CHARLES UNIVERSITY

FACULTY OF SOCIAL SCIENCES

Institute of Economic Studies

Master's thesis

2024 **Jan Cvrček**

CHARLES UNIVERSITY

FACULTY OF SOCIAL SCIENCES

Institute of Economic Studies

Jan Cvrček

Can Image-Based Convolutional Neural Networks Forecast Volatility?

Master's thesis

Prague 2024

Author: Bc. Jan Cvrček

Supervisor: doc. PhDr. Jozef Baruník Ph.D.

Study programme: Ekonomie a Finance

Academic year: 2023/2024

Abstract

The thesis aims to investigate whether image-based convolutional neural networks (CNN) can help predict volatility of financial markets. Unlike any other academic work, it converts price and volume indicators of the index E-Mini S&P 500 from 2010 to 2019 into pictures and strives to forecast one day ahead level of realised volatility. The results suggest that this method can produce reasonable outcomes, but lacks in accuracy compared to benchmark standard models used in the literature, probably due to their autoregressive feature. Apart from this finding, the best length of the input days in the CNN specification appears to be one month, followed by a week and quarter, which achieve similar results. These conclusions were also subject to a robustness check, which, however, did not generate any contradictory evidence.

Keywords

Volatility, Stock Market, CNN, Deep Learning, Image Classification

Abstrakt

Cílem práce je zjistit, zda konvoluční neuronové sítě (KNS) mohou pomoci predikovat volatilitu finančních trhů. Jako dosud první akademická práce převádí cenové a objemové ukazatele indexu E-Mini S&P 500 z let 2010 až 2019 na obrázky a snaží se předpovědět úroveň realizované volatility na jeden den dopředu. Výsledky naznačují, že tato metoda generuje rozumné predikce, ale ve srovnání s referenčními standardními modely používanými v literatuře není tak přesná, pravděpodobně kvůli jejich autoregresivní vlastnosti. Kromě tohoto zjištění se jako nejlepší délka vstupních dnů ve specifikaci KNS jeví jeden měsíc, následovaný týdnem a čtvrtletím, které dosahují podobných hodnot. Tyto závěry byly rovněž podrobeny testu robustnosti, který však nepřinesl žádné protichůdné důkazy.

Klíčová slova

Volatilita, Akciový Trh, KNS, Hluboké učení, Klasifikace Obrázků

Declaration of Authorship

I hereby proclaim that I wrote my master thesis on my own under the leadership of my supervisor and that the references include all resources and literature I have used.

I grant a permission to reproduce and to distribute copies of this thesis document in whole or in part.

Prague, July 31, 2024

Jan Cvrček

Acknowledgments

I am deeply thankful to my supervisor doc. PhDr. Jozef Baruník Ph.D. for the guidance of my thesis. His brisk and complex reactions to my questions have made the process a valuable path, through which I have acquired a lot of knowledge. Nevertheless, I would also like to express the sincere gratitude to my whole family, colleagues and the IES in general for creating such a motivational environment during the studies.

Bibliographic note

Cvrček, Jan. *Can Image-Based Convolutional Neural Networks Forecast Volatility?*. Prague, 2024. (80 pp). Master's thesis (Mgr.), Charles University, Faculty of Social Sciences, Institute of Economic Studies. Thesis Supervisor doc. PhDr. Jozef Baruník Ph.D.

Table of Contents

List of Acronyms

AAPL	Apple Inc.
AdamW	Adaptive Moment Estimation with Weight Decay
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARFIMA	Autoregressive Fractionally Integrated Moving Average
ARSV	Autoregressive Stochastic Volatility
ARCH	Autoregressive Conditional Heteroskedasticity
ARVOL	Average Realized Volatility
BSM	Black-Scholes-Merton
CAPM	Capital Asset Pricing Model
CNN	Convolutional Neural Network
CTC	Close to Close
DL	Deep Learning
E-I-GJR-R-GARCH	Exponential Inverse Generalized Jackknife Residual-Realized GARCH
EMH	Efficient Market Hypothesis
EWMA	Exponentially Weighted Moving Average
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
HAR	Heterogeneous Autoregressive
IES	Institute of Economic Studies
LF	Loss Function
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NN	Neural Network
ReLU	Rectified Linear Unit

RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SMA	Simple Moving Average
SGD	Stochastic Gradient Descent
SV	Stochastic Volatility
SVM	Support Vector Machine

List of Figures

List of Tables

1 Introduction

Throughout the latest history, researchers have striven to describe and predict characteristics of time-series in many possible ways. In finance, the most important variables are return and volatility. The former can be directly observed from the data and is self-explanatory, while the latter requires some implicit measures to be applied. Thus, it has been the focal point of interest for decades. More importantly, the key question has always been how to predict this variable. Although the literature has brought a significant development over the years, the predictability has not ceased to be a struggle. We have witnessed a major growth in storing big data, followed by the computational availability of devices in recent years. In turn, many algorithms, which were time-consuming in the past, have become accessible. These include deep learning methods, encompassing the series of neural networks family and most importantly convolutional neural networks (CNN).

The science has broadly utilised them in other fields like facial and text-recognition, autonomous vehicles etc. Their main advantage is the ability to unfold the deep non-linearities, which these independent and identically distributed data exhibit. Nevertheless, the key question is whether this machine learning (ML) approach can help predict volatility, which resembles a temporal dependence. Since the CNN phenomenon is still at its very beginning, there do not exist many papers using this method in finance at all.

One of the very few is an article from Jiang et al. (2020), who successfully aimed to predict the future returns of a stock by training a CNN on images of the price indicators. Managing images instead of direct price values might be beneficial for the network since it may resemble the human brain, which operates with graphs in a better way than with plain value vectors. Also, it effectively captures structural dependence. This thesis strives to borrow this very recent idea and unlike any other academic work aims to ask a question whether CNN can help predict the volatility.

In the history academicians have first neglected the influence of volatility, then assumed its invariability over the horizon and finally paid more and more attention to it. The crucial point was the inability to observe the behaviour on the market directly. Thus, the initial proxy of taking the difference between the low and high levels of daily price emerged. The pioneers in the attempt to predict the value can be considered models of Parkinson (1980) or Garman and Klass (1980), who aimed to forecast this indicator for the next day. These models had a few caveats, like a potential bias by outliers. Therefore, academicians have come up with estimating the volatility as a latent variable using i.e., generalized autoregressive conditional heteroskedasticity models (GARCH). This setup enabled modelling the persistence and dynamics over time.

Nevertheless, since the technological growth has facilitated storing big data, it led to the emergence of high-frequency data, which contains denser information because it includes the whole daily behaviour consisting of many (usually 1 to 10-minute) intervals. This data could be used by models like heterogenous autoregressive (HAR) or other conventional methods that the literature calls them nowadays. Nonetheless, the evolution of highly functioning computers also enabled the use of the machine learning methods, which either connect the conventional methods (ARIMA with ML) or simply build on ML only (LTSM). The CNN, representing the ML group, thus provides us with an out of the box attitude of predicting volatility.

The first hypothesis is that this method produces reasonable values in forecasting volatility and shows relation patterns between the input and output data. The second one claims that using more days on the input improves the forecast precision since volatility behaves in clusters. The last one states that the image-based CNN does overperform the traditional methods in forecasting volatility.

The structure of the thesis is organised in the following way. First, some theoretical background with a bit of history to the topic is introduced. Then, the main data are presented, followed by a detailed description of the CNN methodology. Later, we describe and discuss the results, evaluate the three hypotheses mentioned above and perform a robustness check. Finally, conclusion with further research suggestions are situated at the end of the thesis.

2 Current State of Knowledge

2.1 The Concept of Volatility

Volatility is a key concept in financial analysis and econometrics, which refers to a price fluctuation of an asset in a specific time period. It is deemed to be an indicator of uncertainty and risk on the market. Higher value suggests that the price is expected to move both up and down to a greater extent. This makes the asset more attractive to risk-lovers than to risk-averse individuals. Therefore, volatility is essential for traders and investors to correctly assess the market environment and choose the most appropriate investment strategies.

2.2 Brief History

William Sharpe is considered to be the pioneer of examining the role of volatility in financial markets. His work dates to 1960s and is mostly famous for introducing the Capital Asset Pricing Model (CAPM) to the public community. Volatility was seen here as the ultimate level of uncertainty and risk, which gives a broader context to the expected return. Sharpe claimed that investors require a higher rate of return of an asset for a higher level of its volatility since they face a higher risk. This behaviour should also hold for the possibility of short selling. Moreover, Ross (1977) in this article even suggests that the restriction of short sale causes the market to be inefficient and thereby distorts the concept of CAPM.

A few years later in the 1960s, Fama and Samuelson contributed to the existing literature with a so-called Efficient Market Hypothesis (EMH). This concept comes in three variations. The first one states that all historical prices are reflected in the current market price of an asset. The second one believes that also the public information is mirrored on the market, while the third one says that even the private information is included. This makes the market highly efficient and thus impossible for the investors to find undervalued stocks and make profit. In regard to volatility, the EMH implies the same unattainability to predict its values in the future. Its significance is thus viewed as a continuous reflection of speed and volume of the new information in the price.

