Realized Volatility Forecasting Using HAR-Style Neural Networks

Forecasting the Realized Volatility of Cryptocurrencies

Sebastian Tved Linde
Study No.: 20156272
Aalborg University Business School - M.Sc Finance
Supervisor: Frederik Steen Lundtofte, Full Professor of Finance

M.Sc Finance - Master's Thesis
28-06-2022

This project was created using Google Colab.
The notebook can be found at Google Colab.
`https://colab.research.google.com/drive/1sJzRECcHC5H5JRsX7B5iEZdrGE-onn3X?usp=sharing`
The computations done in Colab was done on a GPU, with a Colab Pro License. Further, the code used for the analysis can be found at:
`https://github.com/stlinde/Thesis/tree/main/analysis`.

0.1 Abstract

The volatility of financial assets is one of the most well researched and important variables in finance. Being used in everything from portfolio optimization, to option pricing and further risk management, volatility is used throughout all of finance. Volatility is somewhat predictable, but we are not able to observe the level of volatility of a return process. Thus, volatility must be estimated. The most common ways of doing this is using either historical returns or implied volatilities from option pricing. With the increased availability of high-frequency data a new and perhaps better way of estimating volatility has been developed, the realized volatility. The goal of this thesis is to forecast this realized volatility on four cryptocurrencies, using econometric HAR models and deep learning models. Two deep learning models are developed, one simple feed-forward neural network and another neural network utilizing dilated causal convolutions, inspired by Google Deep Mind's WaveNet architecture. The main objective is to compare the three models to understand the forecasting capabilities of deep learning in a realized volatility setting.

The results show that the econometric HAR models perform best on the mean squared error criterion and the simple feed-forward neural network performs best on the QLIKE criterion. This may be indicative of the HAR models being better able to model the extreme levels of volatilities observed and the day-to-day fluctuations. On the other hand, the deep learning models seem better at modeling the inherent base volatility and general movements of the volatility over longer time periods.

The results may be used as a guidance in model selection or the models may be used in an ensemble fashion, creating more balanced forecasts. Further, the results call for further research analyzing the differences in forecasting capabilities of the two model types.

Contents

Chapter 1 Introduction

1.1 The Electronic Currency Bitcoin

Since the mythical author Satoshi Nakamoto published the whitepaper on Bitcoin *"Bitcoin: A Peer-to-Peer Electronic Cash System"* in 2008, blockchain technologies and cryptocurrencies have become everyday names and subject of much research. The original intend for Bitcoin was to be *"A pure peer-to-peer version of electronic cash"* (Nakamoto, 2008, p. 1). The goal of this electronic cash is to allow *"...online payments to be sent directly from one party to another without going to a financial institution"* (Nakamoto, 2008, p. 1). Bitcoin, was thus invented as an alternative to cash and other currencies, as a digital and secure medium to make transactions with. Following Bitcoin a whole host of other cryptocurrencies have been developed, with similar characteristics to Bitcoin.

To be viewed as a currency, a cryptocurrency must satisfy the following characteristics. Bitcoin must be a medium of exchange, which by the fact that it enables peer-to-peer transactions, may be deemed satisfied. It must function as a unit of account, thus, one must be able to use bitcoin as a measure of relative worth of economic transactions (Mankiw, 2009, p. 80-81). Lastly, it must function as a store of value, and therefore, must be able to transfer the purchasing power from the present to the future (Mankiw, 2009, p. 80-81). For Bitcoin the first of the characteristics, being a medium of exchange, seems to be satisfied, particularly because more businesses have begun accepting Bitcoin as a payment (Yermack, 2015, p. 36-38). To function as a store of value, a currency must be able to be securely stored and maintain a relatively stable value in the periods between use. Ordinarily, the ability of a currency of being securely stored is maintained by a bank. Storing cryptocurrencies is done through digital wallets, and the ability of being securely stored is facilitated by insuring the digital wallet, often done by the digital wallet companies. The major concern for cryptocurrencies in terms of being considered a viable currency is its fluctuations in value. The characteristics of being a store of value and a unit of value, is reliant on a somewhat stable value of the currency. Bitcoin shows extreme volatility compared with other currencies, and its market value can vary greatly within a short period of time (Yermack, 2015, p. 38-43). This challenges Bitcoin, and other cryptocurrencies, as a currency, but enables Bitcoin to be seen and used as a speculative investment. It is this extreme volatility that makes cryptocurrencies an interesting set of asset to consider in this thesis.

1.2 Volatility in the Financial Markets

Volatility can, in layman terms, be described as the size of the fluctuations or dispersion of an asset over a given period of time. It is one of the central measures of interest in finance. Volatility is used extensively in risk management, asset allocation, derivatives pricing, hedging and many more financial subjects (Engle & Patton, 2000, p. 237). One of the primary goals of volatility modeling is to predict the future volatility. Being able to accurately predict future volatility, one may better price various financial assets, select portfolios, and manage risks. Volatility is therefore a subject of great interest for financial applications.

Being a subject of much research and application, volatility is well researched. It is, however, not possible to directly observe volatility (Tsay, 2005, p. 98). Thus, volatility must be estimated by the practitioner or researcher, before being modeled or used. The most common ways to estimate volatility is by using the standard deviation of historic returns or estimating the implied volatilty as given by options pricing theory (Tsay, 2005, p. 98) (Hull, 2015, p. 325). A third approach, that has been increasing in its use, for estimating volatility is by using the realized volatility Andersen et al. (2001). Estimating realized volatility however, requires access to high-frequency intraday data, as it is computed as the square root of the sum of squared returns Andersen et al. (2001). Realized volatility is thus the quadratic variation of the intraday returns of an asset (Tsay, 2005, p. 141). Furthermore, estimating volatility by using realized volatility imposes another challenge in ordinary assets. As trading does not happen continuously, all year around, the volatility of an asset may be split into intraday volatility and overnight volatility (Tsay, 2005, p. 141). The choice of cryptocurrencies as the assets of interest in this thesis, is further motivated by the fact that cryptocurrencies trade continuously all year round and thus, there are no overnight volatility. Using realized volatility as a non-parametric estimate of volatility have been empirically shown to provide better forecasts than those of GARCH-type and stochastic volatility models (Andersen et al., 2003, p. 618-621). Thus, studying the realized volatility in a highly volatile asset is deemed of great relevance to the financial subject.

In modeling and forecasting realized volatility the de facto standard models are based on the Heterogeneous Autoregressive (HAR) model introduced by Corsi (2008). The HAR-RV model proposed by Corsi models the realized volatility as a linear function of the lagged realized volatility using the same aggregation period, in most cases a day. Furthermore, the model includes weekly and monthly aggregated realized volatilities (Corsi, 2008, p. 176-178). Later studies in the modeling of realized volatility have mainly been grounded in the HAR-RV model, while attempting to incorporate non-linearities, leverage effects, jumps etc. (Qiu et al., 2019, p. 56). The HAR-RV model and its variants have shown great forecasting performance, but the modeling of highly non-linear relationships is challenged by the fact that they are often modeled using linear regression. With the advance of machine learning techniques, and especially deep learning techniques, the modeling accuracy of non-linear relationships is often increased by using such techniques (Christensen et al., 2021, p. 1-2)

1.3 Machine Learning

This thesis will focus on the application of neural networks or deep learning to the subject of forecasting realized volatility. Deep learning as a subject and technique have existed for many decades. First being introduced in the 1940s as a method of computationally representing the learning that happens in the brain, therefore, the name neural networks (Goodfellow et al., 2016, p. 13-19). The use of deep learning techniques were minor until 2006 when deep learning displayed remarkable results in the recognition of handwritten digits. Since then, the use of deep learning have exploded and is now powering many of todays technologies (LeCun et al., 2015, p. 436-439). Deep learning as a method is very flexible and offers several ways of dealing with highly non-linear relationships. It is this ability to handle non-linear relationships that motivates the study of deep learning as applied to forecasting realized volatility.

1.4 Research Question

How can neural networks enhance the capability of HAR-style models in forecasting the realized volatility of cryptocurrencies?

1.5 Thesis Structure

The thesis will be structured as follows. Chapter 2 will contain an examination of the theory underlying volatility forecasting, cryptocurrency trading and deep learning. The time series used will be presented in chapter 3 with an examination of their structure and characteristics. Chapter 4 will introduce the models used in the project, the training and testing methodology, as well as the evaluation criterions used. Chapter 5 will present the results of the analysis, with a comparison between the econometric models as baselines and the neural network. The discussion, chapter 6, will be revolving around the applicability of neural networks for volatility forecasting and especially the ability of making inference on neural networks. Chapter 7 will conclude the project

Chapter 2 Theory

This chapter will lay the foundation for the forecasting study to be done in this thesis. 2.1 introduces the subject of volatility, describes the stylistic facts of it and how it relates to market microstructure. 2.2 briefly examines Bitcoin and its characteristics as a speculative investment. Lastly, 2.3 gives a summary of the history of neural networks and the developments over the past years, the internal techniques driving the learning in neural networks, and how neural networks are used in time series forecasting.

2.1 The Volatility of Financial Assets

The return of an asset can be computed over a chosen period of time given the prices at each time step. If we denote the price of an asset at time t as P_t we can compute the return of the asset between $t-1$ and t as:

$$R_t = \frac{P_t}{P_{t-1}} - 1 \tag{2.1}$$

This return is often referred to as the simple return (Campbell et al., 2012, p. 9-10). In econometric settings we are most often interested in the continuously compounded return, also referred to as the log return, given as:

$$r_t \equiv \log(1 + R_t) = \log \frac{P_t}{P_{t-1}} = p_t - p_{t-1} \tag{2.2}$$

Here p_t and p_{t-1} denotes the log-transformed prices at time t and $t-1$, respectively (Campbell et al., 2012, p. 11). It is the volatility of these log returns that are of interest in the current work.

One of the central theories in finance is the Efficient Market Hypothesis. The Efficient Market Hypothesis exist in several forms, weak, semi-strong, and strong, but can generally be summarized by the following quote:

> A market is efficient with respect to information set θ_t if it is impossible to make economic profits by trading on the basis of information set θ_t.
> *Some Anomalous Evidence Regarding Market Efficiency* (Jensen, 1978, p. 96).

The three variants of the Efficient Market Hypothesis differs in what is included in the information set θ_t. In effect the Efficient Market Hypothesis, if valid, states that given the information set θ_t it is impossible to create a trading strategy that confidently generates a profit. Seen from a forecasting perspective, the consequence of the Efficient Market Hypothesis is that

we are not able to forecast asset returns, as this would allow us to create a profitable trading strategy. The information set in the three variants of the Efficient Market Hypothesis can be summarized as follows; the information set in the weak form hypothesis contains only present and past asset prices, that in the semi-strong form contains also all publicly available information, and the strong form variant is concerned with all information available, both private and public (Timmermann & Granger, 2004, p. 17). As we are approaching the forecasting of volatility using only past and present asset prices, all forms of the Efficient Market Hypothesis are relevant for the analysis. It has been shown numerous times that, while the strong form version is unlikely to hold, the weak form and in part the semi-strong form of the Efficient Market Hypothesis tend to hold (Fama, 1970, 413-416). The results of the efficient market hypothesis is important to this study, as it implies that we are not able to completely forecast the returns of the asset in question. Rather, we are trying to forecast the volatility of said asset returns and studies have shown that the volatility of asset returns are partially predictable (Timmermann & Granger, 2004, p. 25). The partial predictability of asset return volatility stems, in part, from the fact that, although volatility is unobservable, we are able to extract some stylistic facts about the asset return volatility (Poon & Granger, 2003, p. 481). These stylistic facts of volatility will be introduced in Section 2.1.2.

2.1.1 Realized Volatility

The estimation of the unobservable volatility of financial assets is usually done using either conditional return variance, implied volatility or a realized measure. Probably the most used estimator for volatility is based on the historic returns of an asset. This estimation, however, is not an unbiased estimator of volatility, and is conditionally biased based on the data used for estimation, i.e. the return history. Implied volatility is based on option pricing models, where given observed derivatives prices, one can solve the option pricing models resulting in the implied volatility (Andersen, n.d., p. 1-2) As the implied volatility is estimated using a specific option pricing model, this volatility estimate is model dependent, which in turn means that it is a biased estimator of volatility. The availability of high-frequency price data has made realized measures of volatility more feasible. Realized volatility is the realization of the return variation of a given asset over a given time horizon (Andersen, n.d., p. 1-7). As stated in the research statement, this thesis revolves around forecasting the realized volatility of cryptocurrencies. Thus, this section will introduce the theory behind realized volatility.

To describe the realized volatility we consider the continuously compounded log-price process ds_t in given as the stochastic differential equation:

$$ds(t) = \mu(t)dt + \sigma(t)dW(t) \tag{2.3}$$

where $\mu(t)$ is the instantaneous drift and $\sigma(t)$ the instantaneous volatility of the log-price process (Andersen, n.d., p. 3-5). dW_t is a continuous-time Brownian motion, also referred to as a Wiener process, with the following properties given in (Campbell et al., 2012, p. 346-349):

$$E[dW(t)] = 0 \tag{2.4}$$

$$Var[dW(t)] = dt \tag{2.5}$$

The above properties, along with the rules of stochastic differential equations, yield the following

$$(ds(t))^2 = (\mu dt + \sigma dW(t))^2 = \sigma^2 dt \tag{2.6}$$

which shows that the squared instantaneous log-price process is given by the differencing interval dt and the squared instantaneous volatility σ_t^2 (Campbell et al., 2012, p.346-349). The primary concern of this thesis is the day-ahead forecasting of daily volatility, thus, the variable of interest is the daily volatility of the returns, not the instantaneous volatility. The return over a given period from $t-h$ until t is given by

$$r(t,h) = s(t) - s(t-h) = \int_{t-h}^{t} \mu(\tau)\, d\tau + \int_{t-h}^{t} \sigma(\tau)\, dW(\tau) \tag{2.7}$$

Thus, the return of an asset over a period from $t-h$ to t is given by the finite integral from $t-h$ to t of the conditional instantaneous drift and volatility of the log-price process (Andersen, n.d., p. 5). The quadratic variation of the return from $t-h$ to t is then given by

$$QV(t,h) = \int_{t-h}^{t} \sigma^2(\tau) d\tau \tag{2.8}$$

The quadratic variation is especially interesting to our studies, as it is equal to the conditional variance, only diverging by an error with zero mean. While the conditional variance is not observable, as stated in Section 2.1, the quadratic variance is observable with the access to high-frequency returns (Andersen et al., 2001, p. 44). In the diffusive process under consideration the quadratic variance and conincides with the integrated volatility

$$IV(t,h) = \int_{t-h}^{t} \sigma^2(\tau) d\tau \tag{2.9}$$

which is the variation of the process between $t-h$ and t (Andersen, n.d., p. 5). The integrated volatility, given as $RV(t,h) = \sigma(t,h) = \sqrt{IV(t,h)}$, over one day is the variable this thesis is concerned with. Trading, however, does not happen in continuous time, and we are thus forced to use an approximation of the integrated volatility. The integrated volatility has been shown to be approximated by the square root of sum of squared returns over the given period:

$$RV(t,h) = \sqrt{\sum_{j=0}^{M-1} r_{t-j\Delta}^2} \tag{2.10}$$

where M is the number of observations in over the time period, and Δ the length of each observation (Corsi, 2005, p. 8). This approximation has been shown to theoretically achieve arbitrary accuracy when $M \to \infty$ and thus the length of each observation tends to zero $\Delta \to 0$ (Andersen et al., 2001, p. 44).

