-13870.46	0.511
HARRVJ	-13832.03	-13784.88	0.495
HARRSVJ	-13932.01	-13867.18	0.514
HARRVQ	-13861.71	-13814.55	0.501
HARRSVQ	-13953.05	**-13888.22**	0.518
HARRVJQ	-13862.66	-13797.82	0.501
HARRSVJQ	**-13967.46**	-13884.94	**0.521**

Table 5: AIC, BIC and adjusted R^2 for the HARRV models from 1 January 2015 till 30 April 2022

5.1.4 Comparing the prediction performance of the HARRV-models

Given that Bitcoin was still relatively new in 2015, and is now known worldwide, it is likely that its volatility behavior has changed over the years. Therefore, the forecasts are constructed with a small moving window, namely a moving window of 365 days. This generates forecasts and loss function values for 1 January 2016 till 30 March 2022. The models are refitted everyday.

The loss function values, discussed in section 4.4.1, are given in table 6. The HARRSVJ model has the lowest MSE_1, MSE_2 and $QLIKE$ value, the HARRSV model has the lowest R^2LOG, MAE_1 and MAE_2 value. It is noticeable that both models include the realized semivariance. As mentioned in section 4.4.1, the MSE measures are less robust to outliers than the MAE measures. Since the HARRSVJ model has the lowest loss function value for both MSE measures, the HARRSVJ model seems to handle outliers better. Since both the HARRSV and HARRSVJ model have the lowest value for three loss functions, and the second lowest for the other three loss functions, these models seem to have the best predictive performance.

The results of the CW statistics for nested models and the DB statistics for non-nested

Model	MSE_1	MSE_2	$QLike$	R^2LOG	MAE_1	MAE_2
HARRV	$4,168 * 10^{-4}$	$1.530 * 10^{-5}$	-5.331	0.626	$1.229 * 10^{-2}$	$1.333 * 10^{-3}$
HARRSV	$3.472 * 10^{-4}$	$1.379 * 10^{-5}$	-5.418	**0.503**	$\mathbf{1.068 * 10^{-2}}$	$\mathbf{1.159 * 10^{-3}}$
HARRVJ	$4.116 * 10^{-4}$	$1.508 * 10^{-5}$	-5.345	0.626	$1.234 * 10^{-2}$	$1.336 * 10^{-3}$
HARRSVJ	$\mathbf{3.461 * 10^{-1}}$	$\mathbf{1.379 * 10^{-5}}$	**-5.424**	0.508	$1.080 * 10^{-2}$	$1.168 * 10^{-3}$
HARRVQ	$4.143 * 10^{-4}$	$1.514 * 10^{-5}$	-5.256	0.639	$1.232 * 10^{-2}$	$1.338 * 10^{-3}$
HARRSVQ	$4.207 * 10^{-4}$	$1.584 * 10^{-5}$	-5.250	0.652	$1.229 * 10^{-2}$	$1.336 * 10^{-3}$
HARRVJQ	$3.681 * 10^{-4}$	$1.414 * 10^{-5}$	-4.880	0.578	$1.094 * 10^{-2}$	$1.186 * 10^{-3}$
HARRSVJQ	$3.803 * 10^{-4}$	$1.425 * 10^{-5}$	-5.025	0.595	$1.101 * 10^{-2}$	$1.189 * 10^{-3}$

Table 6: The loss function values for the forecasts of the volatility of Bitcoin for 1 January 2016 till 30 April 2022, using 8 HARRV models

models, as discussed in section 4.4.2, are given in table 7. It shows that the HARRV model is significantly outperformed by the HARRSV, HARRSVJ, HARRVJQ and HARRSVJQ model. This indicates that the HARRV model might be underfitted. The HARRSV model significantly outperforms the HARRV, HARRVJ and HARRVQ model, indicating that it has a good predictive performance.