Another usage of volatility was in option pricing. Price variability was sight even in the beginning of the 20th century, but no model was introduced in that time. The cornerstone, however, traces back to the year 1973, when a trio of American economists came up with a Black-Scholes-Merton (BSM) model used for option pricing. This was a milestone since they were the first to assume volatility might explain the option price, together with several other factors. It was deemed constant, which may seem a little odd nowadays, yet they still received a Nobel Price for the model later. From this period ahead, practitioners have started to pay attention to the crucial role of volatility and have striven to base on that better and better techniques to outperform the market.

The subsequent epoch was characterised by the effort to improve the models because a non-negligible amount of people saw the opportunity to monetise it. This view of volatility changed in October 1987 when a big stock market crash took place. Researchers see the roots among others in the rising volatility, mainly dominant in the era right before the crisis. The effects on the stock market were large scale, which made the academic community reconsider the assumption of constant volatility.

Volatility was thus becoming broadly perceived as time varying. Moreover, it was brought to the attention of the academic community that prices exhibited larger continuous oscillation (e.g., before 1987) in some periods than in other time intervals. This created the idea of volatility clustering. Researchers subsequently developed stochastic models, which allowed volatility to be a dynamic process with clustering characteristics and persistence over time. Most common examples would be an ARCH model or Heston model, presented by R. Engle and L. Heston, respectively. These models later experienced their extensions, for instance the inclusion of volatility jumps, as a reaction to the real-world events. Those techniques are now considered traditional or conventional methods of estimating volatility and will be deeply examined in the following sub-chapters.

Thanks to the rising computational power and desire to acquire money, unconventional models have been introduced to help predict volatility. These include obtaining realised measures from high-frequency data on the one hand and machine learning approaches like deep neural networks, random forests, and many more on the other. These aim to capture latent data patterns by facilitating non-linearities in the process. Again, this topic will be spoken about in detail later on.

2.3 Volatility Measurement Methods

One of the reasons, volatility has been thoroughly studied is that it is an implicit measure. This means that unlike prices, we are not able to directly observe this variable on the market but can create various proxies based on the given data. A very basic, yet accepted, definition of volatility is standard deviation, which is a squared root of variance. Unfortunately, it was not possible to use this statistic in the past when only a single value per day was captured.

Over time, data analysts started to gather more information about the market behaviour. They also monitored the opening, closing, high and low levels of price for each day. This enabled to create a better picture of the market environment. The issue with not enough data points, however, was still prevailing. Four points may already seem to be a satisfactory number, but it may still provide a very biased estimate of the true standard deviation (Figlewski (1997)). Therefore, researchers began presenting their own proxies.

One of them was Parkinson (1980), who described the extreme value method in his work. He claimed that these outliers convey significant information about the variance of the rate of return and thus concentrated on the frequency of their occurrence and their magnitude. Moreover, he pointed out this approach is suitable for capturing heteroscedasticity and clustering of volatility.

With a brisk reaction to this article came Garman and Klass (1980), who extended his work by allowing more variables in the estimation. Unlike Parkinson (1980), their method incorporates the whole price path and better addresses the process dynamics. This technique was, however, also subject to criticism. Firstly, some scientists claim that it is very sensitive to significant price gaps, usually occurring between the end of the trading period and the beginning of the upcoming one. These unbalances are prone to distort the estimating process.

Secondly, the new Garman-Klass method assumes prices follow log-normal distribution. Such an assumption is not valid all the time and thus opens the discussion of the estimation's reliability. Overall, the Garman-Klass procedure brought valuable insights into better understanding the meaning of volatility. They also introduced another method purely based on the closing price called CTC, which has a much higher noise but may be useful for long-term estimation.

All the approaches presented so-far supposed the mean value was held constant. Once it changed, the volatility level was under or over-estimated. Therefore, Rogers and Satchell (1991) decided to propose a model that would capture this hurdle. It does not substantially differ from the Garman-Klass strategy, but it involves the non-zero drift to be present. The other issues, like log-normal distribution assumption and discrete sampling have remained untouched.

Nonetheless, the mostly spread measure before high-frequency data were stored, was a simple squared return for each assumed interval. This could be a day, week, or even larger period. The main advantages lie in quick data manipulation and putting emphasis on significant price movements. On the contrary, there exist several drawbacks. Patton (2006), for example, finds that squaring the returns does not capture the asymmetric effect of positive and negative changes enough. Further, Barndorff-Nielsen and Shephard (2006) arrive at a conclusion that sharp jumps are not factored in either.

To address the former problem, we may lean on using absolute returns. As Davidian and Carroll (1987) conclude, this procedure better accounts for asymmetry. Regarding accuracy, the academia remains rather inconclusive. For instance, McKenzie (1999) finalises her work in favour of absolute returns, while Bandi and Russell (2006) infer that squaring appears to be more efficient.

These definitions were put in the shadow when computers experienced huge improvement of their performance. Practitioners were able to collect more information about assets, which led to the emergence of what we call now high-frequency data. Andersen et al. (2001) published an article where they introduced the concept of realised variance. This method consists in summing up all the squared returns based on sampling frequency per one trading session. Realised volatility would then be a squared root of this number.

Of course, many questions arise as well. Firstly, what is the optimal sampling frequency? Corsi (2009) states that increasing the number of observations reduces bias and levels up precision. On the other hand, Ait-Sahalia et al. (2005) find out in their paper that enlarging the frequency, amplifies a so-called microstructure noise. Hence, there appears to be a trade-off between bias and noise lessening. Hautsch and Podolskij (2013) come to conclusion that every market is different and requires a specific choice of the sampling frequency, which means there is no universal correct decision. The academia, however, mostly operates with 1-minute or 5-minute intervals for stocks.

Secondly, there is a possibility to operate with realised variance, volatility, or log-volatility. Researchers mostly incline to prefer realised volatility over variance for its simplicity and mainly interpretability. But we can see an increasing number of articles in recent years studying realised log-volatility. This is because it features great statistical advantages like symmetry of distribution or additive properties (Barndorff-Nielsen and Shephard (2004)). Again, there is no ultimate rule, which one to use, and every academic work is free to choose its preferred candidate in line with the scope of their work.

In this thesis, we base our hypotheses on realised volatility since the goal is to compare the performance of the CNN method against the literature, which heavily prioritises this method. The objective is to compare the forecast models and using the widely utilised definition facilitates us a more reliable comparison. In the robustness check, we are operating with simple squared returns, which are also substantially employed in academia and provide us with comparable, yet distinct values to their high-frequency neighbours.

2.4 Traditional Models for Forecasting Volatility

So-far we have covered the role and measurement of volatility. In this subchapter, we delve into forecasting. There are theoretically endless possibilities of how far from now one can forecast. Nevertheless, it may be very challenging to use long horizons especially in financial markets since there is plethora of factors, which influence the outcome (Campbell et al. (1998)). Researchers for this reason incline to forecast up to one or a few periods ahead.

Generally, increasing the sampling frequency is said to enhance the forecast quality (Andersen et al. (2001)), but having too many observations can lead to facing substantive microstructure noise (Ait-Sahalia et al. (2005)). In contrast, if one aims to concentrate on long horizon forecasts (months, years), lower sampling frequency may prove useful (Alford and Boatsman (1995)). A paper from Meddahi and Renault (2004) then concludes that every asset might exhibit distinct volatility patterns, which may even vary in time. In this thesis, we focus on one day ahead forecasting, which means we engage with rather high-frequency data. Namely, we use a 5-minute frequency.

Throughout the history, scientists have created many models, which strive to forecast volatility. We first pay attention to the traditional ones and then move to their younger counterparts. In this subset, we can further distinguish those into multivariate and univariate methods. The literature provides mixed findings on the accuracy comparison of those two. Having more independent variables may cover specific spillover effects (Bollerslev (2008)), but the results appear to be too sensitive to the modelling assumptions (Patton (2011)).

This thesis, however, only operates with historical stock prices, which means we will mainly focus on univariate analysis. Poon and Granger (2003) classify this set into five categories:

- Historical Volatility Models – Here can be found Simple Moving Average (SMA), Exponentially Weighted Moving Average (EWMA) or AR(F)IMA.
- Generalised Autoregressive Conditional Heteroskedasticity (GARCH) Models - This group includes the whole ARCH & GARCH zoo: EGARCH, GJR-GARCH, IGARCH, TGARCH, etc.
- Realised Volatility (RV) Models – This set involves Heterogeneous Autoregressive (HAR) model and Realised GARCH (RGARCH).
- Stochastic Volatility (SV) Models – There are two major examples: Heston Model, Autoregressive Stochastic Volatility (ARSV).

- Implied Volatility Models – To this collection, we would place Option Pricing Models (e.g., Black-Scholes model).

The modern literature mainly utilises techniques from the first three groups. Therefore, we will primarily pay attention to them. Poon and Granger (2003) themselves conducted a study comparing the ARIMA and GARCH methods. They found out that the former outperformed the latter in 22 cases, while the opposite outcome occurred 17 times. Overall, they do not make any inference in favour of either of those and stress out the dependency on the data and objective. Another research was done by Bollerslev et al. (1992), who arrived at conclusion that GARCH models yield slightly better results than ARIMA because of the ability to capture volatility clustering.

The paper from McMillan et al. (2000) points out that the moving average models provide more accurate values for long-term forecasting windows, while GARCH models appear to be more suitable for 1-day ahead forecasts, especially using high-frequency data. Generally, it seems that the definite answer is not yet resolved.