In the real markets, however, the increase in sampling frequency does have its limits. The trading that happens on in the markets are limited by the institutional structures that they happen within. Prices are not real valued, that is they are often given to a specified precision, trading

does not happen continuously or even at evenly spaced intervals, and market makers impose the bid-ask spreads on the assets traded (Campbell et al., 2012, p. 84). These factors, referred to as market microstructures, impose a challenge to precisely approximating the integrated volatility by use of realized volatility. The choice of sampling frequency thus comes down to minimizing the noise imposed by market microstructures and minimizing the sampling frequency (Andersen et al., 2001, p. 48). Andersen et al. (2001) proposed a sampling frequency of five minutes, to minimize both of the above challenges. When used realized volatility is used as a benchmark, as is true in this thesis, the 5-min sampling frequency is found to be fairly efficient, not being outperformed in many situations (Liu et al., 2015, p. 309-310).

The above variance measures and approximations hold for an ordinary continuous-tim diffusion process. This requires the assumption that the log-price must follow a continuous sample path process. But, this process is often violated in practice, due to the presence of jump components in the log-price process (Andersen et al., 2007, p. 701-702). To include these jumps we assume that the log-price process follows a continuous-time jump diffusion process

$$ds(t) = \mu(t)dt + \sigma(t)dW(t) + \kappa(t)dq(t) \tag{2.11}$$

where $\mu(t)$ is again the instantaneous drift, $\sigma(t)$ the instantaneous stochastic volatility process and $W(t)$ a Wiener process. The last term $\kappa(t)dq(t)$ is the jump component, where $\kappa(t)$ is the size of the corresponding discrete jumps and $q(t)$ is a counting process with time-varying intensity (Andersen et al., 2007, p. 702). As the jump component is a discrete process, the quadratic variation for the cumulative return in the period from $t-h$ to t is now

$$QV(t,h) = \int_{t-h}^{t} \sigma^2(\tau)d\tau + \sum_{t-h<\tau\leq t} \kappa^2(\tau) \tag{2.12}$$

Thus, the quadratic variation of the cumulative return process is given by the integrated variance of the original diffusive process and the sum of jumps over the period (Andersen et al., 2007, p. 702). The presence of a jump component in the quadratic variance in turn means that the realized variance estimate must include both the variation occurring from the diffusive process and the jump process. This does not change the estimation method, which is still based on the sum of squared returns, but it changes the modeling and forecasting of this estimate. The modeling and forecasting methods will be further developed in Chapter 2, the methodology.

We have shown in this section that using the realized variance we can obtain good estimates of the quadratic variation of a continuous-time diffusive process and a mixed jump diffusion model, increasing the goodness of the estimate by increasing the sampling frequency. Market microstructure noise does however impose some challenges to the increasing sampling frequency, and thus to minimize both the sampling frequency and the noise from market microstructures, we choose a sampling frequency of five minutes. Using realized volatility as our volatility estimate is both motivated by the availability of high-frequnccy returns and the applicability of realized volatility in out-of-sample forecasting. Here realized volatility has been shown to outperform GARCH models and stochastic volatility mdoels (Andersen et al., 2007, p. 701). The next section will explain the characteristics of the volatility of financial assets, which guides the model selection in the thesis.

2.1.2 Stylistic Facts of Volatility

The volatility of a financial asset is not directly observable, although we have rather effective approximations of it. The volatility of a return process, does however show some well-established characteristics, which may guide us in estimation evaluation, model selection and forecasting (Poon & Granger, 2003, p. 481-482). This section will introduce the characteristics, also known as stylized facts, of the volatility of asset prices.

Volatility describes the variation of asset prices over a given time interval. Due to the behavior of investors, underlying structures, market sentiment etc. we expect that the volatility of an asset may be higher in some periods a lower in others. This effect, known as persistence or volatility clustering, has been shown empirically to hold for the volatility of asset prices (Engle & Patton, 2000, p. 239). Essentially, volatility clustering is the phenomenon where small changes in price are expected to be followed by small changes in price, and large changes in price be followed by large changes in price. Thus, the magnitude of the volatility is expected to be highly correlated to the previous values of the volatility.

The magnitude is however not expected to remain at the same level indefinitely. This is due to the stylistic fact that volatility is mean reverting. This means that even though high volatility is expected to be followed by high volatility, and low levels with low volatility, we may expect that the volatility of a financial asset reverts to some normal level (Engle & Patton, 2000, p.239). Thus, in the long term, volatility is expected to fluctuate around some arbitrary average value, with periods of high volatility and periods of low volatility.

The fluctuations around the long-term mean of the volatility process, is however not expected to follow a Gaussian distribution. Instead, extreme values in volatility tends to be much more likely, than in the case of a Gaussian distribution (Engle & Patton, 2000, p. 240). Furthermore, the sign of the returns of the asset prices carry a great weight in the volatility process. High negative returns are expected to cause greater increases in volatility than high positive returns. This can be explained by investors seeing increased debt-to-equaty ratios (Engle & Patton, 2000, p. 239). Thus, the risk of investors are even further increased in the presence of negative returns than in the presence of positive returns, even though they might imply the same rise to volatility in the outset.

Lastly, the publication of a company's annual reports or publication of macroeconomic announcements may entail greater volatility (Engle & Patton, 2000, p. 240). These are effects of exogenous variables on the volatility of a financial asset. The exogenous variables themselves might not convey information that should raise the volatility, but the mere expectation about what the outcomes of these exogenous variables will increase volatility.

The above stylistic facts about volatility show us, that, even though we cannot observe volatility directly, some characteristics are exposed when the process is analyzed. In general volatility is expected to revert to some long-term average, with fluctuations governed by clustering. These fluctuations are influenced by both exogenous variables and the realizations of the return process itself. And, lastly, the volatility process is expected to show more cases of extreme levels than what would be expected if they were distributed as a Gaussian distribution. These stylistic facts of volatility hold true for any financial asset. We are however especially interested in the volatility process of cryptocurrencies, and hence the next section will be devoted to examining

cryptocurrencies as financial assets and their characteristics.

2.2 Cryptocurrencies as a Financial Asset

Cryptocurrencies were, as explained in the introduction, invented as a means of transaction, a decentralized digital currency, challenging the ordinary fiat currencies (Kristoufek, 2015, p. 1). With the potential to facilitate digital payments to happen without the need for a financial institution, cryptocurrencies were promising alternative currencies, for a more efficient financial system (Nakamoto, 2008, p. 1). The status as a currency though, has been challenged by many. This section, will not be focused on whether cryptocurrencies can be seen as a currency, but rather how these cryptocurrencies behave. Understanding this behavior will yield a foundation for model selection and evaluation in the analysis to come.

In financial research and practice assets are usually classified as belonging to some category of assets, such as commodities, equity, currencies etc. The category a certain asset belongs to usually gives indications about the driving forces and behavior of the price of said asset. Cryptocurrencies are hard to classify though (Aalborg et al., 2019, p. 255-256). Some studies argue that cryptocurrencies behave similarly to gold, a commodity, but also the US dollar, a currency (Kristoufek, 2015, p. 12). Yet others show that cryptocurrencies does not behave like any traditional asset group, but should rather be seen primarily as a speculative investment Baur et al. (2018). Thus, we cannot rely on the characteristics of traditional asset classes to describe the behavior and drivers of cryptocurrencies. But, cryptocurrencies have been shown to behave in certain ways, giving way for their own stylized facts Zhang et al. (2018).

When modeled with a GARCH(1, 1) model, cryptocurrencies exhibit significantly positive coefficients α and β. This indicates that the price processes of cryptocurrencies possess strong volatility clustering (Zhang et al., 2018, p. 5957). The presence is further backed by modeling volatility using variants of the GARCH model, all identifying volatility clustering (Takaishi, 2018, p. 516). Modeling with the GJR-GARCH further facil

Cryptocurrencies are found to display strong volatility clustering, while research find varying degrees of leverage, some being positive and others negative (Zhang et al., 2018, p. 5955-5957). A study focusing on the leverage effect of the 20 largest cryptocurrencies, by market capitalization, actually found negative leverage effects. That is, positive return shocks influence the volatility more than negative returns Baur & Dimpfl (2018). The authors argue that the negative leverage effects *"... can be explaiend with uninformed investors' herding, buying due to the fear of missing out on rising cryptocurrency valuation and pump and dump schemes"* (Baur & Dimpfl, 2018, p. 151). Thus, further backing the claim that cryptocurrencies should be seen as a speculative investment. Furthermore, cryptocurrency return series show heavy tails, thus, experiencing extreme outcomes more frequently than would be the case in a Gaussian process. The absolute returns are also heavily autocorrelated, while the long-range dependence of cryptocurrencies vary between the individual cryptocurrencies. It is further shown, that the volume and price of cryptocurrencies are shown to be power-law cross-correlated, underpinning the investor herding and fear of missing out explaining the negative leverage effects Zhang et al. (2018). Lastly, Bitcoin daily returns have been found to exhibit a varying mean volatility along with discontinuous

return jumps (Chaim & Laurini, 2018, p. 162-163). Thus, when modeling the volatility of cryptocurrency returns, one must account for the presence of discontinuous return jumps.

Thus, cryptocurrencies show many of the stylized facts about volatility described in Section 2.1.2. Interestingly however, cryptocurrencies tend to see negative leverage effects, that might be explained by investor herding and pump and dump schemes. Furthermore, Bitcoin return series displays discontinuous jumps, why it is deemed important to consider this in the modeling of realized volatility.

2.3 Deep Learning

This section will introduce the central concepts of deep learning, the frequently used models for time series regression and the foundation of the architecture used in this thesis.

2.3.1 Neural Network Architecture

The basic building block of a neural network is the neuron. A single neuron takes a set of inputs X and returns a single output y. The neuron processes the information based on the weights and biases of the neuron, as well as the activation function. Formally, a neuron can be described as follows

$$y = h\left(\sum_{i=1}^{I} w_i x_i + w_0\right) \tag{2.13}$$

where I denotes the number of inputs and $h(\cdot)$ denotes the activation function (MacKay, 2003, p. 471). A simple feed-forward neural network achieves the ability of modeling non-linear relationships through the activation function. The most used activation function today is the rectified linear unit (ReLU):

$$ReLU(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \tag{2.14}$$

A single neuron does not make a network, and thus to achieve the network-like structure multiple neurons are connected to form a neural network. The architecture of a neural network consists of several layers; the input layer, the output layer, and either a single or multiple hidden layers. Each hidden layer is made from a stack of neurons, where the number of neurons in each layer, denotes the width of the layer. The number of hidden layers in a neural network specifies the depth of the network (Goodfellow et al., 2016, p. 168-171) In a simple feed-forward neural network each neuron takes all the outputs of the previous layer, transforms it according to Equation 2.3 and returns a scalar (Bishop, 2006, p. 227-231). A single layer in the neural network can then be described as a set of M neurons

$$y_j = h\left(\sum_{i=1}^{I} w_{ji} x_i + b_j\right) \tag{2.15}$$

where $j = 1, 2, ..., M$ and b is the bias of each neuron. A typical neural network contains more than one hidden layer, and is often described as a set of composable transformations. Thus, if we have a neural network with two hidden layers it can be described by the function

$$y_k = h\left(\sum_{i=1}^{I} b_k^{(2)} + w_{kj}^{(2)}\left(h\left(\sum_{i=1}^{I} b_j^{(1)} + w_{ji}^{(1)} x_i\right)\right)\right) \tag{2.16}$$

where k denotes the number of neurons in the second layer, and j the number of neurons in the first (Bishop, 2006, p. 227-231). The type of neural network described by the above equation is known as a feed-forward network or a multilayer perceptron. This terminology can be simplified by describing a neural network as a chain of non-linear functions, transforming the set of input variables $\{x_i\}$ to a set of output variables $\{y_k\}$ (Bishop, 2006, p. 227-231). Representing each layer as a function $f^{(\cdot)}(\cdot)$ we can write a neural network of depth three as:

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))) \tag{2.17}$$

where $f(x)$ then denotes the full network architecture.

A neural network is a type of machine learning, and thus the neural network must be able to learn from the set of inputs and outputs in our data. This is achieved by optimizing the weights $\{w\}$ and biases $\{b\}$ of the neural network according to some loss function $J(\theta)$, where θ represents the weights and biases of the network (Goodfellow et al., 2016, p. 171). A common loss function for neural networks used for regression tasks is the mean squared error (MSE)

$$J(\theta) = \frac{1}{N}\sum_{n=1}^{N}(y_n - f(\mathbf{x_n}; \theta))^2 \tag{2.18}$$

where $n = 1, 2, ..., N$ is the number of examples in our training set. The minimization of the loss function is implemented using back-propagation and an optimizer, such as stochastic gradient descent (SGD). Optimization happens by back-propagating the gradients of the loss function with regards to the parameters θ.

Forward propagation is the transformation of the set of input variables $\{x_i\}$ to the set of output variables $\{y_k\}$, which in training is propagated further to the loss function $J(\theta)$. The learning of the neural network then happens by using the gradients, computed by back-propagation, to update and eventually minimize the expectation of the loss function $J(\theta)$. Essentially, back-propagation is computing the vector of gradients of the loss function with regards to the parameters, $\nabla_\theta J(\theta)$, through the chain-rule of calculus (Goodfellow et al., 2016, p. 204). The difference between applying the chain-rule directly to each parameter and using back-propagation, is that back-propagation computes the gradient in a computationally efficient way.

In the minimization of the loss function $J(\theta)$ we are not computing the gradients of the loss function using the entire set of inputs and outputs. Instead, we are back-propagating the loss function through the network using a randomly chosen subset of the entire set of inputs and outputs. Doing this is computationally more efficient, and allows us to perform more optimization steps in a shorter period of time. The method of minimizing the loss functions with regards to only a sub-sample of inputs and outputs is referred to as stochastic gradient descent (Goodfellow et al., 2016, p. 277-282).

Much research has been done trying to optimize the optimization algorithm, with alternatives to SGD such as; AdaGrad, AdaDelta, RMSProp, etc. (Kingma & Ba, 2017, p. 4-5). Today the most used optimization algorithm is the Adam optimizer introduced in 2015. It differs from the classic SGD algorithm in using exponentially moving averages and squared gradients to update the weights Kingma & Ba (2017).

In constructing the network architecture and learning rules of a neural network, there are some important hyperparameters that must be optimized. These include, but are not limited to, the batch size of the randomly chosen sub-samples of the set of inputs and outputs. The optimization algorithm and the learning rate of the optimization algorithm. The epochs of the training loop, that is the number of times the training should iterate over the entire dataset. There are guidelines to choosing these, but in general the final choice must be found through an iterative trial-and-error process (Goodfellow et al., 2016, p. 427-429).