	HARRV	**HARRSV**	**HARRVJ**	**HARRSVJ**	**HARRVQ**	**HARRSVQ**	**HARRVJQ**	**HARRSVJQ**
HARRV		*4.430**	*1.662*	*4.440**	*1.358*	*1.013*	*3.844**	*3.877**
HARRSV			-2.752*	*1.416*	-2.694*	*1.567*	-1.302	*0.450*
HARRVJ				*5.238**	-0.178	-0.700	*4.833**	*4.752**
HARRSVJ					-2.730*	-1.670	-1.420	*-0.075*
HARRVQ						*-0.382*	*3.338**	*3.242**
HARRSVQ							1.394	*1.757*
HARRVJQ								*0.803*
HARRSVJQ								

Table 7: This table presents CW-statistics (in italics) for nested models, and DB-statistics for non-nested models, for the forecasting horizon of 1 day during 1 January 2016 till 30 April 2022. Predictions are constructed using a moving window of 1 year. Positive results in a cell denote that the model in the first column performs better than the model in the first row, and vice versa. The standard errors are computed using Newey and West (1987).
*: Significant on a 5% level

The HARRVJ model is significantly outperformed by 4 other models, one of which is the HARRSVJ model. This HARRSVJ model outperforms 3 models, all of which do not include the realized semivariance. This shows that adding realized semivariance to the model improves the predictive performance. This is also observed for the HARRVQ model, which is significantly outperformed by the HARRSV, HARRSVJ, HARRVJQ and HARRSVJQ model. Three of these

models include realized semivariance. Out of the models that do not include realized semivariance, the HARRVJQ performs the best, since it significantly outperforms HARRV, HARRVJ and HARRVQ. These three models are also outperformed by the HARRSVJQ model.

The results show that the HARRVJQ and HARRSVJQ model perform well, but might be slightly overfitted. The smaller models that include realized semivariance, HARRSV and HARRSVJ, significantly outperform some competitors and seem to have the best predictive performance.

5.2 Neural network forecasts

As mentioned in section 4.2, the neural network is trained with the first two-thirds of the data. The training observations range from 12 January 2015 till 23 November 2019, and the forecasts are constructed from 24 November 2019 till 30 April 2022. The training is done with 30 repetitions, their in-sample measures are given in appendix 7.2. The weights of the repetition with the best BIC value, which is repetition 28, are given in appendix 7.3. To properly compare the neural network with the HARRVJ model, the HARRVJ forecasts are constructed in the same way. This is done by using the coefficients from regressing the model over the observations from 1 January 2015 till 23 November 2019 to forecast the realized volatilities of 24 November till 30 April 2022.

The variance forecasts are presented in figure 3. It can be seen that both the neural network and the HARRVJ model predict the actual variance quite good. However, certain outliers like the high peaks in March 2020, January and May 2021 are poorly forecast by both models. 13 March 2020 for example, the actual variance is 0.314, whereas the neural network predicts 0.132 and the HARRVJ model predicts 0.136.

The loss function values of the realized volatility forecasts are presented in Table 8. The neural network has lower loss values for the MSE_1, MSE_2, R^2LOG, MAE_1 and $MASE_2$ measures. The only value in favor of the HARRVJ model are the $QLIKE$ loss function values. The results therefore seem to indicate that the neural network has a better predictive performance.

Model	$\boldsymbol{MSE_1}$	$\boldsymbol{MSE_2}$	$\boldsymbol{QLIKE}$	$\boldsymbol{R^2LOG}$	$\boldsymbol{MAE_1}$	$\boldsymbol{MAE_2}$
Neural network	$\mathbf{3.005 * 10^{-4}}$	$\mathbf{1.553 * 10^{-5}}$	-5.528	**0.426**	$\mathbf{1.008 * 10^{-2}}$	$\mathbf{1.093 * 10^{-3}}$
HARRVJ	$3.136 * 10^{-4}$	$1.744 * 10^{-5}$	**-5.531**	0.429	$1.021 * 10^{-2}$	$1.123 * 10^{-3}$

Table 8: Loss function values of the realized volatility forecasts for the neural network and the HARRVJ model, both trained on data from 01/01/2015 till 23/11/2019, forecasting 24/11/2019 till 30/04/2022 with a 1 day forecasting horizon.