If we look into the GARCH family models, we find out there is a myriad of modifications. Frimpong and Oteng-Abayie (2006) together with Brooks and Burke (2003) reach a conclusion that the superior specification seems to be GARCH(1,1). According to them, it best fits the data and estimates volatility.

Further, Miron and Tudor (2010) discover that exponential GARCH (EGARCH) yields most precise results within the family of models relying on asymmetric shocks. On the other hand, Rastogi et al. (2018) determine that any deviation from the process can be sustained in the future, Therefore, they infer that integrated GARCH (IGARCH) is the optimal choice.

The third (and last) subset in this chapter encompasses realised volatility models. This concept of realised GARCH was introduced by Hansen et al. (2012) and consisted in using the realised measures help explain the conditional variance of returns. The authors themselves compared the accuracy of their newly presented model, which showed it can outperform standard GARCH methods. Kambouroudis et al. (2016) in their paper on major stock markets upheld that realised volatility contain additional information and is proved useful. In contrast, Huang et al. (2016) demonstrate a single realised-GARCH model is not sufficient for long-term volatility forecasting. They show that adding HAR-specification, presented by Corsi (2009), into the model catches the long-memory dynamics in a much better way.

Another extension of realised GARCH was done by Maciel et al. (2016), who added jumps to the equation. This change is responsible for enhancing robustness to noise and outliers. They use the idea of Andersen et al. (2009), who proclaimed that the return variation comprises jump and non-jump components with the latter having the biggest effect.

In conclusion, we may find dozens or maybe hundreds of traditional volatility models, but the literature showed us there are no clear winners or losers. The performance is rather sensitive to the data (different markets $\rightarrow$ different patterns), specification (daily/minute frequency) and forecasting window (days, months).

2.5 Unconventional Models for Forecasting Volatility

In the recent years, computers have experienced significant improvements, which made people think about the usage in investing and trading. Models relying on thousands or millions of calculations were becoming feasible since the operational power could finally handle it. This led to the rise of previously introduced data science techniques like artificial neural networks (ANN), random forests, support vector machines (SVM) in practice. The application of these methods varies across many fields in the world starting from image recognition to self-driving cars. We now call them deep learning (DL), machine learning (ML), or in general artificial intelligence (AI).

The reason they are becoming more popular is that they can uncover nonlinearities, asymmetry and very complex structure of financial data. One of the first types of these techniques used were support vector machines (SVM). Apart from the features mentioned in the previous article, researchers would use them because of their ability to handle polynomial inputs (e.g., lagged returns) or even incomplete data (Cortes and Vapnik (1995)). For instance, Gavrishchaka and

Banerjee (2006) ran a study on a four-year S&P 500 data and found that SVMs can efficiently compare to traditional and extended GARCH models and in some cases even outperform them. In comparison to the artificial neural networks, Sheta et al. (2015) point out in their study the advantage of SVM to choose various kernels, which best fit particular situations.

On the contrary, there exist several aspects in which SVM lacks when compared to ANN. As Liu (2019) puts it, ANN are able to handle large amount of (raw) data (more than 10Gbs) and can produce long sequence data prediction. Liu (2019) himself bases his research paper on a long short term memory recurrent neural networks also known as LSTM RNN. This is a special type of ANN, introduced by Hochreiter and Schmidhuber (1997), that overcomes the problem of vanishing/exploding gradient, which is typical for recurrent NNs. Liu (2019) draws a conclusion that both LSTM RNN and SVM provide measurably better forecasts than traditional GARCH models in the field of financial indices S&P 500 and AAPL. Further, Petrozziello et al. (2022) took a close look at the performance of LSTM RNN against benchmark models including G-GARCH or GJR-GARCH. Using time series of the most common indices, they found out that neural networks behaved better than benchmark when high-volatility was present. Once the market was more stable, no significant differences took place.

Christensen et al. (2021) decided to compare machine learning (ML) methods including ANN, tree-based specifications and regularisation to a relatively traditional, yet high-quality HAR model. They showed that ML models beat the HAR even when simple lags are taken into account. Moreover, Tyuleubekov (2019) contributed to the literature by analysing traditional and non-traditional models in relation to the volatility forecasting. He made inference that ML proved better than the former group.

Overall, machine learning models seem to be a very perspective tool to forecast volatility. The academia in the last decade has shown their slightly superior performance compared to more traditional methods like GARCH or ARIMA, mostly because of the ability to handle big data and capture non-linearities in the estimation process. Nevertheless, conventional methods are in many cases, especially in stable market environments, equivalent to their more recent counterparts. Hence, the literature should not disregard them at all. We see, however, a potential in delving deeper in the machine learning approaches since the research is still very young in the world of financial data. As artificial intelligence is growing, so is the opportunity to use new or extended models, introduced in vastly unrelated fields, directly in financial time-series data. One of those is a special type of ANN called convolutional neural network (CNN).

2.6 Convolutional Neural Network

In this sub-chapter, we aim to briefly discuss the concept of convolutional neural networks in the literature. Detailed description of the method will be displayed in section 4 Methodology. Zhang (2018) states in his work that the history of CNN can be divided into 3 periods: theoretical proposition (1), model realization (2) and extensive research (3).

The first one dates back to the 1960s, when an American Canadian neurophysiologist Hubel showed that multiple layers of certain receptive field stimulate biological information from retina[1] to the brain. This finding brought the attention of Fukushima (1980), who utilised this concept in his work, which he called *Neocognitron*. This was a self-organising neural network able to capture patterns based on geometrical similarity. This property facilitates robustness of the model to the changes in position, shape and scale. Something that other types of neural network are very sensitive to.

In year 1998, when the second stage emerged, LeCun et al. (1998) proposed an algorithm using a gradient backpropagation to train the model, which they called LeNet-5. It featured characteristics, which are standard in CNN nowadays, like max-pooling, sigmoid activation function or weight-sharing. The network was able to read millions of bank cheques per day and is considered a milestone in the history of convolutional neural networks. It set a starting point in the application of this method in other fields like speech or face recognition.

[1] "Retina is a layer of photoreceptors cells and glial cells within the eye that captures incoming photons and transmits them along neuronal pathways as both electrical and chemical signals for the brain to perceive a visual picture." (Nguyen (2022))

The last epoch was in the name of AlexNet, presented by Krizhevsky in 2012. This CNN model took by far the first place in an image classification contest of ImageNet. Winning the event by a huge margin raised awareness among many researchers and practitioners, who subsequently started to pay attention to convolutional neural networks in greater detail. Therefore, updated version including Visual Geometry Group or Microsoft's ResNet were proposed. Along the time, CNN has been achieving substantial accuracy primarily in tasks related to image and video recognition, where its advantages played crucial role.

Regarding the literature on the application of this phenomenon in financial data, we may find several studies using the CNN. One of them was carried out by Di Persio and Honchar (2016), who examined the precision of one-day ahead stock market indices' price movements. The object of comparison was CNN and multilayer perceptron (MLP) & recurrent neural networks (RNN). In the end, CNN showed it can model the forecasts better than those other two architectures. Interesting enough was mainly the approach chosen by the authors, who considered 1D vector of closing prices, while disregarding further variables.

Another paper taking into account CNN was published by Gunduz et al. (2017). The scope of their work was to compare the performance of this network to logistic regression (LR) on predicting prices on data from 100 stocks in Borsa Istanbul. Unlike, Di Persio and Honchar (2016), they manipulated the information from the current and previous trading hours into 2D tensors. The CNN was shown to outperform the LR even when tested for robustness.

Ding et al. (2015) focused on S&P 500 price prediction using events like news titles as inputs. Once this was pre-processed, it was used in CNN and showed non-negligible improvements in comparison to baseline models. One of their principal conclusions was that deep CNN is able to capture long-term effects (up to 1 month) with better accuracy than any other feed-forward neural network.

Hoseinzade and Haratizadeh (2019) decided to go even further and managed a 3D input tensor beside the 2D ones. These 3D cubes enabled containing data from several markets at once, while the latter would store information about single features only. The highlights of their work are mainly two-fold. Firstly, CNN can handle input data from various sources. And secondly, deep CNN provide significantly enhanced results than shallow ANN.

All of these papers assume the input data in classical way as sort of features, possibly explaining the outcome variable. Jiang et al. (2020), however, have come up with an interesting, highly technical-analysis, idea to look at the stock returns as pictures and have sought to explain the future returns. This thought is based on the observed pattern of humans, for which a graphical representation is mostly more favourable than a matrix of raw numbers. The whole idea consists in training the model on images containing the opening, closing, low and high price, volume and moving average price. The authors thus converted this matrix-wise information into a visual one and adequately scaled it down. The final pictures resemble a simple black&white candlestick charts. In the end, they infer that the CNN is able to outperform conventional models and moreover, it appears to be robust to the context. This means that the predictability does not measurably vary when the input data is short or long-time scaled. Image recognition is heavily used for detecting patterns for classification in AI. But its usage in the field of financial data has been largely omitted. In fact, this exclusive approach seems to be unique in the pool of stock market. In addition, it has been only applied to stock returns, while the world of volatility has remained untouched.