Deep learning is often touted as being black-box methods that perform well at many things. The Universal Approximation Theorem states that the feed-forward neural networks described above are able to approximate any function going from a finite-dimensional input to a finite-dimensional output given enough neurons Hornik et al. (1989). While this is shown to be true, the number of hidden units needed often becomes so large that it is computationally infeasible to use the network (Goodfellow et al., 2016, p. 199-200). Thus, to construct neural networks that are useful in an applied setting, one must pay careful attention to the architectural choices being made in the construction. The next section will elaborate on the common deep learning models used in time series regression problems.

2.3.2 Deep Learning in Time Series Forecasting

Machine learning, including deep learning, has been a subject of great study within the time series forecasting community since its inception (Sezer et al., 2019, p. 3-5). Several deep learning architectures have been used for time series forecasting including; deep multilayer perceptrons, recurrent neural networks, convolutional neural networks, restricted boltzmann machines, deep belief networks, autoencoders and deep reinforcement learning Sezer et al. (2019). The architecture of the deep multilayer perceptron is described in Section 2.3.2. The other architectures use the basic principles from deep multilayer perceptrons, but vary in the way the forward propagation is done, and how the input is transformed. Traditionally, recurrent neural networks (RNNs) have been used with great success on sequence data, such as textual and speech data in natural language processing (Lim & Zohren, 2021, p. 4). Thus, as time series essentially are represented as a sequence of data, RNNs have seen a lot of use in the domain of time series forecasting. In 2017 the natural language processing community witnessed a revolution to the model architectures used, with the introduction of the Transformer architecture Vaswani et al. (2017). The Transformer architecture is now the foundation for most state-of-the-art model in natural language processing. Due to the success of the Transformer architecture and the underlying attention mechanism, the application of these in time series regression has been researched since their inception Lim & Zohren (2021). Another application domain of deep learning is the one dealing with visual data, from image recognition, to object detection in videos. This domain has been dominated by convolutional neural networks since the outstanding performance of AlexNet

in 2012 Krizhevsky et al. (2012). Where convolutional neural networks are usually done on two-dimensional data (images etc.) they have seen some applications in time series regression. To use convolutions on time series one method is to transform the time series into two-dimensional representations using Markov Transition Fields or Gramian Angular Fields Wang & Oates (n.d.). Another, more commonly used method is to use one-dimensional convolutional neural networks on time series data, where DeepMind developed the WaveNet model for generating raw audio van den Oord et al. (2016). Other model architectures are used for time series as well, but due to the novelty and rationale of the convolutional network for time series forecasting and modeling, it will be the one used in this thesis, along with a standard multilayer perceptron. The next section will introduce the convolutional neural network and the architectural choices made when using it on one-dimensional time series data.

2.3.3 Convolutional Neural Networks

In the multilayer perceptron described in Section 2.3.2 a neuron in a given layer is connected to all neurons in the previous layer with its activation computed as a simple matrix multiplication and an activation function. Convolutional neural networks (CNNs) are as the multilayer perceptron a type of feed-forward neural network, in that it does not contain any recurrent connections or feedback loops. CNNs have proven useful and extremely successful in a wide arrangement of tasks, most noticably computer vision. They achieve their efficiency by employing a special mathematical operation on the input called a convolution (LeCun et al., 2015, p. 326). These convolutions replace the fully connected matrix multiplication that drives the multilayer perceptron, and are well-suited for structured data, such as time series, a one-dimensional grid of observations, or images, a two-dimensional grid of pixels. The basis for the convolutional neural networks is found in the brains primary visual cortex, where the neurons responds to very simple abstract structures in the light coming in through the eyes. This is done through local receptive fields, that focus on a small part of the input, scanning the entire visible area, and summing it up

To define the convolutions consider the following function

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a) * w(t-a) \tag{2.19}$$

where, $s(t)$ is the output of the convolution at time step t and a is the size of the receptive field, or kernel size (LeCun et al., 2015, p. 332). A convolution layer is then the application of the convolution function at all time steps in the input. The trainable parameters of the convolutional layer is the parameters of the function $x(a)$, thus, for a kernel with a size of a we have a trainable parameters in the convolution layer. The benefit of the convolutional layer is its ability to process input in a grid-like topology with fewer parameters and more efficiency.

Dilated Causal Convolutional Neural Networks

One issue of the convolutional layer when used in forecasting applications is that when processing data for time t it utilizes both data before and after the time. This is due to how the convolution is implemented and how the receptive field works. In image processing this is perfectly fine, but

when processing sequential data, this poses challenges, as we wish to estimate a forecast based on all the information available at time t and before. To overcome this issue the convolutional layers used in this thesis is of the form dilated causal convolutions, inspired by the WaveNet algorithm of Google Deep Mind (van den Oord et al., 2016, p. 2). The dilated causal convolutions used in the WaveNet algorithm, and in the algorithm used in this thesis, only takes in information from $t - a$ to t, thus ensuring that only known information at time t is used (van den Oord et al., 2016, p. 2). Furthermore, multiple convolutional layers is stacked on top of each other, with an increasing factor of dilation. This means that in the first layer, the convolution is run on all inputs, in the second the convolution is run on every second input and so forth (van den Oord et al., 2016, p. 2). This enables the neural network to have a very large receptive field, without needing too many parameters, and thus allowing the network to be efficient. The dilated causal convolutional neural network will be one of the two primary neural networks that will be compared to the standard econometric models in this thesis.

Chapter 3 Exploratory Data Analysis

This section will introduce the data that are being used throughout the analysis. The different time series will be analyzed, through which a foundation for the model development will be made. Furthermore, the statitionarity of the time series will be tested.

3.1 Overview

The thesis is based on an evaluation of deep learning methods for forecasting realized variance in crypto currencies. The dataset used in the analysis consists of 1-minute price data from four crypto currencies: Bitcoin (BTC), Ethereum (ETH), Ada (ADA), and Binance Coin (BNB). The start date, end date, number of observations and the volume traded over the entire period is listed below for each asset.

	BTC	**ETH**	**ADA**	**BNB**
Start Date	2018-01-01	2018-01-01	2018-04-17	2018-01-01
End Date	2021-09-21	2021-09-21	2021-09-21	2021-09-21
Number of Observations	1,956,282	1,956,200	1,791,867	1,942,619
Volume	$1.816e^8$	$2.222e^9$	$5.930e^{11}$	$3.615e^9$

Table 3.1: Overview of dataset start and end dates, no. of observations, and volume traded in the period.

Bitcoin was, as mentioned in the introduction, started in 2008 with the whitepaper by Satoshi Nakamoto and is in June 2022 the largest cryptocurrency by market capitalization Nakamoto (2008); *Cryptocurrencies with Highest Market Cap - Yahoo Finance* (2022). Ethereum was introduced in 2014 by Vitalik Buterin and launched in 2015, with a market capitalization of $ 225 billion it is the second largest cryptocurrency Buterin (2013); *Cryptocurrencies with Highest Market Cap - Yahoo Finance* (2022). Ada is the native token of the Cardano blockchain platform, introduced in 2017 and currently the sixth largest cryptocurrency by market capitalization Hoskinson (2015); *Cryptocurrencies with Highest Market Cap - Yahoo Finance* (2022). Lastly, Binance Coin, the cryptocurrency released by the Binance Exchange in 2017, which currently sits as the fifth largest cryptocurrency *Binance Whitepaper* (2017); *Cryptocurrencies with Highest Market Cap - Yahoo Finance* (2022).

The development of the price of the cryptocurrencies can be seen in Figure 3.1. As seen, all cryptocurrencies experienced a massive price increase from the beginning of 2021 followed by a

large decline. The price path seen in the plots speak to the extreme behavior of cryptocurrency prices described in the introduction. Furthermore, we see that the 5-minute log-returns of the cryptocurrencies see spikes to absolute returns of up to 20 %.

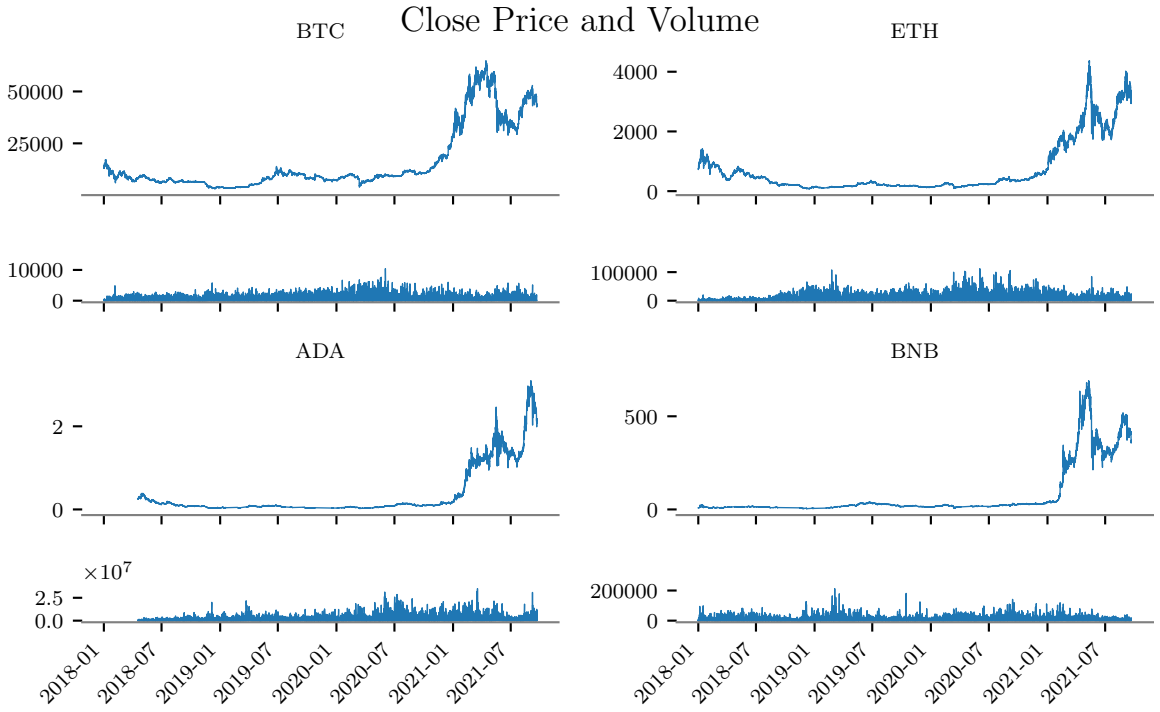

Figure 3.1: Close Prices and Volume of Assets (own production).

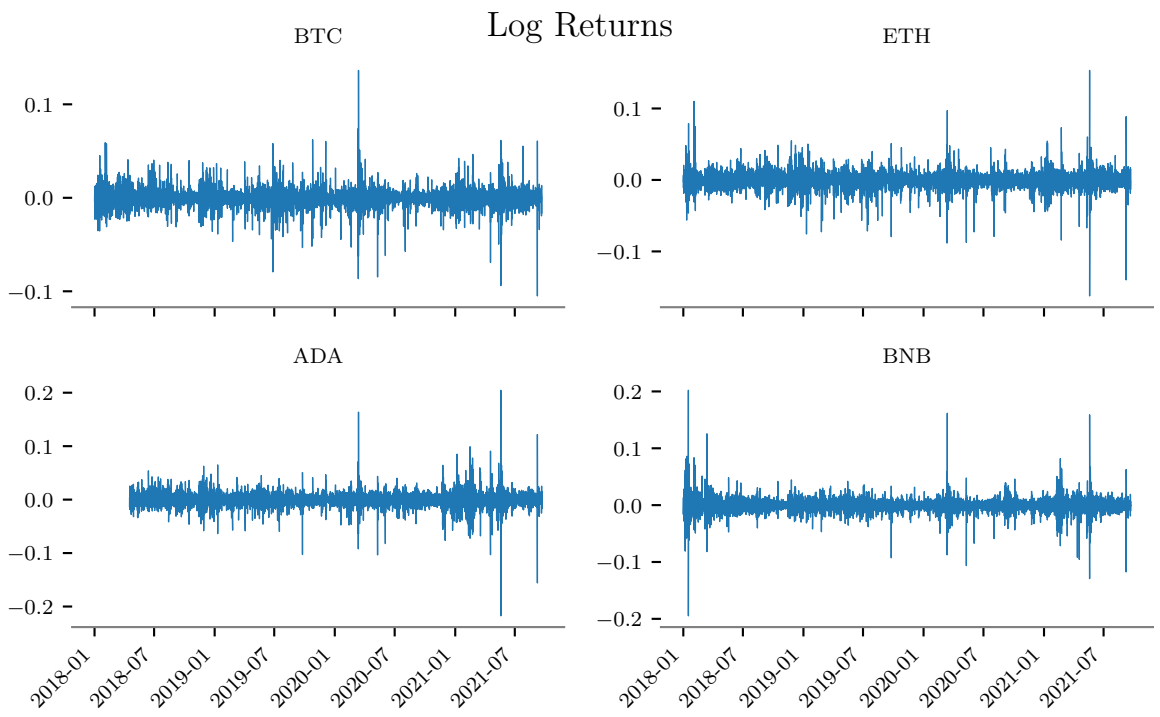

Figure 3.2: 5-min log-returns of cryptocurrencies (own production).

The realized volatility and log realized volatility of the cryptocurrencies are shown in Figure 3.3 and Figure 3.4, respectively. All cryptocurrencies exhibit rather volatile log returns as seen in Figure 3.2, and this shows in the realized volatilities. There are some things to note however. The realized volatility of all cryptocurrencies seem to fluctuate around some rather constant average. Thus, the stylized fact of volatility being mean reverting seems plausible in this case. The stylized fact of volatility clustering, suggests that high volatility should be followed by high volatility, and low volatility by low volatility. Our time series all appear to exhibit this characteristic, albeit it is rather vague, due to the volatility constantly being rather high.