To test whether the neural network signficantly outperforms the HARRVJ model, a Diebold

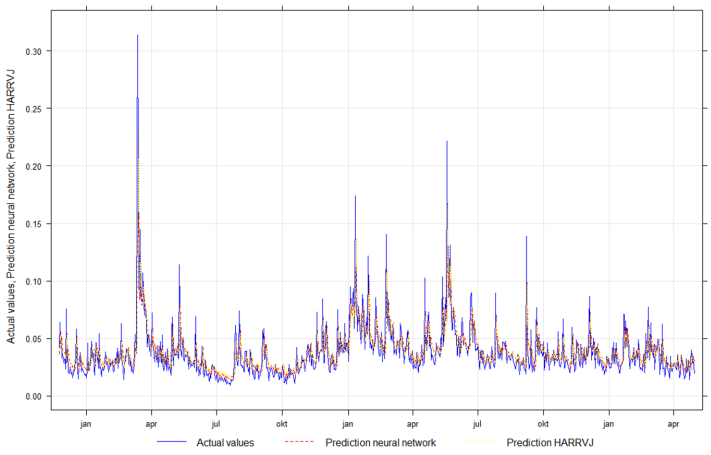


Figure 3: Actual values of the Bitcoin variance, forecasts from the HARRVJ model and neural network, from 24 November 2019 till 30 April 2022

Mariano test is performed, as discussed in section 4.4.2. Taking the neural network as the null model, and HARRVJ as the alternative model, the test statistic is -0.995. This indicates that the neural network has a better predictive performance. However, since it is not a significant value, no conclusions can be drawn.

6 Discussion and conclusion

In sum, I see that the models that exclude realized quarticity but include the negative semivariance, namely HARRSV and HARRSVJ, have a significantly better predictive performance than the HARRVJ model. The HARRSV and HARRSVJ model also show a better in-sample fit. This indicates that the HARRVJ model overlooks the different impact positive and negative Bitcoin have on its volatility. Feeding a neural network with the variables of the HARRVJ model gives lower loss function values than HARRVJ for 5 of the 6 loss functions, but a Diebold Mariano test can not conclude that the performance of the neural network was significantly better than that of the linear HARRVJ model.

In general, the addition of realized quarticity, jump components and a leverage effect through realized negative semivariance, increases the in-sample fit of the linear HARRV-models. None of the jump components in one of the eight models is significant however, and quarticity coefficients are only significant in models that do not include jump components. The HARRSVQ and HARRSVJQ have the best in-sample fit. These models seem overfitted however, since they have higher loss function values than the HARRSV and HARRSVJ models that do not include the realized quarticity. From the models that do not include the realized semivariance, the HARRVJQ model that includes jumps and realized quarticity has the best out-of-sample performance.

As already mentioned, the HARRVJ model overlooks the leverage effect, and is therefore outperformed by the HARRSV and HARRSVJ model. It is noticeable that the in-sample results show that for Bitcoin, a leverage effect means that negative returns have less impact on the volatility than positive returns. Volatility due to negative returns the day before lead to less volatility of Bitcoin prices the next day than volatility due to positive returns. This may be due to the central role of public attention for cryptocurrencies, which is often reported (for example by Kristoufek,2013). Positive returns might generate more public attention and thus increase the Bitcoin trades, which generates more volatility. The average negative realized semivariance of the past 10 days had a positive effect on the volatility of the next day however. Longer periods of negative returns might lead to panic amongst Bitcoin holders, generating higher volatility.

It is also interesting to see that the HARRSVJ model has lower loss function values for the two MSE measures, whereas the HARRSV model has lower loss function values for the MAE measures. Hansen and Lunde (2005) mention that the MAE are more robust to outliers than MSE_2. Given that the HARRSVJ model has the lowest loss function value for a measure that is not robust to outliers, it seems that the HARRSVJ model handles outliers better than the HARRSV model. Modelling discontinuities by jump components thus seems to be beneficial when it comes to handling outliers. I conclude that a HARRSVJ model is an improvement of the HARRVJ model when predicting Bitcoin volatility, as it accounts for the leverage effect of the returns.