This thesis, therefore, seeks to explore the forecasting power of stock market volatility, where the input data is a set of past price information of S&P 500, converted into images. Later, the forecast precision is compared to baseline models' architectures, described in the previous parts of this chapter. With respect to the reviewed literature, we operate with realised volatility based on 5-minute high-frequency data, as the most utilised proxy of volatility nowadays. To the best of our knowledge, this is the pioneering work to convert prices into images and using it as an input to convolutional neural network while forecasting stock market volatility.

3 Data

In this chapter, we introduce the data used to address the research hypotheses presented at the beginning part of this thesis. The focal point are the prices of S&P 500, which is a broadly recognised index tracking the performance of 500 large-scale companies listed on the US stock markets. It has been widely used in academia as a benchmark model for investment and trading funds and a great number of research papers base their studies on data coming from this index. The reason is that the S&P 500 encompasses roughly 76% of the US market capitalisation and 33% of the whole world's market capitalisation (Neufeld (2023)). Therefore, it can be considered a good proxy for the stock market. In line with the current literature, we concentrate our work on S&P 500 as well, to have the best comparison opportunity.

Our data come from one source only, which is the IES e-library. Thanks to my supervisor, doc. PhDr. Jozef Baruník, Ph.D., I was provided with high-frequency data measured at 5-minute intervals for E-Mini S&P 500.

This is a futures contract traded on the Chicago Mercantile Exchange marketplace. Its main advantages lay in giving us a continuous and highly liquid asset to analyse, while it still tracks the performance of the S&P 500.

This thesis is using a 10-year time period starting from January, 2010 and ending in December, 2019. The goal is to test how the method stands in relatively normal market conditions since our prime objective is comparing the performance to other broadly utilised techniques and this facilitates us an unbiased examination. This period should thus satisfy both sufficient length and no affection of any major stock market crisis like the global financial crisis or the Covid pandemic.

3.1 Data Introduction

Table 1 tells us we deal with 704 073 observations, for which we have accessed the following variables. Those include open, close, low and high price, followed by the volume. Every point relates to a 5-minute time period. Note that there are only minor differences across the four price indicators. This is due to the fact that the 5-minute period is very short and if there is a substantial jump within

Table 1: Raw High-Frequency Data Statistics

Statistic	N	Mean	St. Dev.	Min	Max
open	704,073	1,955.941	590.243	1,004.000	3,253.750
close	704,073	1,955.943	590.243	1,004.000	3,253.750
low	704,073	1,955.303	590.151	1,002.750	3,253.000
high	704,073	1,956.573	590.333	1,004.500	3,254.000
volume	704,073	5,929.859	10,955.160	1	276,094

*S*ource: Author's computations

a day, all the four indicators will be affected relatively the same in the following 5-minute periods. Volume is, on the other hand, extremely dispersed, as its values range from 1 to 276 000. The average value is approximately 6 000, which means the number of trading positions can skyrocket to more than 40x of its value.

3.2 Transformation to Daily Data

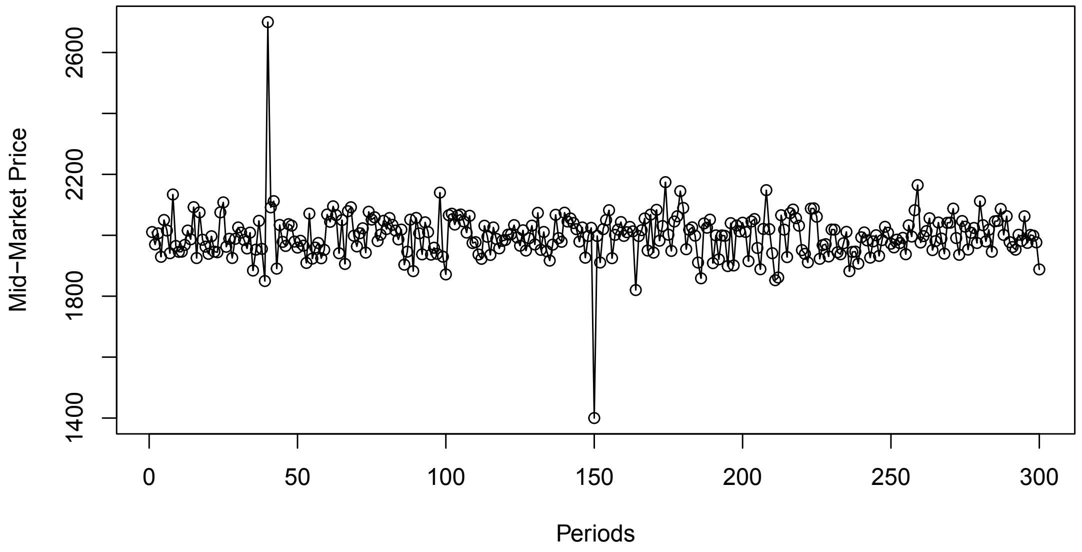
In this subsection, we introduce the calculations taken to transform the data from high-frequency to daily. This involves choosing the appropriate representative for each price indicator, summing up all the volumes within a day and calculating daily realised variance and volatility.

We will first pay attention to the prices. An ideal candidate for a daily open price is the open price in the first 5-minute period since it best matches the definition. Regarding the close price, we analogically chose the one in the last period of the considered day. The low price was chosen as the minimum of the low prices within a day, while the high was the maximum across the high prices.

One might argue that this method might overestimate the variation. In relation to Figure 1, imagine an arbitrary scenario with stable market prices over 24 hours, but two opposite sharp jumps occur. Then the low and high prices are extremely far from the rest, which suggests the market was very volatile. Some could argue it was in deed, while others could point out that these were just quick outliers, which do not represent the overall daily behaviour.

A solution would be to calculate the mean values across the day for all the four indicators separately to weight these jumps by duration. The issue with

Figure 1: Arbitrary Daily Mid-Market Price over 24 Hours



Source: Author's computations

this method is that the indicators will be of little variation, as described at the beginning of this chapter. This would cause problems when used as inputs in the model. Therefore, we will stick to the first technique in spite of its possible inefficiency in rare situations.

Let us follow with the volume. This time we only calculate the sum of all the 5-minute volumes within a day and divide it by the number of trading sessions on the same day. The reason behind this is that the market operates in different hours for some days and, for example, Fridays and Sundays would bias this variable.

The last part of data transformation involves volatility and variance. The discussion of all proxies is described in the literature review, but we will only focus on the realised measures further on. This manipulation involves applying the following formula 1. For given stock i on day t, we calculate realised variance as a sum of squared returns within that day. To adjust it for different trading hours on different days, this sum is then divided by the number of trading sessions m on day t. In case of 5-minute data, we can have up to 288 sessions, but some days only feature a maximum of 72. For returns, logarithmic transformation was applied and realised volatility is just a square root of realised variance.

$$RV_{i,t} = \frac{1}{m}\sum_{j=1}^{m} r_{i,t,j}^2 \tag{1}$$

For given asset and day, r_j is defined as:

$$r_j = log\left(\frac{closing\,price_j}{closing\,price_{j-1}}\right) = log(closing\,price_j) - log(closing\,price_{j-1}) \tag{2}$$

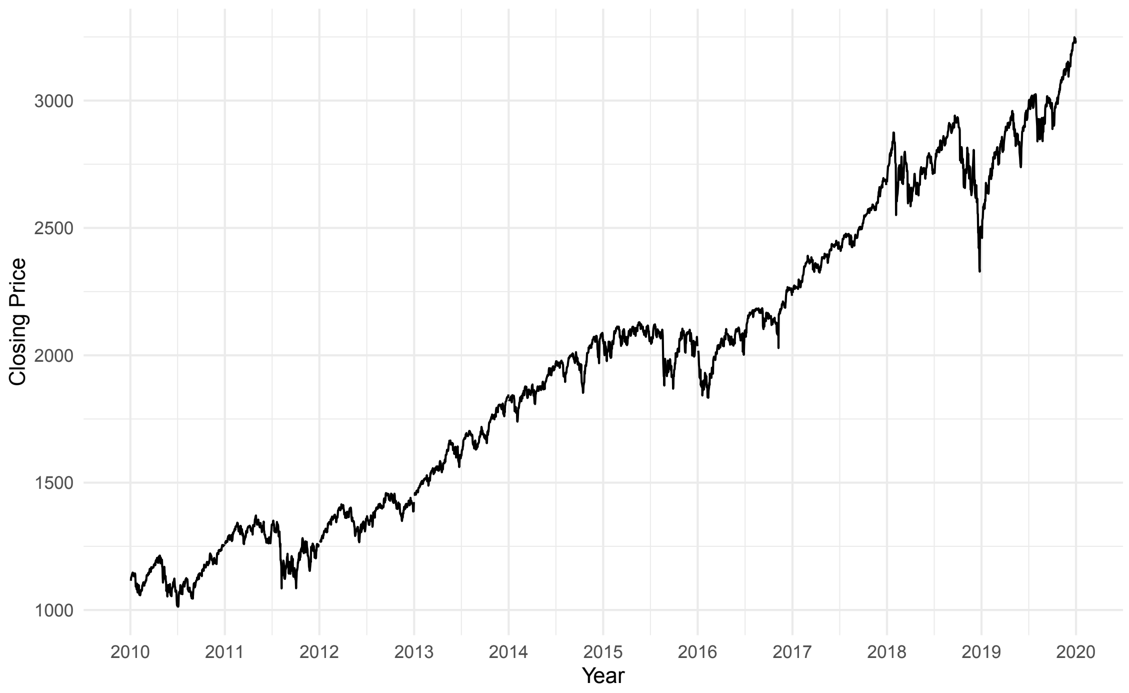
Realised volatility is then defined as:

$$RVol_{i,t} = \sqrt{RV_{i,t}} \tag{3}$$

3.3 Data Inspection

This subsection presents and visualises the daily data obtained by applying the previous calculations. Let us first look at closing prices.