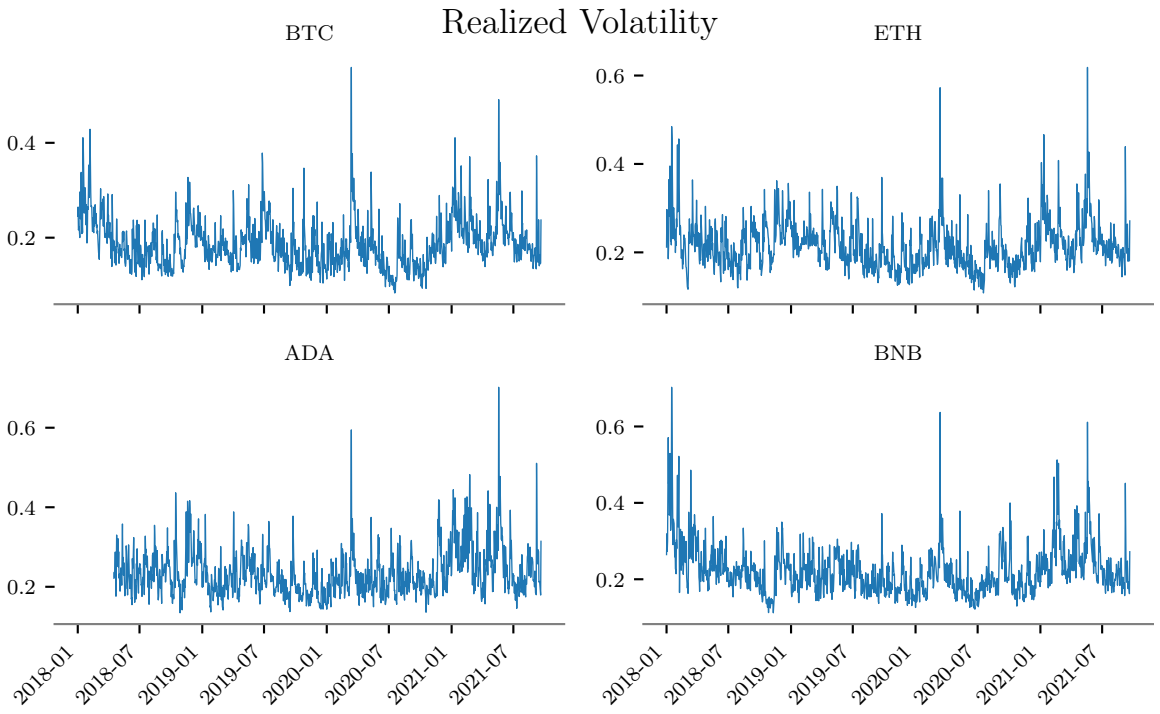

Figure 3.3: Realized Volatility of cryptocurrencies (own production).

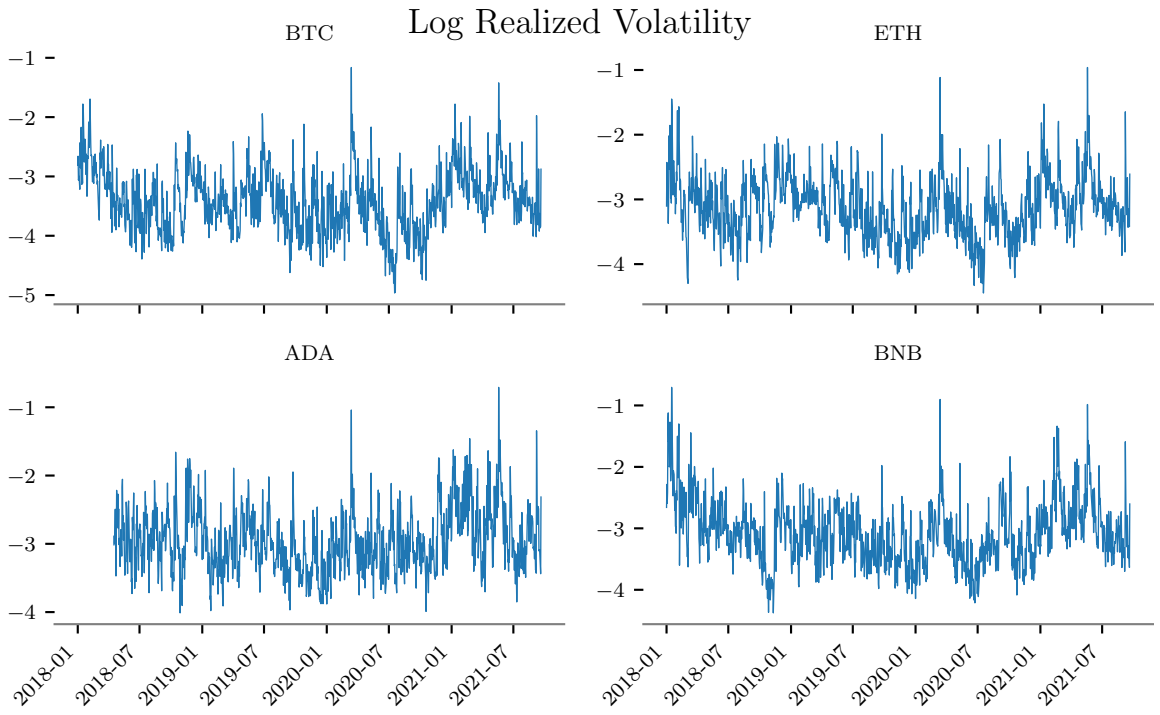

Figure 3.4: $\ln RV$ *of cryptocurrencies (own production).*

The characteristic that volatility is mean reverting is rather important for the ability to model it. In time series analysis we require that the time series we are modeling are stationary.

Stationarity can be defined in both its strict form and its weak form. Strictly stationary time series are defined by a having joint distributions that do not change over time. That is, for our time series of realized volatility $\{RV_t\}$ to be strictly stationary, the joint distribution of $(RV_{t_1}, ..., RV_{t_k})$ must be identical to that of $(RV_{t_1+t}, ..., RV_{t_k+t})$ (Tsay, 2005, p. 25). Weakly stationary time series does not require the joint distributions to be identical under time shift. Rather, a weakly stationary time series must meet two assumptions:

$$E[RV_t] = \mu \tag{3.1}$$

$$Cov[RV_t, RV_{t-l}] = \gamma_l \tag{3.2}$$

That is, the expectation of the realized variance must be constant, and the covariance between two observations in the realized variance must only depend on the lag l (Tsay, 2005, p. 25). The stationarity of a time series can be tested using and Augmented Dickey-Fuller test, by estimating the test regresssion

$$\Delta RV_t = \phi_0 + \gamma RV_{t-1} + \sum_{i=1}^{p} \phi_i \Delta RV_{t-i} + \varepsilon_t \tag{3.3}$$

where p is the number of lags to include in the ADF test (Tsay, 2005, p. 25). The null hypothesis in the ADF test is $H_0 : \beta = 1$ and the alternative $H_a : \beta > 1$. The test statistic used is thus

$$ADF - test = \frac{\hat{\beta} - 1}{std(\hat{\beta})} \tag{3.4}$$

The p-values of the ADF test for our four eight time series (realized volatility and log-realized volatility) is seen in Table 3.2. As can be seen with a confidence interval of 95 % we can reject the null hypotheses in all cases, and thus all our time series are stationary.

	Adjusted Dickey-Fuller			
	BTC	**ETH**	**ADA**	**BNB**
P-Value (RV)	0.000000	0.000000	0.000000	0.000003
P-Value ($\ln RV$	0.000000	0.000000	0.000000	0.000000

Table 3.2: Augmented Dickey-Fuller test for RV and lnRV (own production)

Closely related to the autocovariances of described above is the autocorrelation of the observations in the time series. As described in Section 2.2 the absolute return series of cryptocurrencies typically exhibits strong autocorrelations. Autocorrelation can be defined as being the correlation between two observations of the same time series but at different times. We have shown that the realized volatility and log-realized volatility series we are working with are weakly stationary, and there fore, that the autocovariance is only dependent on the lag l between two observations, and not the time of the observations. Thus, the autocorrelation between observations of

the realized volatility is only dependent on the lag l too. Therefore, we can describe the lag-l autocorrelation of our realized volatility (and log-realized volatilit) series as

$$\rho_l = \frac{Cov(RV_t, RV_{t-l})}{\sqrt{Var(RV_t)Var(RV_{t-l}}} = \frac{Cov(RV_t, RV_{t-l})}{Var(RV_t)} = \frac{\gamma_l}{\gamma_0} \tag{3.5}$$

where for weakly stationary series $Var(RV_t) = Var(RV_{t-}$ (Tsay, 2005, p. 26). The autocorrelation function is used to create the autocorrelation function (ACF) plots seen in Figures 3.5 and 3.6. The blue shaded area of the ACF plots show the significance level, of at the given lag. Note that all realized volatility time series have significant autocorrelations up until $l = 30$. BTC realized volatility shows the longest dependence of the four cryptocurrencies, while ADA shows the shortest. Albeit, ADA does also have significant autocorrelations at $l = 26$ and $l = 30$. The autocorrelations of the log-realized volatility time series are in general slower decaying, which is expected as it smoothes the distributions.

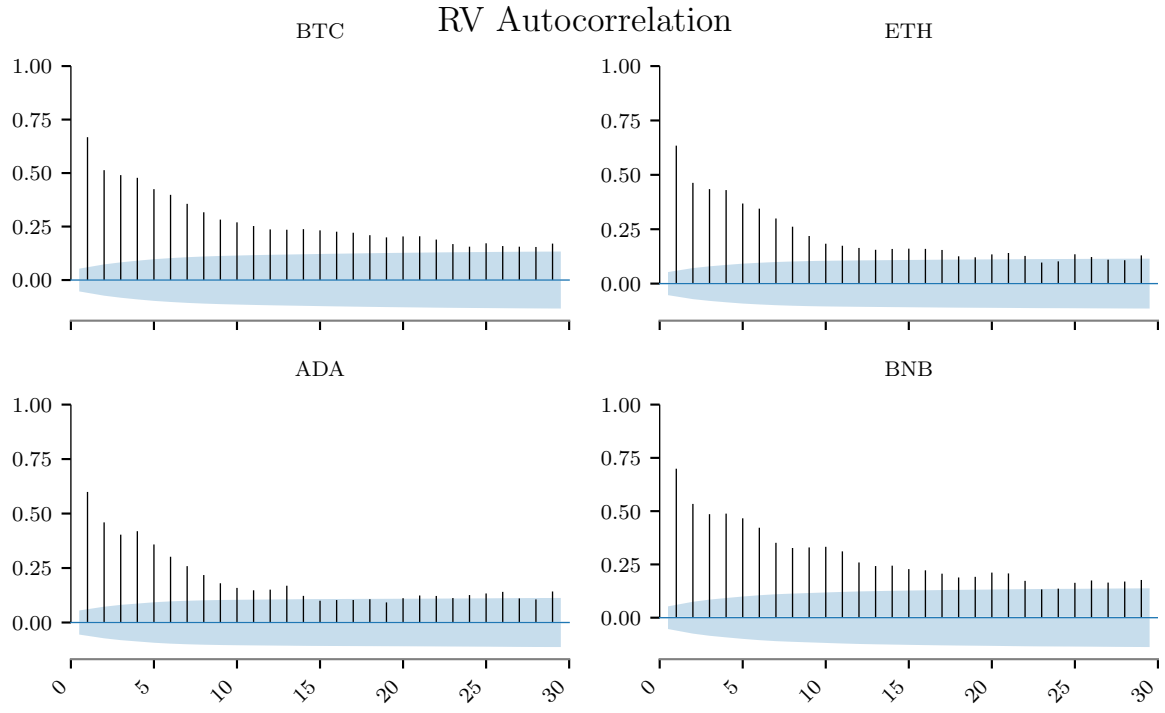

Figure 3.5: Autocorrelation function of realized variance (own production).

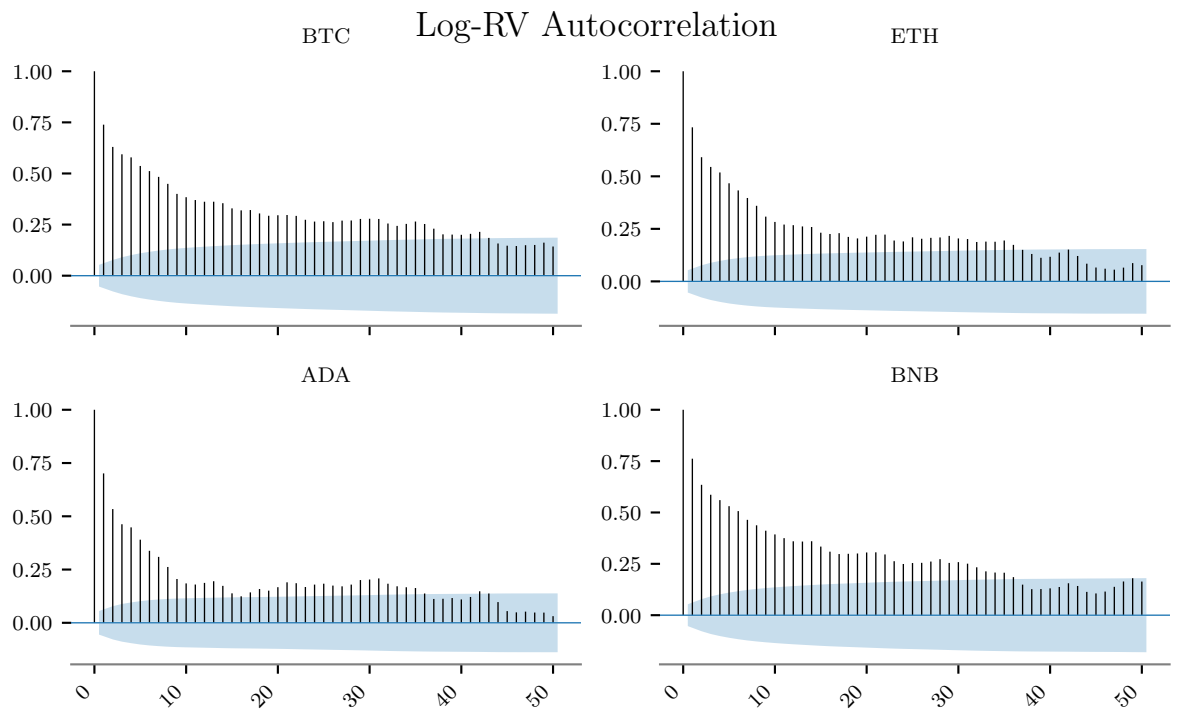

Figure 3.6: Autocorrelation function of $\ln RV$ *(own production).*

3.2 Datasets

This section will describe the datasets used for training and testing the models of the thesis. The datasets consist of 1359 (1253 days for ADA) days of 1-minute observations each. Each dataset is split in a training series and a testing series. The first 30 observations are not part of either the training and testing series, as some computations uses rolling means, and thus are only viable from day 30 and forward. The last observation is also removed. This results in the training and testing split seen in Tables 3.3 through 3.6.

	BTC	
	Train	**Forecast**
No. Observations	930	398
Min	0.006976	0.008646
Max	0.312463	0.241197
Mean	0.035595	0.040740
Variance	0.000515	0.000600
Skewness	3.812920	2.763470
Kurtosis	30.619722	14.010251

Table 3.3: BTC dataset statistics (own creation)

	ETH	
	Train	**Forecast**
No. Observations	930	398
Min	0.011718	0.014873
Max	0.327716	0.382379
Mean	0.046842	0.053152
Variance	0.000653	0.001090
Skewness	3.073369	3.912822
Kurtosis	21.284050	27.439998

Table 3.4: ETH dataset statistics (own creation)

	ADA	
	Train	**Forecast**
No. Observations	855	367
Min	0.018103	0.018494
Max	0.353426	0.492362
Mean	0.053065	0.070949
Variance	0.000728	0.002067
Skewness	3.082102	3.240852
Kurtosis	21.198474	20.739392

Table 3.5: ADA dataset statistics (own creation)

	BNB	
	Train	**Forecast**
No. Observations	930	398
Min	0.012578	0.016822
Max	0.405579	0.373495
Mean	0.048076	0.060885
Variance	0.000838	0.001611
Skewness	4.101679	2.917429
Kurtosis	34.129322	13.107855

Table 3.6: BNB dataset statistics (own creation)

As can be seen in Tables 3.3 through 3.6 the mean and variance of the four time series generally increases from the training dataset to the testing dataset. This might pose a challenge when forecasting the realized volatility, as the dataset that the models are trained on differs in the first and second moment from the dataset that they are tested on.