In further research, the effect of the square root transfortmation can be investigated by comparing it with an untransformed HARRVJ model and a logarithmically transformed HARRVJ model. The 8 linear HARRV models that are considered all use a square root transformation. Andersen et al. (2007) mention that transforming realized volatilities logarithmically resulted in an approximately normal distribution, which might motivate investigating this transformation.

The results of the neural network look promising, but no significant improvement in predictive performance has been found. Maybe more advanced machine learning (ML) methods would show better out-of-sample results. Seo and Kim (2020) for example compares a neural network with a higher order neural network (HONN) when forecasting the volatility of Bitcoin. These HONNs generally have a better predictive performance than the neural networks. Further research could assess whether more advanced ML applications fed by the variables of HARRV-models generate better volatility forecasts.

References

Andersen, T. G. and Bollerslev, T. (1998). Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International Economic Review*, pages 885–905.

Andersen, T. G., Bollerslev, T., and Diebold, F. X. (2007). Roughing it up: Including jump components in the measurement, modeling, and forecasting of return volatility. *The Review of Economics and Statistics*, 89(4):701–720.

Andersen, T. G., Bollerslev, T., Diebold, F. X., and Labys, P. (2000). Exchange rate returns standardized by realized volatility are (nearly) Gaussian. *Econometric Reviews*, 27(1-3):10–45.

Barndorff-Nielsen, O. E., Kinnebrock, S., and Shephard, N. (2008). Measuring downside risk-realised semivariance. *CREATES Research Paper*, (2008-42).

Barndorff-Nielsen, O. E. and Shephard, N. (2004). Power and bipower variation with stochastic volatility and jumps. *Journal of Financial Econometrics*, 2(1):1–37.

Bergsli, L. Ø., Lind, A. F., Molnár, P., and Polasik, M. (2022). Forecasting volatility of Bitcoin. *Research in International Business and Finance*, 59:101540.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327.

Bollerslev, T., Engle, R. F., and Nelson, D. B. (1994). Arch models. *Handbook of Econometrics*, 4:2959–3038.

Bollerslev, T., Patton, A. J., and Quaedvlieg, R. (2016). Exploiting the errors: A simple approach for improved volatility forecasting. *Journal of Econometrics*, 192(1):1–18.

Boudt, K., Kleen, O., and Sjørup, E. (2021). Analyzing intraday financial data in R: The highfrequency package. *Available at SSRN 3917548*.

Carnero, M. A., Peña, D., and Ruiz, E. (2004). Persistence and kurtosis in GARCH and stochastic volatility models. *Journal of Financial Econometrics*, 2(2):319–342.

Chen, X. and Ghysels, E. (2011). News—good or bad—and its impact on volatility predictions over multiple horizons. *The Review of Financial Studies*, 24(1):46–81.

Clark, T. E. and West, K. D. (2007). Approximately normal tests for equal predictive accuracy in nested models. *Journal of Econometrics*, 138(1):291–311.

Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196.

Corsi, F. and Renò, R. (2012). Discrete-time volatility forecasting with persistent leverage effect and the link with continuous-time volatility modeling. *Journal of Business & Economic Statistics*, 30(3):368–380.

Diebold, F. X. (2015). Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of Diebold–Mariano tests. *Journal of Business & Economic Statistics*, 33(1):1–1.

Diebold, F. X. and Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic statistics*, 20(1):134–144.

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

Hansen, P. R. and Lunde, A. (2005). A forecast comparison of volatility models: does anything beat a GARCH (1, 1)? *Journal of Applied Econometrics*, 20(7):873–889.

Harvey, D., Leybourne, S., and Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of forecasting*, 13(2):281–291.