Figure 2: Closing Price from 2010 to 2019

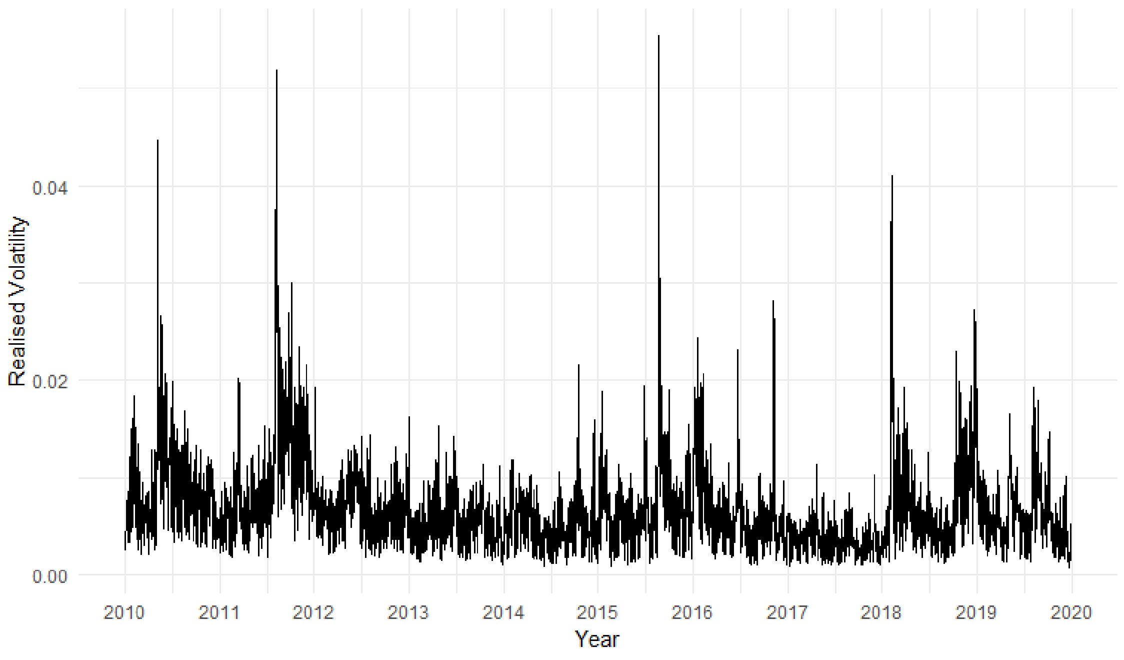


Source: Author's computations

As mentioned at the beginning of this chapter, the data range from 2010 to 2019, which should be a relatively less volatile growing period. The price has approximately tripled over the interval with a substantial annual return of 11%. There is one jump at the beginning of 2019, but the overall mood is rather less volatile.

The following figure 3 represents calculated realised volatility. The values seem to be centred around zero with several peaks.

Figure 3: Realised Volatility



Source: Author's computations

There are also some apparent volatility clusters, each lasting for a couple of months. The first one occurs at the beginning, where volatility remained relatively high. Later, we have a visible period of stable average volatility for 5 consecutive years. This epoch is followed by constantly low values in the whole year 2017, while we can spot volatility to return to previous levels in the final period.

It can be noted that there are a few points, where the returns sharply oscillate. These peaks correspond to the few deviations of the linear trend of the market.

3.3.1 Transformation to Images

This section describes how the data is transformed so that it can be used in the models. All explanatory variables including the opening, closing, open, high price and volume are converted into images, while the explained variable realised volatility is standardised into the 0-1 range.

Let us first focus on the former part. It is interesting to study the effect of the last week, month and quarter in forecasting the future value of volatility since these are easy to understand intervals, which may exhibit certain market patterns. Therefore, we first need to split the input data into these bins of respective number of days. The analysis will be introduced on five days for simplicity, but the procedure holds for the other two types as well unless specified otherwise.

Let us consider five observations from the dataset. Their matrix specification is displayed in the following table 2 and the transformed version is in table 25.

Table 2: Input Data 5 Days - Example Bin

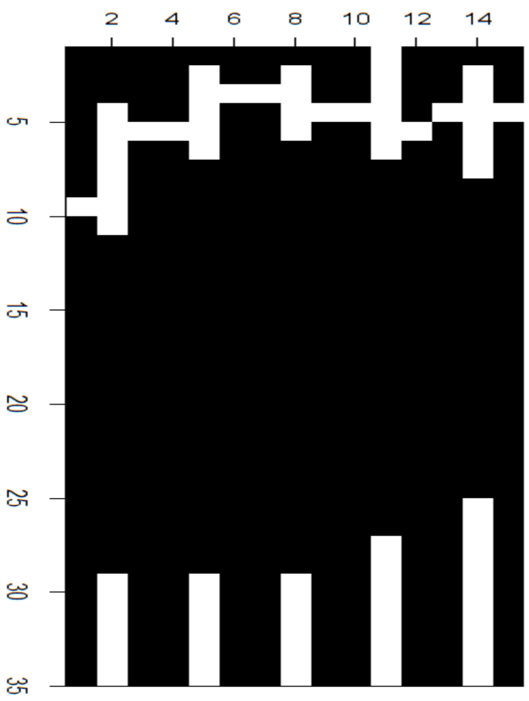
Date	Open	High	Low	Close	Volume
2010-03-22	1149.75	1163.75	1146.75	1162.00	6806.77
2010-03-23	1161.75	1170.50	1159.00	1168.50	6624.35
2010-03-24	1168.25	1169.50	1161.00	1164.00	7062.56
2010-03-25	1164.25	1176.50	1158.25	1163.00	8539.46
2010-03-26	1163.25	1169.75	1156.50	1163.25	10809.91

*S*ource: Author's computations

The bins are joint; therefore, the following input contains 4 common data points and adds one new. The rationale behind this approach is to ensure a continuous dataset, facilitating the ability to forecast each subsequent day. As displayed in table 25, the price indicators were transformed in such a way that there are three columns for each day. The first one stands for the opening value, the second one is the low-high difference and the third one represents the closing level, at which the price settled. The values for price indicators start at the 16th pixel and range up to pixel 35, potentially covering 20 price intervals. The volumes are located at the bottom of the image and can achieve pixels from 0 to 10.

The most important part was to adequately scale the values into the respective pixels. For every data input, i.e., one image of n days, the price indicators were

Figure 4: Input Data 5 Days - Transformed Matrix



Source: Author's computations

standardised by using the maximum and minimum values of the last $(sm_p + 1) \cdot n$ values including the current input n. [2] The sm_p is a smoothing parameter, which was set to 3. For $n = 5$, this would give us 15 days back plus 5 days in the current data input for every input. For 20 and 60 days, the total number of days used in standardisation of one input is equal 80 and 240. The number of 3 for smoothing was chosen in order to have the data scaled adequately by the encompassing days. If the scaling was conducted on each data input separately, it might happen that the days, which are common for two following data inputs, could have totally different pixel representations. This would result in high fluctuation and low consistency.

On the contrary, if the standardisation was done based on the whole training dataset, there would be little to no variation in all data inputs since the price differences are extremely high in the period of 10 years. Hence, the universal value

[2]Fewer observations were taken for the first 15 data inputs, as there was not enough data by definition. Similarly to 60 and 180 for the other two types of input

of 3, which was chosen semi-arbitrary and is, of course subject to improvement in later studies.

The other data inputs featuring 20 and 60 days contain 60 and 180 number of columns, respectively. The number of rows is also subject to discussion. Due to a more likely enhanced variation when more days are used, it was decided to use 40 and 80 price intervals, respectively, to capture the differences. Sample images are illustrated in the appendix.

4 Methodology

In this section, we introduce the concept of convolutional neural networks, which is the type of NN used for volatility forecasting in this thesis. Firstly, we present the theoretical background of artificial neural networks and then pay attention to its special type called convolutional neural networks (CNN).