From the skewness of the eight datasets we can conclude that they are all highly skewed to the right. That is, in all datasets most observations occur around some normal level, the long-term average. The realized volatility cannot decrease below zero, but is able to peak to extreme values. Thus, the right skewness of the time series is to be expected.

All our dataset are also highly leptokurtic as the kurtosis far exceeds that of a normal distribution, where the kurtosis is three. This indicates that the distribution of the realized volatility in our datasets are heavy tailed, and more suceptible to extreme values. According to the stylistic facts of both volatility and cryptocurrency returns mentioned in Section 2.2 we

expect the distributions to have heavy tails, and thus, this confirms that expectation.

The datasets introduced above all show stationary realized volatility and log-realized volatility. Furthermore, they all exhibit heavy tails, volatility clustering, and right skewness. The econometric and deep learning models used for forecasting the realized volatility will be described in the next section.

Chapter 4 Methodology

4.1 Models

4.1.1 Linear Time Series Models

All econometric models used in this thesis are based on the Heterogeneous Autoregressive model of realized volatility (HAR). The variations in the models are based around the stylized facts of volatility that has been previously described. This section will introduce these econometric models.

HAR

The Heterogeneous Autoregressive model of realized volatility (HAR) is the de facto standard for modeling realized volatility, as mentioned in Section 1.2. The HAR model is motivated by considering that the financial markets are heterogeneous, that is, the agents in the financial markets are not all alike. This is grounded in the Heterogeneous Market Hypothesis (Corsi, 2008, p. 178-179). The Heterogeneous Market Hypothesis deserves some study, as it is central to all the econometric models applied in this thesis. The hypothesis states that the actors of financial markets do not all have the same trading horizons and frequencies. Secondly, increases in actors in the market and trading volume increase the volatilty of the market. This is seen against the logic of the homogeneous market, where one would assume that more actors and volume would decrease the volatility, as the price would reach the intrinsic value faster. Lastly, markets and their memory component is also influence by the geographic locations of the traders Müller et al. (1993).

The central concept that is incorporated into the HAR model is that of different time horizons and frequencies. Actors trading with different time horizons all influence the market differently and therefore the overall volatility of the market is deemed to be aggregated from these components. These market components will be referred to as the latent partial volatility $\tilde{\sigma}_t^{(\cdot)}$ (Corsi, 2008, p. 179). The HAR model is then constructed from three of theses latent partial volatilities: $\tilde{\sigma}_t^{(d)}$, $\tilde{\sigma}_t{}^w$, and $\tilde{\sigma}_t{}^m$, where d is daily, w is weekly, and m is monthly. To ensure that the volatilities are comparable they are scaled to match the daily volatility. Thus, the final model becomes:

$$RV_{t+1d}^{(d)} = c + \beta^{(d)} RV_t^{(d)} + \beta^{(w)} RV_t^{(w)} + \beta^{(m)} RV_t^{(m)} + \omega_{t+1d}^{(d)} \tag{4.1}$$

The computation of $RV_t^{(w)}$ and $RV_t^{(m)}$ is given below:

$$RV_t^{(w)} = \frac{1}{5}\sum_{i=1}^{5} RV_{t-i} \tag{4.2}$$

$$RV_t^{(m)} = \frac{1}{22}\sum_{i=1}^{22} RV_{t-i} \tag{4.3}$$

The periods chosen for the weekly and monthly component is based on the average days in a trading week and month, respectively. Since this study is centered around forecasting bitcoin volatility, which trades 24 hours a day 365 days a year, the periods will be 7 and 30.

The HAR model is essentially a multivariate linear regression with similarities to an AR(1) model, in that it uses one-period lagged values of the realized volatility Corsi (2008). This model have some restrictions however. In particular, the model is not able to incorporate jumps and leverage effects (Xu & Wang, 2014, p. 1303) (Bollerslev et al., 2020, p. 8). Thus, the HAR variants HAR-J and HAR-RS-I will be implemented, to be able to better model the volatility, and its stylized facts, of cryptocurrencies, as described in Section 2.2. These models have shown to be better able to capture the fat tails, jumps and leverage effects (albeit negative) as cryptocurrencies exhibit Xu & Wang (2014)Bollerslev et al. (2020).

HAR-J

Considering an asset that whose log-price process follows a jump diffusion process, described in Section 2.1:

$$ds(t) = \mu(t)dt + \sigma(t)dW(t) + \kappa(t)dq(t) \tag{4.4}$$

The quadratic variation of this process is then defined to be

$$QV(t,h) = \int_{t-h}^{t} \sigma^2(\tau)d\tau + \sum_{t-h<\tau\leq t} \kappa^2(\tau) \tag{4.5}$$

.

Incorporation of jumps in the price process of an asset have been shown to be important in modeling its behavior, and further, as mentioned in Section 2.2, cryptocurrency price processes are exhibiting jump dynamics. Furthermore, realized variance converges to the quadratic variation of the price process describing the asset being modeled. Thus, in the case of a jump diffusion process, realized volatility would, in theory, converge to the square root of the quadratic variation described above. Incorporating jump components in the modeling of realized volatility is therefore deemed important.

To model the jump component and in turn the realized volatility the HAR-J, given below, will be implemented:

$$RV_{t+1d}^{(d)} = c + \beta^{(d)}RV_t^{(d)} + \beta^{(w)}RV_t^{(w)} + \beta^{(m)}RV_t^{(m)} + \beta^{(J)}J_t + \omega_{t+1d}^{(d)} \tag{4.6}$$

where J_t is the jump compenent (Xu & Wang, 2014, p. 1303). The jump component is computed as

$$J_t(h) \equiv \max(RV_t(h) - BV_t(h), 0) \tag{4.7}$$

Where $RV_t(h)$ is the realized variance of a given day and $BV_t(h)$ is the realized bi-power variation of the day (Xu & Wang, 2014, p. 1303). The realized bi-power variation is given as

$$BV_t(h) \equiv \mu_1^{-2} \sum_{j=2}^{1/h} |r_{t+j*h,h}||r_{t+(j-1)*h,h}| \tag{4.8}$$

where μ_t is the expectation of the absolute value of a standard normal variable, $E(|Z|) = \sqrt{2/\pi}$ (Andersen et al., 2007, p. 4). The realized bi-power variation of a price process be shown to converge to the continuous-time integrated volatility

$$BV_t(h) \to \int_{t-1}^{t} \sigma^2(\tau) d\tau \tag{4.9}$$

as $h \to 0$, that is the sampling frequency increases to infinity (Andersen et al., 2007, p. 4). Thus, when combining the fact that the realized variance converges to the quadratic variation described above and the realized bi-power variation converges to the integrated variance we see that the jump component can be estimated by

$$RV_t(h) - BV_t(h) \to \sum_{t-1<\tau\leq t} \kappa^2(\tau) \tag{4.10}$$

The term $J_t(h)$ is truncated at zero, to ensure that negative values of the difference on the left hand side is not observed (Andersen et al., 2007, p. 5).

HAR-RS-I

Leverage effects are one of the stylized facts of volatilty. As described in Section 2.1.2 the term leverage effect refers to the fact that the volatility of a financial asset may be influence asymmetrically by negative and positive returns, respectively. Neither the HAR model nor the HAR-J model facilitate modeling this leverage effect in the modeling of realized volatility, and we have seen that cryptocurrencies may be influence greatly by the leverage effects, negative leverage effects, through herding behavior and "*pump and dump*" strategies. Thus, the last econometric model that will be implemented is the HAR-RS-I model, which allows for modeling the leverage effect Bollerslev et al. (2020). The HAR-RS-I model replaces the daily lagged realized volatility with realized up and down semi-volatility measures Qiu et al. (2019). Realized up and down semi-variance measures can be defined as

$$RS_t^+ = \sum_{j=1}^{M} r_{t,j}^2 * \mathbb{I}(r_{t,j} > 0) \tag{4.11}$$

$$RS_t^- = \sum_{j=1}^{M} r_{t,j}^2 * \mathbb{I}(r_{t,j} < 0) \tag{4.12}$$

such that the up semi-variance is composed as the realized variance of a period considering only the positive returns, and the down semi-variance considering only the negative returns Qiu et al. (2019). The up and down negative variances can be shown to converge to half the integrated variance plus the sum of squared positive and negative jumps, respectively Bollerslev et al. (2020):

$$RS_t^+ = \frac{1}{2}\int_{t-1}^{t} \sigma_s^2 ds + \sum_{t-1\leq\tau\leq t} J_\tau^2 \mathbb{I}_{(J_{tau}>0)} \tag{4.13}$$

$$RS_t^- = \frac{1}{2}\int_{t-1}^{t} \sigma_s^2 ds + \sum_{t-1\leq\tau\leq t} J_\tau^2 \mathbb{I}_{(J_{tau}<0)} \tag{4.14}$$

The sum of the up and down variances thus equals the realized variance, and therefore, replaces it in the HAR model. The final HAR-RS-I model is then:

$$RV_{t+1d}^{(d)} = c + \beta^{(-)} RS_t^{(-)} + \beta^{(+)} RS_t^{(+)} + \beta^{(w)} RV_t^{(w)} + \beta^{(m)} RV_t^{(m)} + \omega_{t+1d}^{(d)} \tag{4.15}$$

This section has described the econometric models that will be used as a benchmark to the deep learning models, described in next section.

4.1.2 Neural Networks

This section will describe the deep learning models used in the thesis. Overall two model architectures are used. The first is a simple feed-forward neural network and the second a dilated causal convolutional neural network. In total 8 distinct neural networks are trained and tested on the four cryptocurrencies. The difference of the neural networks within the two groups of model architectures is that a neural network corresponding to each of the HAR models is developed. Thus, a two neural networks with the data from the original HAR model as input, two for the lnHAR model, two for the HAR-J model and two for the HAR-RS-I model. Firstly, the model architecture of the simple artificial neural network (ANN) is described and secondly, the dilated causal convolutional neural network (DCCNN) architecture is described.

Feed-Forward Neural Network

All ANN models consist of 8 layers in the neural network. The activation function is the LeakyReLU, a type of ReLU activation function where values below 0 are not zeroed, but multiplied with a small value. Furthermore, the dropout regularization technique is used. Using dropout we introduce a probabilistic deactivation of neurons in the specific layer. That is with a probability of 0.3 a neuron is shut down during the forward call of the neural network. In this way we regularize the neural network as well as enabling the network to utilize larger learning rates. The layers of neural network can be seen below in table 4.1.

	Input Neurons	Output Neurons	Dropout Parameter
Layer 1 (Input)	3-5	8	0.0
Layer 2	8	16	0.0
Layer 3	16	32	0.0
Layer 4	32	64	0.0
Layer 5	64	32	0.3
Layer 6	32	16	0.3
Layer 7	16	8	0.0
Layer 8 (Output)	8	1	0.0

Table 4.1: BTC-HAR OLS Regression Results (own production).

The ANN models are trained using the Adam optimizer with a initial learning rate of 0.003, which is reduced at every 10 epoch by a factor of 10. Futhermore, the loss function that the ANN is minimizing is the MSE loss criterion.

Temporal Convolutional Network

The DCCNN models all consist of 5 causal convolutional blocks, which include a dropout parameter of 0.2. The input to all DCCNN models are the realized volatility time series with a look-back window of 32 days. Each consecutive causal convolutional layer has the dilation increased by a factor of 2, thus ending with a dilation of 16, and therefore, only two convolutions on the last layer. This way all inputs in the input of 32 time-steps are processed, but most importance is given to the inputs closest to the time t. Futhermore, a 1D convolutional layer with kernel size 1 is used to process extra inputs other than the realized volatility time series. Thus, the jump component J_t is processed using this 1D convolution, to enable the model to learn the best possible weight of the jump.

The HAR-RS-I-DCCNN model processes the input a bit differently than the rest of the models. Instead of taking the realized volatility time series as input, it takes two time series as input, the positive realized semi-variance and the negative realized semi-variance. The outputs of the two dilated causal convolutional blocks are then processed using linear feed-forward layers.

For all models the last step is two linear layers, with the first having 32 inputs, from the dilated causal convolutional blocks, and 32 outputs. This output is run through the ReLU activation function, and fed into a linear layer consisting of 32 inputs and a single output, the forecasted value. The DCCNN models are trained using the Adam optimizer with an initial learning rate of 0.03 decreasing by a magnitude of 10 every 10 epochs. The loss function to be minimized is the MSE loss criterion.

4.2 Evaluation

The central comparison metrics in the thesis are the evaluation criterions introduced in this section. An evaluation criterion describes the error of a prediction against a set of true values.

The choice of evaluation metric, is not arbitrary, as each metric is biased in some way.

The most well-known evaluation criterion is probably the Mean Squared Error (MSE):

$$\text{MSE: } L(\hat{\sigma}^2, h) = (\hat{\sigma}^2 - h)^2 \tag{4.16}$$

where $\hat{\sigma}^2$ is the volatility proxy, and h is the volatility proxy forecast (Patton, 2011, p. 248).

As a forecast comparison metric the MSE criterion has been shown to be robust in the sense that the ranking of two forecasts of conditional variance stays the same no matter whether a conditionally unbiased volatility proxy or the true conditional variance is used (Patton, 2011, p. 248). This robustness has only been shown for two evaluation criterions when comparing forecasts of volatility proxies, the second being the quasi-likelihood (QLIKE) criterion (Patton, 2011, p. 251). The QLIKE criterion is given as

$$\text{QLIKE: } L(\hat{\sigma}^2, h) = \log h + \frac{\hat{\sigma}^2}{h} \tag{4.17}$$

As can be seen from the equations above, the QLIKE and MSE criterions differ in their computation of the loss for each observation. This gives rise to differing characteristics of the metrics when used as evaluation metrics. The MSE loss is computed using the standard forecast error $\hat{\sigma}^2 - h$ and is especially sensitive to extreme observations in errors and the magnitude of the volatility proxy. QLIKE on the other hand is more susceptible to negative errors, where the forecasted volatility is higher than the true volatility (Patton, 2011, p. 251-252). The QLIKE and MSE metrics thus differ in their sensitivities and will yield two different perspectives on the evaluation of the forecasting models. The overall goal of the thesis is to evaluate whether or not the two different neural network architectures are better at forecasting the realized volatility of cryptocurrencies. Thus, MSE and QLIKE will be the only two criterions used, as they are shown to be the only robust loss functions when comparing volatility models.