Huang, X. and Tauchen, G. (2005). The relative contribution of jumps to total price variance. *Journal of Financial Econometrics*, 3(4):456–499.

Igel, C. and Hüsken, M. (2003). Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50:105–123.

Kristjanpoller, W., Fadic, A., and Minutolo, M. C. (2014). Volatility forecast using hybrid neural network models. *Expert Systems with Applications*, 41(5):2437–2442.

Kristoufek, L. (2013). Bitcoin meets Google Trends and Wikipedia: Quantifying the relationship between phenomena of the internet era. *Scientific Reports*, 3(1):1–7.

McAleer, M. and Medeiros, M. C. (2008). Realized volatility: A review. *Econometric Reviews*, 27(1-3):10–45.

McNally, S., Roche, J., and Caton, S. (2018). Predicting the price of Bitcoin using machine learning. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 339–343. IEEE.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260.

Newey, W. K. and West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelationconsistent covariance matrix. *Econometrica*, pages 703–889.

Patton, A. J. and Sheppard, K. (2015). Good volatility, bad volatility: Signed jumps and the persistence of volatility. *Review of Economics and Statistics*, 97(3):683–697.

Pichl, L. and Kaizoji, T. (2017). Volatility analysis of Bitcoin. *Quantitative Finance and Economics*, 1(4):474–485.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591. IEEE.

Seo, M. and Kim, G. (2020). Hybrid forecasting models based on the neural networks for the volatility of Bitcoin. *Applied Sciences*, 10(14):4768.

Sheela, K. G. and Deepa, S. N. (2013). Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering*, 2013.

Shen, D., Urquhart, A., and Wang, P. (2020). Forecasting the volatility of Bitcoin: The importance of jumps and structural breaks. *European Financial Management*, 26(5):1294–1323.

Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62.

Wang, Y. and Liu, Q. (2006). Comparison of Akaike information criterion (AIC) and Bayesian information criterion (BIC) in selection of stock–recruitment relationships. *Fisheries Research*, 77(2):220–225.

White, H. (2006). Approximate nonlinear forecasting methods. *Handbook of Economic Forecasting*, 1:459–512.

Wong, B. K. and Selvi, Y. (1998). Neural network applications in finance: A review and analysis of literature (1990–1996). *Information & Management*, 34(3):129–139.

7 Appendix

7.1 Loss functions

$$MSE_1 = n^{-1} \sum_{t=1}^{n} \left(\sqrt{RV_t} - \sqrt{\hat{RV}_t} \right)^2, \tag{7.1}$$

$$MSE_2 = n^{-1} \sum_{t=1}^{n} \left(RV_t - \hat{RV}_t \right)^2, \tag{7.2}$$

$$QLIKE = n^{-1} \sum_{t=1}^{n} \left(\log(\hat{RV}_t) + \frac{RV_t}{\hat{RV}_t} \right), \tag{7.3}$$

$$R^2LOG = n^{-1} \sum_{t=1}^{n} \left[log \left(\frac{RV_t}{\hat{RV}_t} \right) \right]^2, \tag{7.4}$$

$$MAE_1 = n^{-1} \sum_{t=1}^{n} \left| \sqrt{RV_t} - \sqrt{\hat{RV}_t} \right|, \tag{7.5}$$

$$MAE_2 = n^{-1} \sum_{t=1}^{n} \left| RV_t - \hat{RV}_t \right|. \tag{7.6}$$

Here, RV_t stands for the actual realized volatility at day t, $\hat{RV}_t$ stands for the forecasted value of the realized variance, and n is the total number of days for which a volatility forecast is constructed. These formulations are taken from Hansen and Lunde (2005).