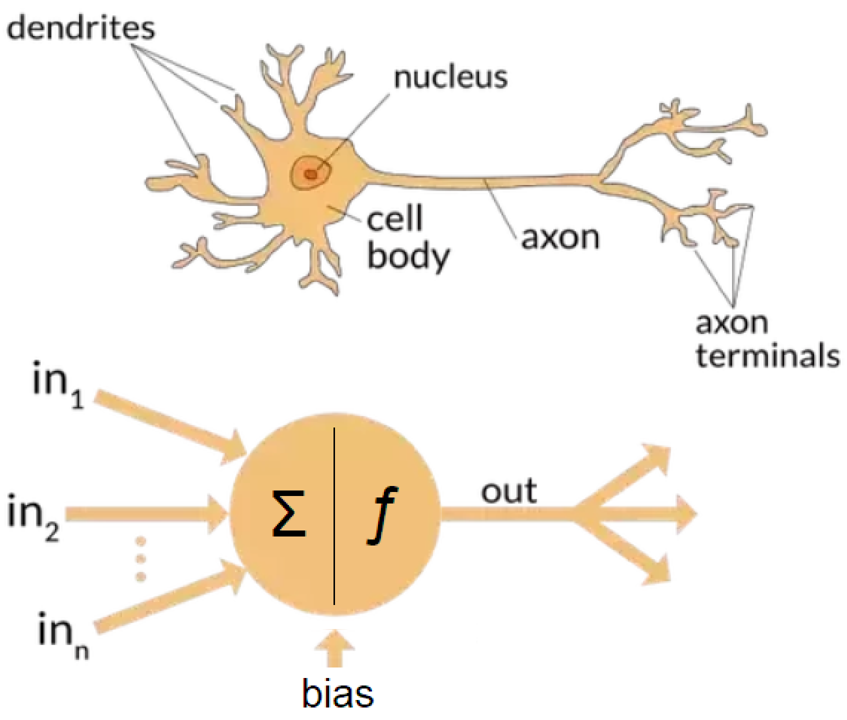
4.1 Artificial Neural Networks

Artificial neural networks is a complex mathematical model based on biological structure of human brain (Nagyfi (2018)). The key component of human brain is a cell called neuron. We can find billions of those within the neural network and each is responsible for receiving, processing and transmitting information. Firstly, there are dendrites through which the information is delivered to the main cell. Then, this new piece of data is processed within the neuron's body before it is sent forwards via axon to synaptic terminals.

Artificial neural networks aim to resemble this behaviour, although in a very simplified way. At the beginning, there are a number of inputs, usually variables. These are all weighted by particular weights, a bias term is added, and everything is then summed to a scalar. This whole initial step corresponds to the dendrites. Further, we move to what happens in the main cell. The obtained scalar serves as an input for a so called activation function, which transforms the data and passes it along.

What happens next depends on the deepness of the network. It can either serve as an input to other neuron in the following layer or directly represent the output value(s). Note that there might be more neurons in one layer as well as more hidden layers than one in the process. Figure 5 shows how the mathematical model is based on the real world observation.

Figure 5: Biological and Artificial Neuron



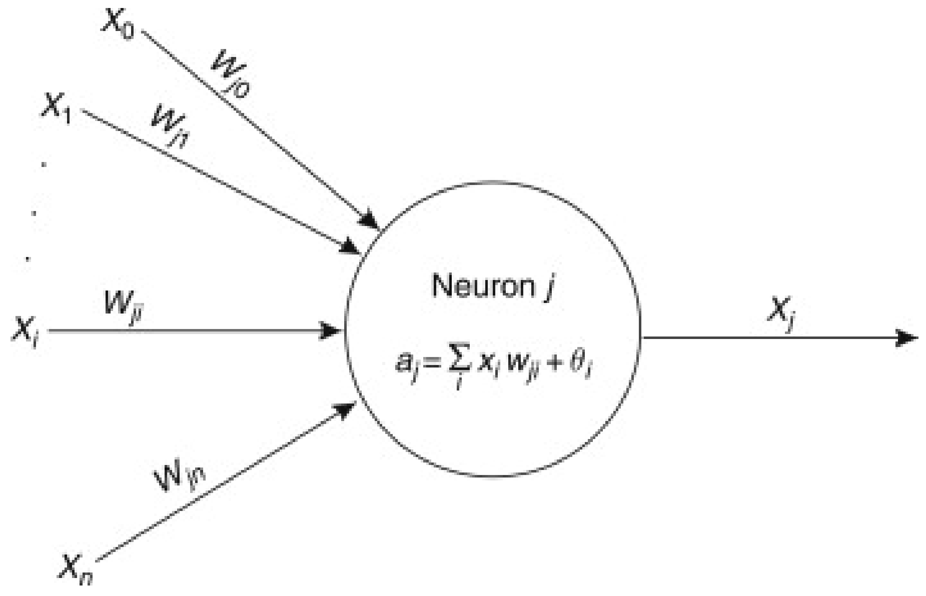
Source: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

4.1.1 ANN in detail

In this subsection, we dive into artificial neural networks in greater detail. Let us consider a very simple scenario, where we have several input variables n for estimating the output value. These inputs can be denoted as x_i with $i \in [1, 2, ..., n]$. Each input can be understood as a variable affecting the output, generally in a different way.

The next step is situated in a hidden layer, which is a layer consisting of one or more neurons, depending on the complexity of the network. Each of the neurons $j \in [1, 2, ..., J]$ is connected to the input variables by its specific weights w_{ji} with $i \in [1, 2, ..., n]$. At this moment, the weights are generated randomly from a chosen distribution. For each neuron, we perform sumproduct of these two variables and add a bias term θ, initially generated as 0 (LeCun et al. (1998)). This yields a scalar value, which is denoted as a_j. Graphical representation can be seen in Figure 6.

Figure 6: Neuron Calculation



Source: Lek and Park (2008)

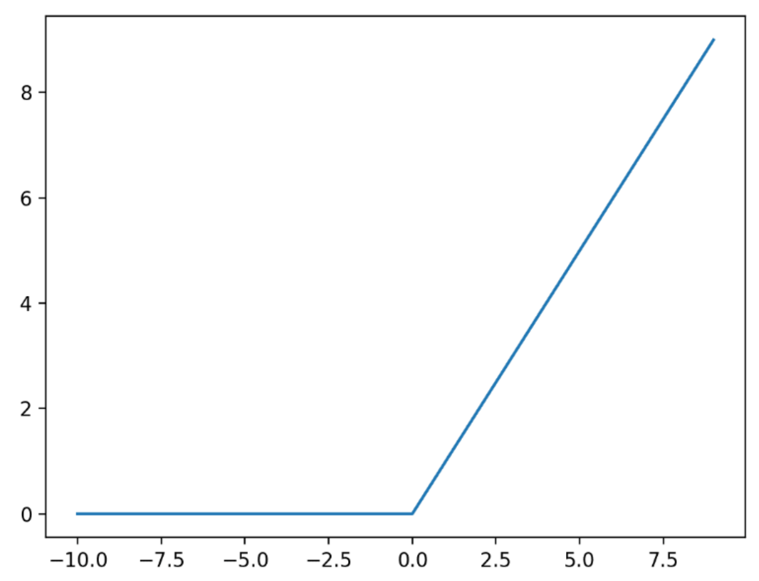
After the process is conducted, the value a_i enters as an input into an activation function within the particular neuron. This function can theoretically be of any kind, but there are certain features, which are advantageous. One of them is monotonicity, which allows the function to be consistent both on input and output. Second is differentiability, which is important for backpropagation, discussed later on. Nevertheless, there are more characteristics, but their usage hinges on the concrete type of neural network (Zupan (1994)).

For a hidden layer, there are two most utilised functions. The first one is called Rectified Linear activation function, shortly ReLU. It is calculated in the following way.

$$max(0, x) \tag{4}$$

Therefore, the output value is always non-negative. One of the reasons this function is popular is its simplicity, which enhances the calculation speed. Moreover, it helps prevent vanishing gradients issues, which will be discussed in section backpropagation together with some drawbacks. Nowadays, this function is being used mostly for convolutional neural networks, which is the technique used in this thesis. The following figure 7 shows its input-output relation in a graph.

Figure 7: ReLU Activation Function



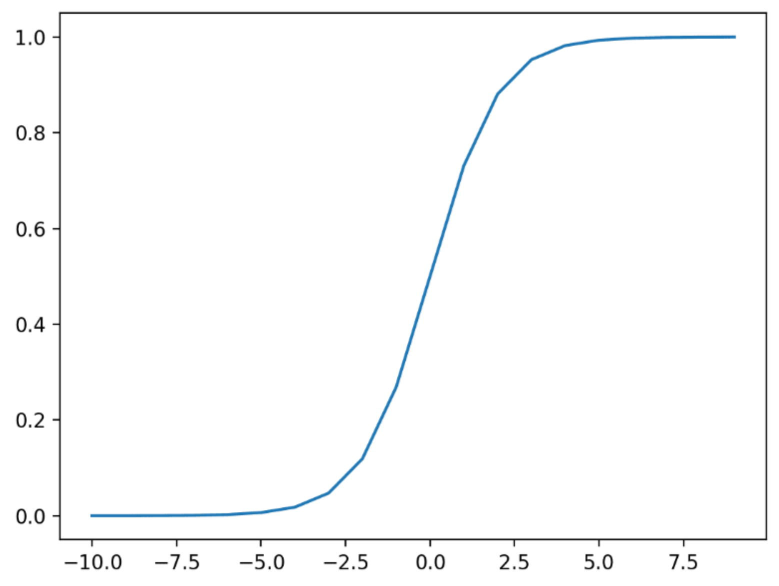
Source:
https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning

Second one is a sigmoid function, also referred to as a logistic function. It is defined as

$$\frac{1}{1+e^{-x}} \tag{5}$$

Contrary to ReLU, it is non-linear and has a smoother gradient, which is preferable in backpropagation. Nevertheless, it may suffer from vanishing gradient issue. The graph is in figure 8.