Chapter 5 Results

The primary results of this thesis is the evaluation of the forecasting performance of HAR models compared to ordinary feed-forward neural networks and dilated causal convolutional neural networks. Model setup, architecture and setup have been described in the Methodology, chapter 4. Thus, this chapter will only be on the in-sample and forecasting evaluation. First, the models will be briefly evaluated in-sample. Evaluating neural networks ability to learn the structure of the training data is critical, as it gives us a tool for evaluating different architectures, criterions, optimizers, batch sizes, and training periods. The focus when doing so must be to get the model to learn, but not overfit to the training data. Various methods for regularizing neural networks are described in Chapter 2, Theory. Furthermore, depending on the structure and statistical properties of the data, the batch sizes, training periods, and criterions must be fine-tuned to achieve the best results. There are some general guidelines to how to choose these parameters, but given the complexity and flexibility of deep learning models, much of the performance is dependent on trial and error. The second section of the results will be devoted to evaluating and comparing the forecasting performance of the different models. Here, the forcasting performance will be evaluated using the evaluation criterions described in the methodology. Furthermore, a visual inspection of the forecasted time series of the best performing models and the true realized volatility time series will be conducted.

5.1 In-Sample Evaluation

The in-sample evaluation of the HAR models and the deep learning models will differ due to their characteristics. Where the HAR models are fitted using the closed form OLS and contains only few parameters, the neural networks are trained using stochastic gradient descent and contain a far greater number of parameters. Thus, the evaluation must differ. The in-sample evaluation of the HAR models will consider the importance of the parameters, and their performance in terms of the forecast evaluation criterion on the training data. The in-sample evaluation of the deep learning models will consider the minimization of the loss criterion, the number of parameters and the forecast evaluation criterion.

5.1.1 HAR Models

As the thesis is primarily concerned with comparing and evaluating forecasting performance, the in-sample evaluation of the HAR models will be focused around BTC. The results of the in-sample evaluation of the HAR models on the other cryptocurrencies are similar and thus omitted

in the main text.

The HAR models employed for forecasting the realized volatility of the cryptocurrency returns are the HAR, *ln*HAR, HAR-J, and HAR-RS-I, as presented in the methodology. The original HAR model, models the realized volatility as the sum of previous day realized volatility, $RV_t^{(d)}$, the averaged daily realized volatility over the past week $RV_t^{(w)}$, and the averaged daily realized volatility over the past month $RV_t^{(m)}$. The parameters $\beta^{(d)}$, $\beta^{(w)}$, $\beta^{(m)}$, and c are estimated using OLS and given below:

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.0056	0.002	3.682	0.000	0.003	0.009
Daily	0.5214	0.034	15.306	0.000	0.455	0.588
Weekly	0.2275	0.055	4.169	0.000	0.120	0.335
Monthly	0.0901	0.054	1.657	0.098	-0.017	0.197

Table 5.1: BTC-HAR OLS Regression Results (own production).

Since the thesis is focused on forecasting performance we are not removing any input variables, and thus, doing inference on the estimated model parameters, confidence intervals, and significance levels, should be done cautiously. Nevertheless, it can be seen that the past daily realized volatility $RV_t^{(d)}$ is the most significant and driving parameter, followed by the weekly $RV_t^{(w)}$. The autocorrelation function of the BTC realized volatility exhibited significant autocorrelation until a lag of approximately 30, however, the first seven to ten lags were by far the most significant. Thus, considering the autocorrelation function, the significance of the daily and weekly realized volatilities seems plausible.

The second HAR model is the log-transformed HAR model, *ln*HAR, where both the dependent and independent variables have been log transformed. Considering again the autocorrelation for the $log-RV$ of BTC in chapter 3, the log-transformed realized volatility, displayed a stronger autocorrelation for higher lags, thus it might be expected that $lnRV_t^{(m)}$ is more significant in the log-transformed HAR model. Seen below is the results of the OLS regression on the *ln*HAR model:

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	-0.4544	0.113	-4.037	0.000	-0.675	-0.234
Daily	0.4710	0.036	13.021	0.000	0.400	0.542
Weekly	0.3091	0.054	5.679	0.000	0.202	0.416
Monthly	0.0964	0.050	1.937	0.053	-0.001	0.194

Table 5.2: BTC-lnHAR OLS Regression Results (own production).

As expected, the monthly $lnRV_t^{(m)}$ is more significant in this model, although not by much. Again it is the daily and weekly realized volatility measures $RV_t^{(d)}$ and $RV_t^{(d)}$ that are most significant and with the highest influence.

The previous two HAR models, have modeled the realized volatility using only aggregate measures of the previous levels of the realized volatility. As explained previously, the price process of cryptocurrencies are susceptible to jumps, and that, these jumps are integral in estimating and modeling the volatility of the return process. By describing the return process as a continuous-time jump diffusion process, the HAR-J model have been made to account for the jump component in the process. Thus, other than the daily, weekly, and monthly aggregate measures of the previous level of realized volatilty, a jump compenent is included:

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.0004	0.000	2.347	0.019	6.52e-05	0.001
Daily	0.3054	0.037	8.211	0.000	0.232	0.378
Weekly	0.1310	0.063	2.073	0.038	0.007	0.255
Monthly	0.0667	0.081	0.819	0.413	-0.093	0.226
Jumps	2.2527	0.308	7.323	0.000	1.649	2.856

Table 5.3: BTC-HAR-J OLS Regression Results (own production).

The result of the OLS regression for the HAR-J model shown above, shows that the jump component is by far the most important parameter when it is present.

It is established in the theory section that the volatility of cryptocurrencies exhibit leverage effects. That is, the magnitude of the influence of negative returns on the volatility is larger than the influence from positive returns. Contrasting to the usual characteristics of the volatililty of return processes, cryptocurrencies have also been shown to exhibit negative leverage effects. To model this, the HAR-RS-I, as introduced in the methodology, is employed. The results of the OLS regression on the HAR-RS-I model for BTC is shown below.

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.0003	0.000	1.854	0.064	-1.52e-05	0.001
Weekly	0.1371	0.052	2.617	0.009	0.034	0.240
Monthly	0.1095	0.067	1.628	0.104	-0.022	0.241
RS+	-1.2976	0.082	-15.827	0.000	-1.459	-1.137
RS-	2.4372	0.094	25.907	0.000	2.253	2.622

Table 5.4: BTC-HAR-RS-I OLS Regression Results (own production).

As is seen in the parameter estimates above, the negative realized semi-variance has a positive influence on the volatility of the return process, while the positive realized semi-variance has a negative effect. The sign of the influence of the positive and negative semi-variances is consistent throughout all the cryptocurrencies used in this thesis (see Appendix A: In-Sample Regression Results).

It is noted in the introduction of this section, that the main goal of this section is not inference, but rather evaluating the regression results and in-sample predictions of the different models. As seen above in the OLS results of the models, the variable that is consistently the least significant is the monthly averaged daily realized volatility (variance), $RV_t^{(m)}$. The significant and influential variable is the daily realized volatility $RV_t^{(d)}$, and the jump variable J_t and realized semi-variances, $RS{+}_t^{(d)}$ and $RS{-}_t^{(d)}$.

Figure 5.1 depicts the predictions of the four HAR models on the in-sample data of BTC. Comparing to the true realized volatility in the BTC data, the HAR and lnHAR models seem to be quite similar. This is also expected, as the only difference between the two models is that the dependent and independent variables have been log-transformed in the lnHAR model. Comparing to the HAR-J and HAR-RS-I models, these models seem to be better able to capture the extreme levels of realized volatility. However, the HAR-J model consistently is not able to capture the low levels of realized volattility, while the HAR-RS-I model at times are more volatile than the true realized volatility.

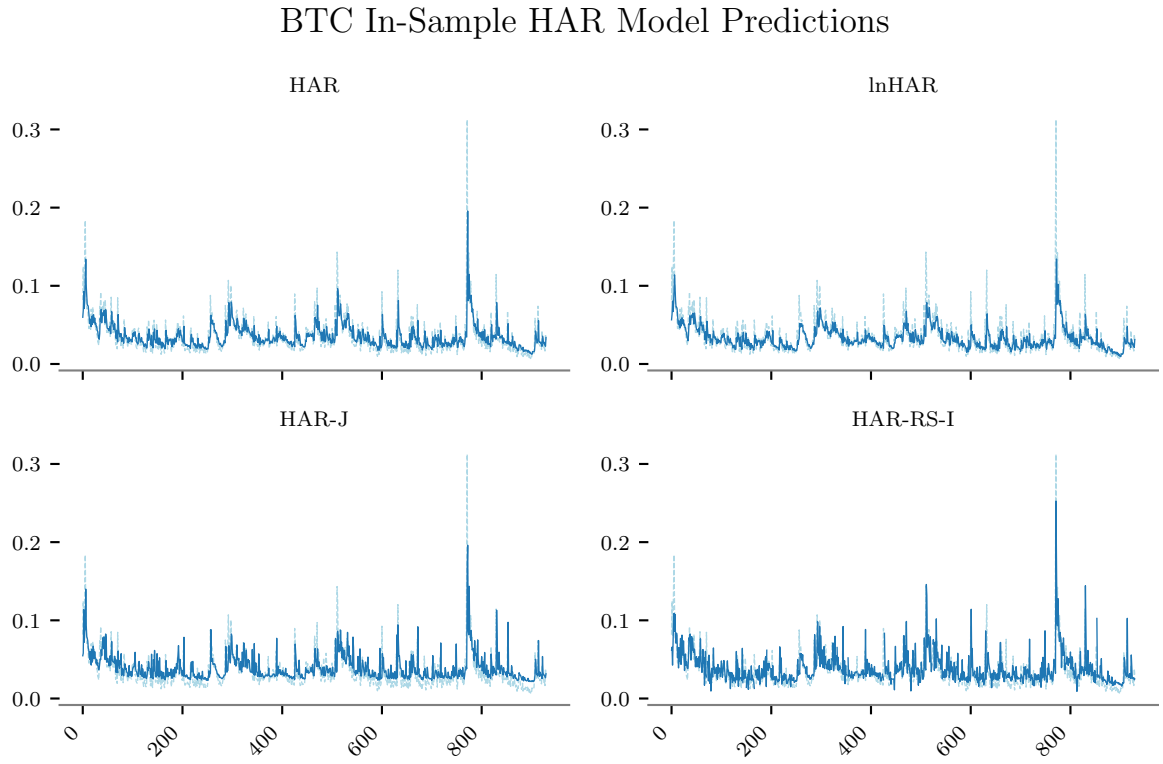

Figure 5.1: In-Sample predictions for the different HAR models on the BTC dataset. True realized volatility is the dashed line and predicted realized volatility is the solid line (own production).

The accuracy of the four models in terms of the two forecast evaluation criterions, QLIKE and MSE, is shown in the tables 5.5 and 5.6. It is seen that the in-sample performance of the HAR model is best on all cryptocurrencies.

index	HAR	lnHAR	HAR-J	HAR-RS-I
BTC	0.000268	0.000281	0.000331	0.000332
ETH	0.000365	0.000367	0.000406	0.000447
ADA	0.000446	0.000449	0.000489	0.000545
BNB	0.000473	0.000479	0.000573	0.000577

Table 5.5: MSE criterion in-sample results of HAR models (own production).

index	HAR	lnHAR	HAR-J	HAR-RS-I
BTC	-2.409417	-2.409417	-2.392299	-2.385255
ETH	-2.119608	-2.119608	-2.107122	-2.095617
ADA	-1.980645	-1.980645	-1.969535	-1.950203
BNB	-2.101092	-2.101092	-2.081021	-2.079821

Table 5.6: QLIKE criterion in-sample results of HAR models (own production).

This section introduced the in-sample results of the HAR models employed in the thesis. No conclusions can be made about the importance of the individual variables or the performance of the models, as the necessary statistical tests have not been employed. However, this is not the goal of the section, rather it is to get an overview of the in-sample performance and behavior of the models. The next section briefly covers the in-sample results from the deep learning models developed.

5.1.2 Deep Learning Models

The deep learning models developed in this thesis is described in the methodology. Architecturally there are two distinct models, the ANN, which consists of simple feed-forward layers, and the DCCNN, which consists of dilated convolutional layers and feed-forward layers. The ANN architecture is the simplest and most general architecture of neural networks, while the DCCNN architecture is developed for audio processing and employs convolutions to detect and use structures and patterns in the sequential data input. The DCCNN architecture allows the neural network to process long time-series without needing a vast number of parameters and thus, is, in thoery, able to process sequential information more efficiently. The number of parameters in the DCCNN models is therefore smaller than in the ANN models, as seen in Table 5.7.

	HAR	lnHAR	HAR-J	HAR-RS-I
ANN	5537	5537	5537	5545
DCCNN	1110	1110	1115	2194

Table 5.7: Number of trainable weights in the deep learning models (own production).

In general, the greater the number of parameters used in a neural network, the more complex a function the neural network is able to represent. This is proven by the Universal Approximation Theorem, described in the theory. However, as also noted in the theory section, some deep learning architectures have proved better at specific tasks than others. Recurrent neural networks and dilated convolutional neural networks, have proven superior in tasks involving sequential data, to the standard feed-forward neural network. Therefore, the dilated causal convolutional neural network is employed to model and forecast the realized volatility of cryptocurrencies. The models are trained on a batch size of 8 and for 20 epochs. Training the models takes around 1.14 minutes for the ANN models and 2.30 minutes for the DCCNN models. As stated there are general guidelines for developing efficient and accurate neural networks, but no rules. Therefore, a lot of the task in developing deep learning models is to fine-tune the network architecture and hyperparameters by trial and error. The neural networks in this thesis are the result of many iterations of fine-tuning the following parameters:

- Batch size.
- Optimizer.
- Regularization.
- Width of layers.
- Dropout parameters.
- Normalization.
- Learning rate.
- Loss criterion.
- Number of layers.
- Convolution parameters.
- Scaling method.

The final models employed is described in the methodology and the in-sample accuracy computed by the MSE and QLIKE criterions are displayed in Tables 5.8 and 5.9 for the ANN models and Tables 5.10 and 5.11 for the DCCNN models below.

index	HAR-ANN	lnHAR	HAR-J	HAR-RS-I
BTC	0.000439	0.000566	0.000376	0.000508
ETH	0.000606	0.000662	0.000525	0.000647
ADA	0.000827	0.001090	0.000713	0.000855
BNB	0.000667	0.000217	0.000627	0.000849

Table 5.8: MSE criterion in-sample results of ANN models (own production).

index	HAR	lnHAR	HAR-J	HAR-RS-I
BTC	-5.415063	-5.415063	-5.190651	-4.363010
ETH	-5.060013	-5.060013	-4.905804	-4.073563
ADA	-5.551744	-5.551744	-4.762834	-3.892110
BNB	-4.959763	-4.959763	-4.810081	-4.023506

Table 5.9: QLIKE criterion in-sample results of ANN models (own production).