7.2 Neural network selection

Iteration	AIC	BIC
1	142.611	531.922
2	142.611	531.922
3	142.611	531.921
4	142.609	531.920
5	142.613	531.924
6	142.609	531.919
7	142.611	531.922
8	142.612	531.922
9	142.608	531.918
10	142.616	531.926
11	142.611	531.921
12	142.612	531.922
13	142.610	531.920
14	142.612	531.922
15	142.609	531.920
16	142.611	531.921
17	142.610	531.920
18	142.617	531.927
19	142.617	531.928
20	142.612	531.923
21	142.611	531.921
22	142.605	531.915
23	142.613	531.923
24	142.611	531.921
25	142.607	531.917
26	142.617	531.927
27	142.616	531.927
28	**142.604**	**531.914**
29	142.609	531.919
30	142.611	531.921

Table 9: AIC and BIC values of 30 iterations of the neural network

7.3 Weights of the selected neural network

	Neuron 1 layer 1	**Neuron 2 layer 1**	**Neuron 3 layer 1**	**Neuron 4 layer 1**	**Neuron 5 layer 1**
Intercept	0.521	-1.836	0.194	-1.700	0.270
RV1	-0.735	-2.723	4.643	3.365	6.524
RV5	-7.986	-0.479	6.630	0.113	0.306
RV10	-4.159	-0.297	1.301	1.666	0.962
J1	5.888	2.185	-1.653	0.001	-4.908
J5	-2.917	7.189	-1.059	1.436	1.424
J10	-1.186	5.863	0.965	0.842	0.535

Table 10: Weights between the input layer and the first hidden layer of the neural network of repetion 28

	Neuron 1 layer 2	**Neuron 2 layer 2**	**Neuron 3 layer 2**	**Neuron 4 layer 2**	**Neuron 5 layer 2**
Intercept	-2.451	-0.393	0.505	0.845	0.258
Neuron 1 layer 1	-2.807	0.103	0.694	1.915	1.617
Neuron 2 layer 1	-0.810	-0.575	-1.223	-0.058	-0.910
Neuron 3 layer 1	-0.368	1.590	1.401	-2.334	0.411
Neuron 4 layer 1	2.178	0.241	-1.633	-0.787	0.849
Neuron 5 layer 1	0.138	-1.222	-0.335	0.194	0.935

Table 11: Weights between the first hidden layer and the second hidden layer of the neural network of iteration 28

	RV prediction
Intercept	-0.793
Neuron 1 layer 2	-0.443
Neuron 2 layer 2	-0.301
Neuron 3 layer 2	-0.062
Neuron 4 layer 2	-0.423
Neuron 5 layer 2	1.416

Table 12: Weights between the second hidden layer and the output layer of the neural network of iteration 28

7.4 Code files

One programming file is used, which is named 'Thesis_Bitcoin.R'. This file combines all data analysis done to receive the result mentioned in this thesis. It contains 596 lines of code.

Lines 1-45 are used to install certain packages and write them into the library of R. The most important packages used are the `highfrequency` and `neuralnet` package, which are used for constructing HARRV models and neural networks respectively. Lines 47-56 write Excel files containing Bitcoin price data, downloaded from `https://www.cryptodatadownload.com/data/bitstamp`. Lines 57-79 create a function that deletes every n^{th} element, such that 5 minute frequency price data and daily price data are created.

The logarithmic returns of this price data are calculated in lines 80-101, which also plots a graph of the daily data and calculates some descriptive statistics. Lines 102-142 calculates realized volatility, realized bipower variance and jump statistics for each day in the sample. Lines 143-278 calculate the coefficients of 8 HAR-models over the full sample. These models are HARRV, HARRSV, HARRVJ, HARRSVJ, HARRVQ, HARRSVQ, HARRVJQ, and HARRSVJQ. Lines 279-418 calculates forecasts with these 8 models, using a moving window.

The loss functions of these forecasts are calculated in lines 419-444. DM-statistics and CW-statistics are calculated in lines 445-465. Forecasts can be plotted with the true values using lines 466-475.

Line 476-558 reruns the HARRVJ model and runs a neural network over the first two thirds of the data. These predictions are plotted with the actual values in lines 559-567. Loss functions of both predictions are calculated in lines 568-593. DM-statistics and CW-statistics for these predictions are calculated in lines 593-596.