Figure 8: Sigmoid Activation Function



Source:
https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning

After the value is processed through the activation function, it may either continue to another hidden layer if the network is deep, or directly reach the output layer. Let us discuss the latter procedure, which is also analogical to the previous step, meaning there exists one weight for each connection and one bias for each neuron (Zupan (1994)).

There are two main activation functions for an output layer. Again, we may use the advantages of the sigmoid function. It is efficient if the decision is binary, whether the output belongs to one class or the other. The resulting value can be thus deemed as a probability of the observation to be type 1. Nonetheless, it can also be useful when the output is continuous and ranges between 0 and 1, which will be our case

The other heavily utilised function is softmax. It is defined in the following way.

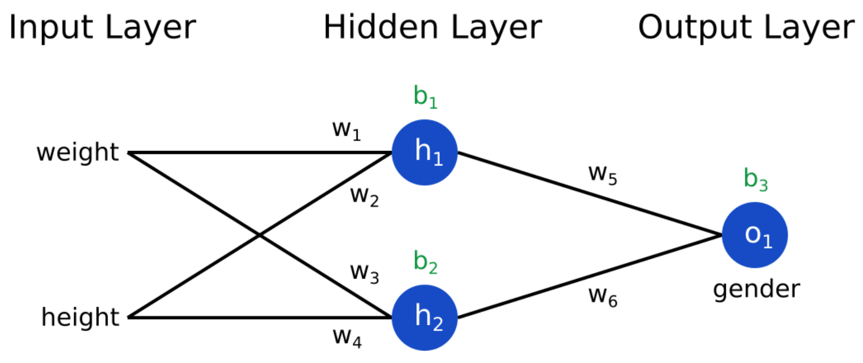
$$\frac{e^x}{\sum e^x} \tag{6}$$

For every output value x, it calculates e^x and divides by the sum of these exponential values. Therefore, it is proven useful for multiclass classification (Pratiwi et al. (2020)).

Let us wrap up the forward feed on the example in figure 9. There are two variables on the input, one hidden layer with two neurons and one output layer with one neuron. The input variables are scaled by w_1 to w_4 and two sumproducts z_1, z_2 are formed. Then, bias terms b_1 and b_2 are added to them and the activation functions return the output values. These are then multiplied by w_5 and w_6 and summed up. This scalar together with bias b_3 serves as input z_3 for the output activation function, which yields a value between 0 and 1 in the case of sigmoid.

Therefore, we are left with 6 weights and 3 biases that make our 9 trainable parameters in total. The most important part takes place in the middle layer, which processes the data through the activation functions. If those functions were just linear, the method would not be much distinguishable from conventional linear regressions. Nonlinear transformations, on the other hand, facilitate complex causalities, but may not be efficient in basic data structures. One must always use the advantages of appropriate tools and choose adequate number of hidden layers and neurons. Opposite to our example, there can be deep models that have thousands or millions of parameters to estimate.

Figure 9: Simple Neural Network



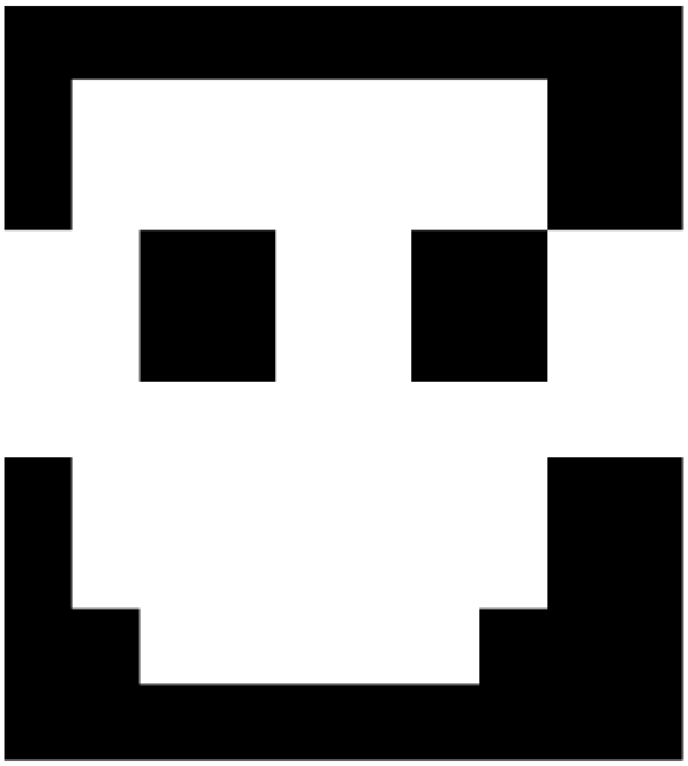
Source: https://victorzhou.com/blog/intro-to-neural-networks/

4.2 Convolutional Neural Networks

Throughout the latest evolution of computers, there have been a plethora of neural networks' derivatives. One of them is called convolutional neural networks. This is a process, which starts by performing a so-called convolution that is followed by regular neural network's operations described in the previous chapter. What happens in the first part will be discussed in this subsection.

The word convolution stems from a mathematical operation involving a product of two functions or in our case two sets (Jeong (2019)). For clarity, imagine a matrix of 10x10, which contains only black or white pixels. In programming terminology, these pixels are viewed as 0 for black and 255 for white units. Representations of both of these is located in figure 10 and table 3, respectively. Note that we used a division of 2 in the table to make the process simple, while maintaining generality.

Figure 10: Sample Matrix



Source: Author's computations

To convolute this object, we need another matrix called kernel. This is an NxN

Table 3: Sample Matrix as Numbers

0	0	0	0	0	0	0	0	0	0
0	0	122.5	122.5	122.5	122.5	122.5	0	0	0
0	122.5	122.5	122.5	122.5	122.5	122.5	122.5	0	0
0	122.5	122.5	122.5	122.5	122.5	122.5	122.5	0	0
122.5	122.5	122.5	122.5	122.5	122.5	122.5	122.5	122.5	122.5
122.5	122.5	0	0	122.5	122.5	0	0	122.5	122.5
122.5	122.5	0	0	122.5	122.5	0	0	122.5	122.5
0	122.5	122.5	122.5	122.5	122.5	122.5	122.5	0	0
0	122.5	122.5	122.5	122.5	122.5	122.5	122.5	0	0
0	0	0	0	0	0	0	0	0	0

*S*ource: Author's computations

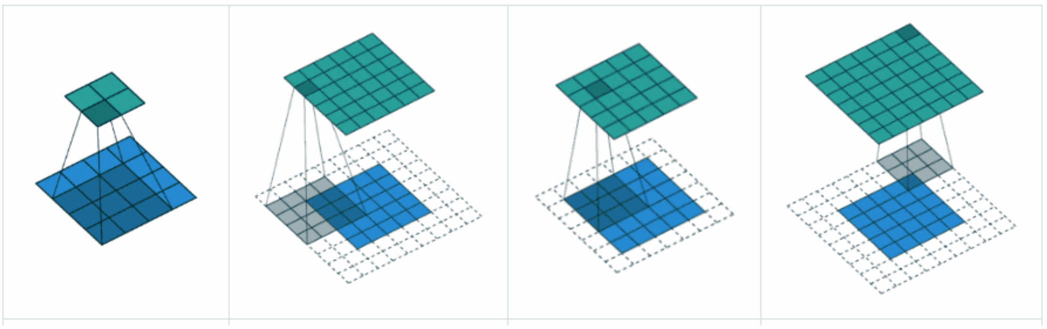
dimensional object, whose values are first randomly generated and then continuously modified in the training phase. Let us assume that the matrix looks like in table 4. This way the sample matrix 10x10 should transform into a blurred version of itself since the emphasis is gradually diminishing farther from the adjacent values. It should be noted that this is just a single kernel for one convolution. There can exist multiple kernels independent to each other, which introduce higher complexity into the process by increasing the number of channels, i.e., depth dimension. This means that the height and width of the image usually decrease, while the depth grows. The reason is capturing more complexity to the model. The process description will, however, work with a single channel procedure for brevity.

Table 4: Sample Kernel

	1	2	1
$\frac{1}{16}\times$	2	4	2
	1	2	1

Now since we have these two matrices, the next step is to choose the appropriate padding. This defines the level of overlapping of the two objects when calculating the product and thus the dimension of the output matrix. Standard approaches are *valid* and *same* (Jeong (2019)). The former uses only the actual matrix 10x10 and in turn reduces the dimension, which may help make the computation faster. The latter introduces helping cells around the matrix to maintain the same dimensions on the output. This can be beneficial because the information from the edges is used on the output but on the other hand, it causes the calculations to be more demanding. Clearer explanation is in figure 11, where *valid* padding is the most left image and *same* is the third from left.

Figure 11: Different Types of Padding



Source: Jeong (2019)

The interconnected step to the padding parameter is stride. This defines the number of columns and rows, by which the kernel moves to the right (and down) in one step (O'shea and Nash (2015)). Usually its value is equal to one to use the advantages of the whole dataset at maximum. Nevertheless, strides of more than one can be used as well in order to make the calculations faster.