The best ANN model in terms of MSE for BTC, ETH, and ADA is the one based on the HAR-J data, whereas the best for the BNB cryptocurrency is the lnHAR model. In general, however, all models, except the BNB-lnHAR model, perform worse than their standard econometric counterparts on the in-sample data with the MSE criterion. This is not the case for the QLIKE criterion, where the ANN models consistently beats their econometric counterparts. The important thing to note is that, in the number of parameters may allow the deep learning to overfit to the training data, and thus, the only truly important metrics are the forecasting performance, as this allows us to gauge the generalization ability of the deep learning models.

The DCCNN models in general perform better on the MSE criterion than the ANN models and worse on the QLIKE criterion. The best model for the MSE criterion is the model with HAR data input on BTC, ETH, and ADA, whereas the lnHAR is performining the best on the BNB cryptocurrency. For the QLIKE criterion the HAR-J-DCCNN model performs the best on all cryptocurrencies. It is again the case that the DCCNN models all outperform the standard econometric HAR model on the QLIKE criterion and is worse on the MSE criterion, on the in-sample data.

index	HAR	lnHAR	HAR-J	HAR-RS-I
BTC	0.000375	0.000399	0.005682	0.000382
ETH	0.000493	0.000503	0.000597	0.000519
ADA	0.000749	0.000772	0.000927	0.000802
BNB	0.000594	0.000171	0.000637	0.000560

Table 5.10: MSE criterion in-sample results of DCCNN models (own production).

index	HAR	lnHAR	HAR-J	HAR-RS-I
BTC	-4.663832	-4.663832	-2.928715	-4.765283
ETH	-4.284034	-4.284034	-3.714721	-4.343763
ADA	-4.044130	-4.044130	-3.632308	-4.121609
BNB	-4.293869	-4.293869	-3.820101	-4.049508

Table 5.11: QLIKE criterion in-sample results of DCCNN models (own production).

5.2 Forecasting Performance

This section will compare the forecasting performance of the econometric and deep learning models on the data left out for testing. The testing data is the same for both the neural networks and the econometric models. The only difference is that the econometric models are re-calibrated at each time-step, thus using an adaptive rolling window for retraining. This is not done on the neural networks as the evaluation time for all models combined would amount to approximately 320 hours.

5.2.1 Comparison of Evaluation Criterions

In Tables 5.12 and 5.13 the results of the one-step ahead forecast evaluations are shown for the MSE criterion and the QLIKE criterion, respectively. It is evident from the results that the standard econometric results perform better across the board on the MSE criterion. The best model for forecasting in when evaluated by the MSE criterion, is the HAR model for ADA and the lnHAR model for BTC, ETH, and BNB. These two models does not incorporate any additional regressors other than $RV_t^{(d)}$, $RV_t^{(w)}$ and $RV_t^{(m)}$. Thus, in the settings used here, the additional regressors does not seem to add further forecasting performance compared to the standard HAR model. This is also seen in the deep learning models, where the realized volatility for all cryptocurrencies, except BTC, is best predicted using the HAR-ANN model. BTC, however, shows that for the MSE criterion, the best deep learning model for forecasting the realized volatility is the HAR-RS-I-DCCNN model. This may be caused by the realized semi-variances, and thus leverage effects, have a larger influence on the BTC realized volatility. The HAR-RS-I-DCCNN model is the only model that incorporates the realized semi-variances over a longer timeframe (32 days), and thus, the influence of the leverage effects may not be immediate, and persist over time. In general, however, the standard econometric models performs the best on forecasting the realized volatility of all cryptocurrencies, when evaluated by the MSE criterion.

Index Model	ADA	BNB	BTC	ETH
HAR	**0.001396**	0.000874	0.000344	0.000657
HAR-ANN	0.002056	0.001151	0.000518	0.000955
HAR-DCCNN	0.002225	0.001581	0.000558	0.001173
HAR-J	0.001535	0.001065	0.000447	0.000736
HAR-J-ANN	0.002244	0.001328	0.000586	0.001039
HAR-J-DCCNN	0.002447	0.001416	0.011308	0.001142
HAR-RS-I	0.001685	0.000976	0.000408	0.000763
HAR-RS-I-ANN	0.002871	0.001966	0.000724	0.001303
HAR-RS-I-DCCNN	0.003624	0.001616	0.000461	0.001399
lnHAR	0.001407	**0.000866**	**0.000341**	**0.000651**
lnHAR-ANN	0.003487	0.002250	0.001034	0.001420
lnHAR-DCCNN	0.002263	0.001652	0.000595	0.001192

Table 5.12: MSE criterion results for all models on the four cryptocurrencies. Best model for each cryptocurrency is boldfaced (own production).

The superiority of the standard HAR models is not evident in the evaluation of forecasting performance using the QLIKE criterion though. Here the deep learning models consistently outperform the standard econometric HAR model, with the lnHAR-ANN model performing the best on all cryptocurrencies. As described in the methodology, the QLIKE criterion is especially sensitive to downside errors, that is when the forecasted volatility is higher than the true volatility. The MSE criterion, is more susceptible to extreme values however, and thus, it would seem that the standard HAR models are better able to capture the jumps and extreme levels of volatility of the cryptocurrencies. Referring to the stylized facts of volatility, the standard HAR model seem to be modeling the volatility clustering, fat tails and jumps in volatility, whereas the deep learning models are better at capturing the long-term tendencies and mean reversion in the realized volatility. This, of course, is speculation, as we would need to conduct extensive tests to be able to conclude anything.

Index Model	ADA	BNB	BTC	ETH
HAR	0.021539	0.016688	0.010722	0.013507
HAR-ANN	-4.350425	-4.660872	-5.226993	-4.797500
HAR-DCCNN	-3.804357	-4.037620	-4.311311	-4.077711
HAR-J	-1.697530	-1.857860	-2.252019	-1.993235
HAR-J-ANN	-4.141295	-4.609692	-4.896517	-4.616358
HAR-J-DCCNN	-3.346927	-3.547818	-2.423480	-3.509326
HAR-RS-I	-1.660325	-1.869390	-2.253699	-1.983087
HAR-RS-I-ANN	-4.261471	-4.327744	-4.684976	-4.314452
HAR-RS-I-DCCNN	-3.511995	-4.226131	-4.407700	-3.856696
lnHAR	-1.706124	-1.880472	-2.275557	-1.998397
lnHAR-ANN	**-5.513195**	**-5.458666**	**-5.994108**	**-5.176487**
lnHAR-DCCNN	-3.870313	-4.060311	-4.346369	-4.118026

Table 5.13: QLIKE criterion results for all models on the four cryptocurrencies. Best model for each cryptocurrency is boldfaced (own production).

5.2.2 Best Models

The best model for forecasting the realized volatility of the four cryptocurrencies in terms of the MSE criterion is the standard lnHAR model, and in terms of the QLIKE criterion, it is the lnHAR-ANN models. In figure 5.2 the predictions made from the best MSE model and the best QLIKE model is seen plotted against the true realized volatility.

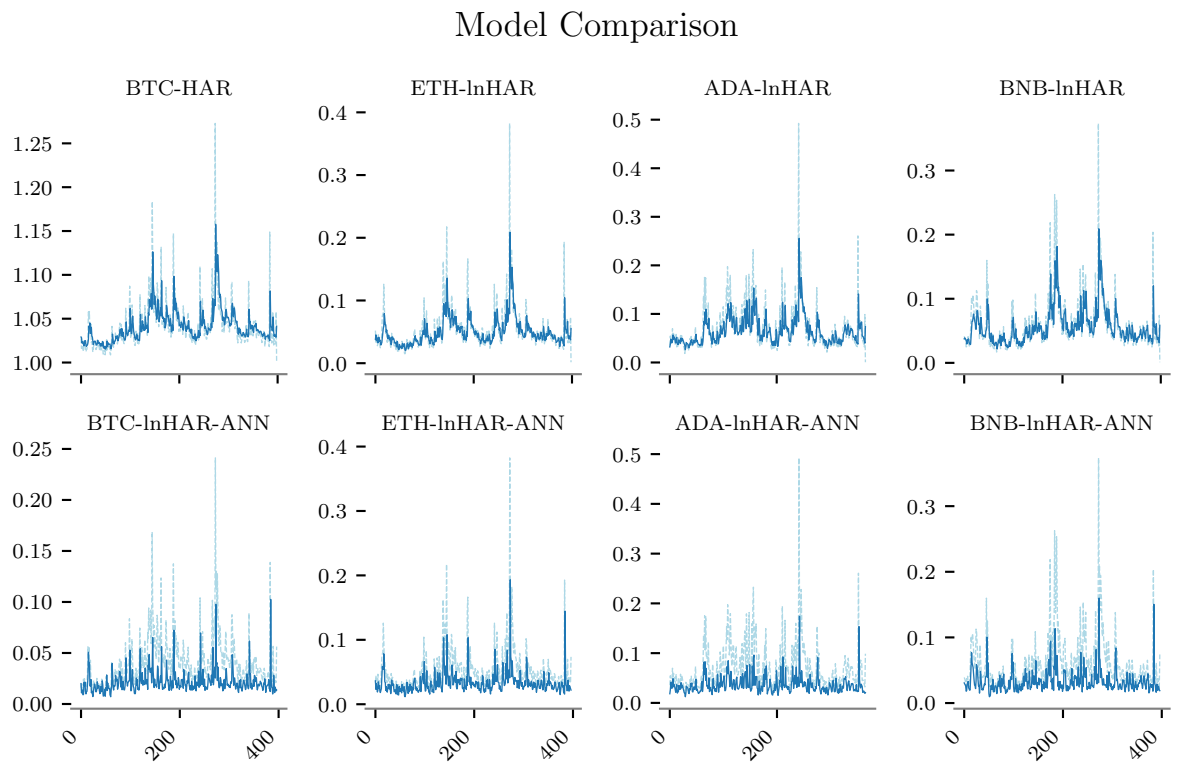

Figure 5.2: Best models for each cryptocurrency in terms of MSE and QLIKE. Top row is best model in terms of MSE and bottom in terms of QLIKE. Dashed line is true realized volatility and solid predicted (own production)

Judging from the plots above, the HAR models is better at capturing and following the day-to-day variation in the realized volatilty, while the deep learning models in general predicts a lower value of the realized volatility. Both models types are equally good at predicting the extreme values in the realized volatlity, although the tendency is that the forecasted value is too low. In general, the conclusion of the model experiments must be that the deep learning models seem better able to capture the long-term mean and stay on the lower side of the realized volatility, while the standard HAR models are better able at capturing the day-to-day variation of the realized volatility.

Chapter 6 Discussion

The discussion will attempt to provide answers, ideas and perspectives on the following questions:

- What interpretation could be made from the results?
- What direction could further research take?
- What might these results be used for?

6.1 Interpretation of Results

The results presented in Chapter 5 shows that the standard econometric HAR models perform best at forecasting the realized volatility of cryptocurrencies when evalauted on the MSE loss criterion. This might be evidence of the HAR models ability to model the extreme values and day-to-day fluctuations in the realized volatility. On the other hand, the ANN models showed better performance on forecasting the realized volatility of the cryptocurrencies when evaluated on the QLIKE loss criterion. Thus, the ANN models seem better a level of realized volatility that is not too large. In other, words this might show that the ANN models are better at modeling the latent strucure of the realized volatility of the cryptocurrencies.

The DCCNN models performed better on the MSE criterion than the ANN models, but worse than the HAR models. Further, they performed worse than the ANN models on the QLIKE criterion, but better than the HAR models. The mediocre performance of the DCCNN models might be due to the fact that they contained far less parameters compared to the ANN models, and thus were not able to sufficiently model the realized volatility of the cryptocurrencies. To enable the DCCNN models to perform better one might consider increasing the look-back window of the DCCNN models, thus enabling more information to be inputted to the model. This would automatically increase the number of parameters as a larger number of dilated causal convolutional layers would have to be used. Another direction to go would be increasing the number of linear layers used after the dilated causal convolutional blocks, to enable the neural network to further increase the model complexity of the model for the forecasted realized volatility. This would again increase the number of neural networks, and increase the number a lot more than with the extra dilated causal convolutional layers.

All in all, it is quite interesting that the ANN model seems better at modeling the realized volatility when evaluated by the QLIKE loss function, and especially why there is an difference between the HAR model and the ANN model. To better understand this difference, a direction of further research could be to analyse the residuals of the two models on the cryptocurrencies.

This would allow us to gain an understanding of where the models differ significantly in their predictions, whether it is in times of high volatility or low volatility, or when the level of volatility changes. Further, one could run the models on a larger set of data, to check if the difference persists in other asset classes with other volatility characteristics.

6.2 Usability of Results

The results and the interpretation described above could have multiple uses in the financial research or application. As the results show that the ANN model is best on the QLIKE criterion and the HAR model is best on the MSE criterion, one could argue that an ensemble model would perform more well rounded in a practical setting. In most areas of use the most important thing about the volatility models used are their ability to compute an unbiased and accurate forecast of the volatility. Furthermore, it is important that the volatility model can predict jumps and extreme levels of volatility as these are especially important in risk management and option pricing. An ensemble model would be a model which takes the output of both the ANN model and the HAR model and use both in a separate model, either by a weighted average or another regression. The ensemble model should then be able to take the best of both of the models and therefore generate a more well rounded, and unbiased, estimate of volatility.

Furthermore, the results might be used as a guidance towards choosing the right model type for modeling the realized volatility of cryptocurrencies and perhaps other assets. That is, practitioners may use the results to make an informed decision about which model to employ in their work. For example, if a pratitioner were most concerned with modeling the extreme levels of volatility, he/ she should go with the HAR model as it would perform best at that task, if considering the results of this thesis. On the other hand, if the practitioner were most concerned with not predicting a too large level of volatility, say for option pricing, he/ she should go with the ANN model.

The conclusion of the discussion must be that more research is needed to understand the differences between the performance of the two model types. Furthermore, one could experiment with adding more parameters to the DCCNN models to analyze if the DCCNN models are capped in their performance by the relatively low number of parameters. The usability of the results are twofold; firstly, the results may be used to motivate a development of an ensemble model to create a model balanced realized volatility forecast, and secondly, the results may be used a guide for choosing the right model type for the specific task at hand.

Chapter 7 Conclusion

This thesis set out to answer the research statement:

> *How can neural networks enhance the capability of HAR-style models in forecasting the realized volatility of cryptocurrencies?*

To accomplish this four HAR-style models were implemented as baseline models. The Heterogenous Autoregressive (HAR) model and the lnHAR models are the standard models used, which are developed based un the Heterogenous Market Hypothesis. The Heterogenous Market Hypothesis in general concerns itself with the differing trading horizons and frequencies of traders. This leads to the HAR model which includes an averaged daily, weekly and monthly component of the realized volatility. To allow better represent the stylized facts of volatility the two models, HAR-J and HAR-RS-I, are implemented as well. The HAR-J tries to model the realized volatility using both the realized volatility and a jump component in the regression model, thus attempting to include volatility jumps in the model. Similarly, the HAR-RS-I model includes the positive and negative realized semi-variances, to model the leverage effects seen in volatility.

The deep learning models used in the thesis are of two architectures. The first is the simple ANN model, consisting only of linear feed-forward layers. The second is a dilated causal convolutional neural network, inspired by the WaveNet algorithm devised by Google's Deep Mind. Both architectures are implemented in four ways, each according to one of the HAR models mentioned above.

The results of the thesis is that on forecasting the one-day ahead realized volatility of the cryptocurrencies the HAR model performs best when evaluated on teh MSE criterion, while the ANN model performs best on the QLIKE criterion. Thus, the econometric HAR models seem to be better at modeling the day-to-day movements in the volatility while the ANN models seem to be better at modeling the latent structure of the volatility. The DCCNN models perform worse than the ANN on QLIKE and better on MSE, while performing worse on the MSE and better on the QLIKE criterion than the HAR models.

More research is needed to better understand the differences between the model predictions of the realized volatility. This may be done either by a more in-depth scrutinization of the predictions of the models on the data in this thesis or by evaluating the models on other asset classes with different volatility characteristic.

The results of the thesis can possibly be used in two ways. Either, by using the as a guidance to model selection or by using the models in an ensemble fashion to achieve a more balanced forecast of realized volatility.

References

Aalborg, H. A., Molnár, P., & de Vries, J. E. (2019, June). What can explain the price, volatility and trading volume of Bitcoin? *Finance Research Letters*, *29*, 255–265. doi: 10.1016/j.frl.2018.08.010

Andersen, T. G. (n.d.). Realized Volatility;. *Federal Reserve Bank of Chicago*(No. 2008-14), 30.

Andersen, T. G., Bollerslev, T., & Diebold, F. X. (2007, November). Roughing It Up: Including Jump Components in the Measurement, Modeling and Forecasting of Return Volatility. *The Review of Economics and Statistics*, *4*(89), 19.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Labys, P. (2001, March). The Distribution of Realized Exchange Rate Volatility. *Journal of the American Statistical Association*, *96*(453), 42–55. doi: 10.1198/016214501750332965

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Labys, P. (2003). Modeling and forecasting realized volatility. *Econometrica : journal of the Econometric Society*, *71*(2), 579–625.

Baur, D. G., & Dimpfl, T. (2018, December). Asymmetric volatility in cryptocurrencies. *Economics Letters*, *173*, 148–151. doi: 10.1016/j.econlet.2018.10.008

Baur, D. G., Hong, K., & Lee, A. D. (2018, May). Bitcoin: Medium of exchange or speculative assets? *Journal of International Financial Markets, Institutions and Money*, *54*, 177–189. doi: 10.1016/j.intfin.2017.12.004

Binance Whitepaper. (2017). https://www.exodus.com/assets/docs/binance-coin-whitepaper.pdf.

Bishop, C. M. (2006). *Pattern recognition and machine learning.* New York: Springer.

Bollerslev, T., Li, S. Z., & Zhao, B. (2020, May). Good Volatility, Bad Volatility, and the Cross Section of Stock Returns. *Journal of Financial and Quantitative Analysis*, *55*(3), 751–781. doi: 10.1017/S0022109019000097

Buterin, V. (2013). Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. , 36.

Campbell, J. Y., Lo, A. W., & MacKinlay, A. C. (2012). *Econometrics of Financial Markets.* New Jersey: Princeton University Press.

Chaim, P., & Laurini, M. P. (2018, December). Volatility and return jumps in bitcoin. *Economics Letters*, *173*, 158–163. doi: 10.1016/j.econlet.2018.10.011

Christensen, K., Siggaard, M., & Veliyev, B. (2021, January). *A machine learning approach to volatility forecasting* (WorkingPaper No. 2021-03). Institut for Økonomi, Aarhus Universitet.

Corsi, F. (2005). *Measuring and Modelling Realized Volatility: From Tick-by-tick to Long Memory* (Unpublished doctoral dissertation). University of Lugano, Lugano, Switzerland.

Corsi, F. (2008, November). A Simple Approximate Long-Memory Model of Realized Volatility. *Journal of Financial Econometrics*, *7*(2), 174–196. doi: 10.1093/jjfinec/nbp001

Cryptocurrencies with Highest Market Cap - Yahoo Finance. (2022, July). https://finance.yahoo.com/u/yahoo-finance/watchlists/crypto-top-market-cap/.

Engle, R. F., & Patton, A. J. (2000, October). What good is a volatility model? *Quantitative Finance*, *1*, 9.

Fama, E. F. (1970, May). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2), 383. doi: 10.2307/2325486

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* Cambridge, Massachusetts: The MIT Press.

Hornik, K., Stinchcombe, M., & White, H. (1989, January). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. doi: 10.1016/0893-6080(89)90020-8

Hoskinson, C. (2015). *Motivation.* https://why.cardano.org/en/introduction/motivation/.

Hull, J. (2015). *Options, Futures, and Other Derivatives* (Ninth edition ed.). Boston: Pearson.

Jensen, M. C. (1978, June). Some anomalous evidence regarding market efficiency. *Journal of Financial Economics*, *6*(2-3), 95–101. doi: 10.1016/0304-405X(78)90025-9

Kingma, D. P., & Ba, J. (2017, January). *Adam: A Method for Stochastic Optimization* (No. arXiv:1412.6980).

Kristoufek, L. (2015, April). What are the main drivers of the Bitcoin price? Evidence from wavelet coherence analysis. *PLOS ONE*, *10*(4), e0123923. doi: 10.1371/journal .pone.0123923

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc.

LeCun, Y., Bengio, Y., & Hinton, G. (2015, May). Deep learning. *Nature*, *521*(7553), 436–444. doi: 10.1038/nature14539

Lim, B., & Zohren, S. (2021, April). Time Series Forecasting With Deep Learning: A Survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194), 20200209. doi: 10.1098/rsta.2020.0209

Liu, L. Y., Patton, A. J., & Sheppard, K. (2015, July). Does anything beat 5-minute RV? A comparison of realized measures across multiple asset classes. *Journal of Econometrics*, *187*(1), 293–311. doi: 10.1016/j.jeconom.2015.02.008

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms* (7.2 ed.). Cambridge University Press.

Mankiw, N. G. (2009). *Macroeconomics* (7th ed ed.). New York, NY: Worth Publishers.

Müller, U. A., Dacorogna, M. M., Rakhal, D., V. Pictet, O., Olsen, R. B., & Ward, J. R. (1993, August). *Fractals and Intrinsic Time - a Challenge to Econometricians* (Working Papers). Olsen and Associates.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. , 9.

Patton, A. J. (2011, January). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, *160*(1), 246–256. doi: 10.1016/j.jeconom.2010.03 .034

Poon, S.-H., & Granger, C. W. J. (2003, June). Forecasting Volatility in Financial Markets: A Review. *Journal of Economic Literature*, *41*(2), 478–539. doi: 10.1257/ jel.41.2.478

Qiu, Y., Zhang, X., Xie, T., & Zhao, S. (2019, March). Versatile HAR model for realized volatility: A least square model averaging perspective. *Journal of Management Science and Engineering*, *4*(1), 55–73. doi: 10.1016/j.jmse.2019.03.003

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2019, November). *Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019* (No. arXiv:1911.13288).

Takaishi, T. (2018, September). Statistical properties and multifractality of Bitcoin. *Physica A: Statistical Mechanics and its Applications*, *506*, 507–519. doi: 10.1016/j.physa.2018.04.046

Timmermann, A., & Granger, C. W. (2004, January). Efficient market hypothesis and forecasting. *International Journal of Forecasting*, *20*(1), 15–27. doi: 10.1016/S0169-2070(03)00012-8

Tsay, R. S. (2005). *Analysis of Financial Time Series* (Second ed.). Wiley.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016, September). *WaveNet: A Generative Model for Raw Audio* (No. arXiv:1609.03499).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017, December). *Attention Is All You Need* (No. arXiv:1706.03762).

Wang, Z., & Oates, T. (n.d.). Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks. , 8.

Xu, J., & Wang, S.-s. (2014, August). HAR volatility modelling with jump. In *2014 International Conference on Management Science & Engineering 21th Annual Conference Proceedings* (pp. 1301–1306). Helsinki, Finland: IEEE. doi: 10.1109/ICMSE.2014.6930380

Yermack, D. (2015). Chapter 2 - is bitcoin a real currency? An economic appraisal. In D. Lee Kuo Chuen (Ed.), *Handbook of digital currency* (pp. 31–43). San Diego: Academic Press. doi: 10.1016/B978-0-12-802117-0.00002-3

Zhang, W., Wang, P., Li, X., & Shen, D. (2018, November). Some stylized facts of the cryptocurrency market. *Applied Economics*, *50*(55), 5950–5965. doi: 10.1080/00036846.2018.1488076

Appendix A OLS Regression Results for HAR Models

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0056	0.002	3.682	0.000	0.003	0.009
Daily	0.5214	0.034	15.306	0.000	0.455	0.588
Weekly	0.2275	0.055	4.169	0.000	0.120	0.335
Monthly	0.0901	0.054	1.657	0.098	-0.017	0.197

Table A.1: BTC-HAR OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0081	0.002	3.721	0.000	0.004	0.012
Daily	0.5239	0.034	15.488	0.000	0.458	0.590
Weekly	0.1867	0.055	3.394	0.001	0.079	0.295
Monthly	0.1135	0.059	1.914	0.056	-0.003	0.230

Table A.2: ETH-HAR OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0130	0.003	3.991	0.000	0.007	0.019
Daily	0.5375	0.035	15.569	0.000	0.470	0.605
Weekly	0.1535	0.058	2.667	0.008	0.041	0.267
Monthly	0.0628	0.074	0.853	0.394	-0.082	0.207

Table A.3: ADA-HAR OLS Regression Results (own production).

Appendix A. OLS Regression Results for HAR Models

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0071	0.002	3.650	0.000	0.003	0.011
Daily	0.4683	0.034	13.656	0.000	0.401	0.536
Weekly	0.1476	0.059	2.502	0.013	0.032	0.263
Monthly	0.2287	0.052	4.384	0.000	0.126	0.331

Table A.4: BNB-HAR OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	-0.4544	0.113	-4.037	0.000	-0.675	-0.234
Daily	0.4710	0.036	13.021	0.000	0.400	0.542
Weekly	0.3091	0.054	5.679	0.000	0.202	0.416
Monthly	0.0964	0.050	1.937	0.053	-0.001	0.194

Table A.5: BTC-lnHAR OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	-0.4419	0.110	-4.021	0.000	-0.658	-0.226
Daily	0.5753	0.034	16.818	0.000	0.508	0.642
Weekly	0.1813	0.051	3.565	0.000	0.081	0.281
Monthly	0.1096	0.049	2.216	0.027	0.013	0.207

Table A.6: ETH-lnHAR OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	-0.6295	0.144	-4.385	0.000	-0.911	-0.348
Daily	0.5907	0.034	17.223	0.000	0.523	0.658
Weekly	0.1222	0.052	2.338	0.020	0.020	0.225
Monthly	0.0836	0.061	1.372	0.170	-0.036	0.203

Table A.7: ADA-lnHAR OLS Regression Results (own production).

Appendix A. OLS Regression Results for HAR Models

	coef	std err	t	P> |t|	[0.025	0.975]
const	-0.4184	0.096	-4.358	0.000	-0.607	-0.230
Daily	0.5183	0.034	15.038	0.000	0.451	0.586
Weekly	0.1989	0.054	3.675	0.000	0.093	0.305
Monthly	0.1577	0.048	3.256	0.001	0.063	0.253

Table A.8: BNB-lnHAR OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0004	0.000	2.347	0.019	6.52e-05	0.001
Daily	0.3054	0.037	8.211	0.000	0.232	0.378
Weekly	0.1310	0.063	2.073	0.038	0.007	0.255
Monthly	0.0667	0.081	0.819	0.413	-0.093	0.226
Jumps	2.2527	0.308	7.323	0.000	1.649	2.856

Table A.9: BTC-HAR-J OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0008	0.000	3.399	0.001	0.000	0.001
Daily	0.3981	0.037	10.871	0.000	0.326	0.470
Weekly	0.1364	0.066	2.058	0.040	0.006	0.266
Monthly	0.1405	0.085	1.657	0.098	-0.026	0.307
Jumps	0.2659	0.282	0.944	0.345	-0.287	0.819

Table A.10: ETH-HAR-J OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0016	0.000	4.277	0.000	0.001	0.002
Daily	0.4441	0.043	10.350	0.000	0.360	0.528
Weekly	0.1832	0.071	2.595	0.010	0.045	0.322
Monthly	0.0329	0.105	0.313	0.754	-0.173	0.239
Jumps	-1.2629	0.376	-3.362	0.001	-2.000	-0.526

Table A.11: ADA-HAR-J OLS Regression Results (own production).

Appendix A. OLS Regression Results for HAR Models

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0008	0.000	2.955	0.003	0.000	0.001
Daily	0.2000	0.038	5.220	0.000	0.125	0.275
Weekly	0.0186	0.069	0.271	0.786	-0.116	0.154
Monthly	0.2272	0.060	3.767	0.000	0.109	0.346
Jumps	3.8413	0.517	7.424	0.000	2.826	4.857

Table A.12: BNB-HAR-J OLS Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0003	0.000	1.854	0.064	-1.52e-05	0.001
Weekly	0.1371	0.052	2.617	0.009	0.034	0.240
Monthly	0.1095	0.067	1.628	0.104	-0.022	0.241
RS+	-1.2976	0.082	-15.827	0.000	-1.459	-1.137
RS-	2.4372	0.094	25.907	0.000	2.253	2.622

Table A.13: BTC-HAR-RS-I Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0004	0.000	1.883	0.060	-1.75e-05	0.001
Weekly	0.1299	0.059	2.216	0.027	0.015	0.245
Monthly	0.1989	0.075	2.652	0.008	0.052	0.346
RS+	-0.8742	0.085	-10.245	0.000	-1.042	-0.707
RS-	1.8377	0.094	19.614	0.000	1.654	2.022

Table A.14: ETH-HAR-RS-I Regression Results (own production).

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0011	0.000	3.363	0.001	0.000	0.002
Weekly	0.1564	0.063	2.492	0.013	0.033	0.280
Monthly	0.0607	0.093	0.649	0.517	-0.123	0.244
RS+	-1.3680	0.116	-11.776	0.000	-1.596	-1.140
RS-	2.2589	0.126	17.920	0.000	2.011	2.506

Table A.15: ADA-HAR-RS-I Regression Results (own production).

Appendix A. OLS Regression Results for HAR Models

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.0002	0.000	0.789	0.430	-0.000	0.001
Weekly	-0.0270	0.060	-0.451	0.652	-0.144	0.090
Monthly	0.3669	0.052	7.090	0.000	0.265	0.468
RS+	-1.7607	0.113	-15.619	0.000	-1.982	-1.539
RS-	2.8298	0.133	21.297	0.000	2.569	3.091

Table A.16: BNB-HAR-RS-I Regression Results (own production).