The actual convolutional process consists in performing a sumproduct of the kernel matrix and each area of the convoluted matrix, which is relevant. Exact definition is in equation 7 below.

$$\mathbf{I} = Input\ matrix\ mxn$$

$$\mathbf{K} = Kernel\ matrix\ pxq$$

$$\mathbf{O} = Output\ matrix$$

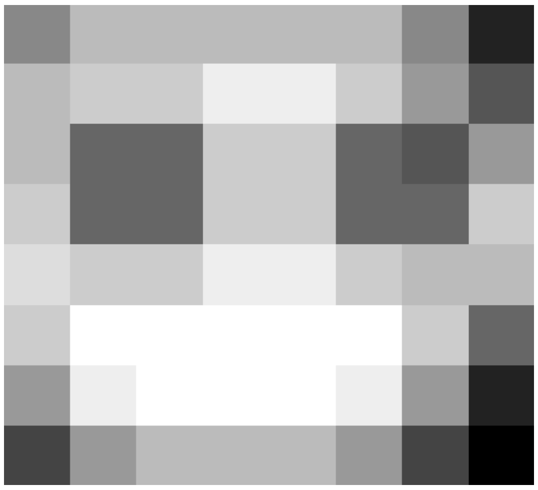
$$O_{i,j} = \sum_{u=0}^{p-1}\sum_{v=0}^{q-1} I_{i+u,j+v} \cdot K_{u,v} \tag{7}$$

The dimension of the output matrix can be calculated as:

$$m_{\text{out}} = \left\lfloor \frac{m - p_{\text{pad}} + 1}{s} \right\rfloor$$
$$n_{\text{out}} = \left\lfloor \frac{n - q_{\text{pad}} + 1}{s} \right\rfloor$$

After the convolution is performed, we use the ReLU activation function, which makes sure the output values are within the 0 to 255 range (O'shea and Nash (2015)). The resulting matrix is in figure 12.

Figure 12: Sample Matrix Convoluted



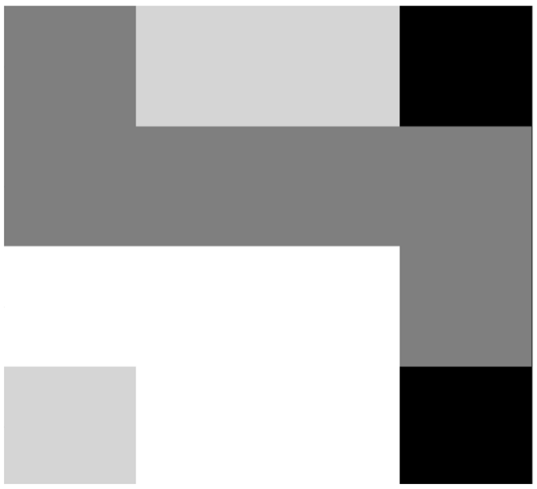
Source: Author's computations

The next step involves pooling, which is, similarly to convolution, applying a filter to the input map. Nevertheless, this time there is no kernel and we only need to set the dimensions and method of the filter (Wu (2017)). The former is usually 2x2, but larger space can be used for bigger images. In regard to the method, the literature most frequently uses max pooling and average pooling. As the titles suggest, we either take the maximum value or the average of the sample.

The stride is usually equal to 2 in order to have non-overlapping data and reduce more dimensions. Again, we start in the top-left corner and move towards bottom-right. The reason for this operation lies in heavily reducing the number of parameters, while still carrying the information further on.

The final matrix after applying the max pooling is visible in figure 13.

Figure 13: Sample Matrix after Max Pooling



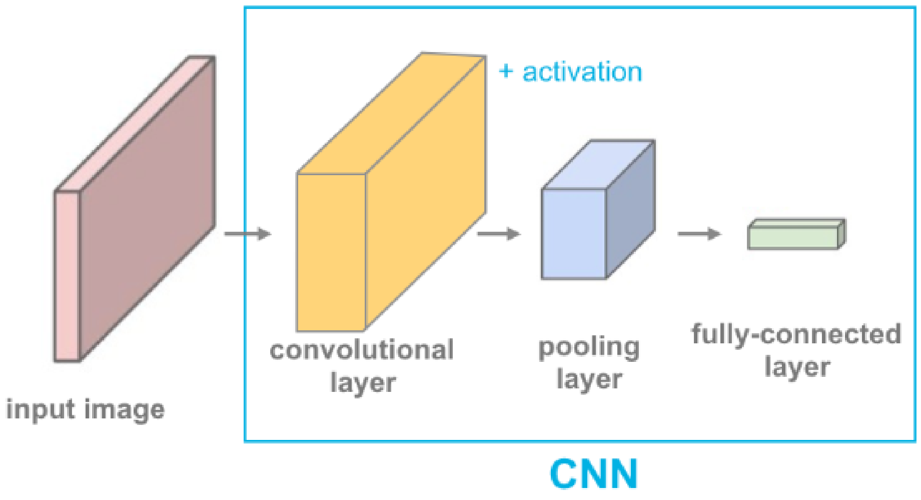
Source: Author's computations

As we can see, the final matrix contains a very narrowed information compared to the original one. Yet, it is in deed sensitive to the input matrix. Depending on the deepness of the network, one might repeat the convolution and pooling n-times to effectively decrease the dimension of the network.

Nevertheless, the last step is always flattening the final output matrix into a single column and use it as an input into a fully connected neural network, which was described at the beginning of this chapter. An image representation of the previous process is illustrated in the following picture 14

We have an input image, on which we perform convolution with kernel(s). Then, ReLU activation function is applied and the image proceeds to max pooling, which cuts the dimensions even lower. This may repeat until the complexity is exploited. Finally, the signal converges into a fully-connected layer, in which standard NN operations come to play. This was just a narrow explanation of how the signal actually goes forward in a CNN. Throughout the quick evolution, data scientists have come up with a plethora of modifications that occur within the process, but these introduce too much complexity for our rather simple models.

Figure 14: CNN Architecture



Source: Udacity (2024)

4.2.1 Training of NN

So far we have assumed the parameters like the kernels, weights, biases somehow exist and we only need to plug in the input image. This section will introduce how to optimise those parameters.

After each input flows through the network until the very end, it needs to be evaluated how close it is to the value to compare with. As mentioned in the previous chapter, realised volatility was standardised to range from 0 to 1. Therefore, we can use the advantage of a sigmoid function in the output layer to return values within this interval as well.

There exist many alternations of the so-called loss function, which quantifies the prediction efficiency. The criteria, which must or should be met by this function include monotonicity, so that the loss always grows when predictions are poorer. Another one is non-negativity that causes the error to be either positive or 0, in case of perfect prediction. Further, the loss function should be convex in order to have all the local minima global as well. This is a crucial feature for gradient optimisation described later on (Ramachandran et al. (2017)).

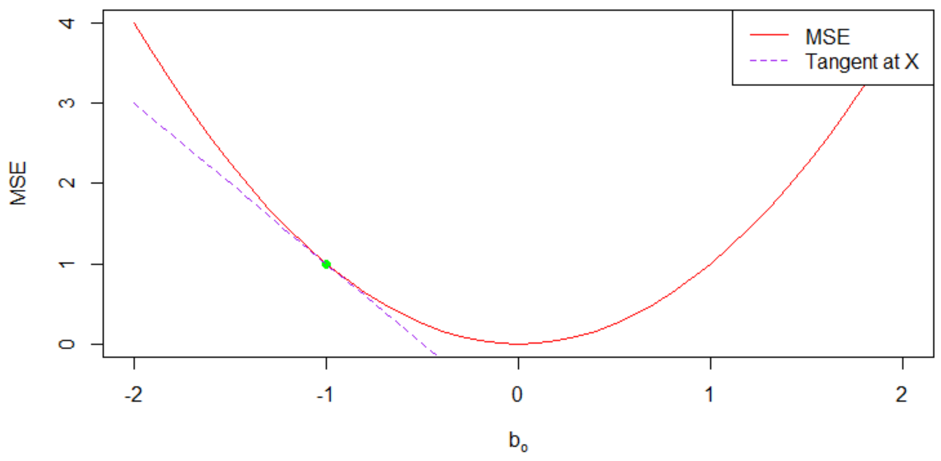
One of the ways to evaluate the outcomes is to use the mean squared error

(MSE), which is calculated in the following way. Assuming $\mathbf{x}$ is a matrix of input variables, θ is the set of trainable parameters and y is the real outcome values, our loss function is defined as in equation 8. We want to find such θ, which minimise this function L (Koyuncu (2020)).

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - f(x^{(i)};\theta))^2 \tag{8}$$

Let us go back and use the already defined simple NN in figure 9 with two neurons on the input, two in the hidden layer and one on the output. After passing one observation to the network, we can calculate the loss function (LF) for the first value with respect to the predicted value. Further, we can prolong it and calculate it with respect to the bias term in the output neuron. Such a graph might look similarly to the one in figure 15. If, for instance, the LF value would be equal to 1, as in the plot, the best way to diminish the error is to move to the right towards the minimum. But since it is not straightforward to tell where the minimum is, it is necessary to calculate the derivative of MSE with respect to b_3 at this point and go in the opposite direction. That is to go to the left if positive and to the right if negative (Wythoff (1993)).

Figure 15: Loss Function wrt to b_3



Source: Author's computations

To calculate the derivative, we can simplify the MSE expression to facilitate the process by multiplying it by 0.5, while maintaining generality, and use the chain rule to compute it (Wythoff (1993)).

The MSE equation is given by:

$$MSE = (y - f(\mathbf{x};\theta))^2 = (y - \hat{y})^2 \implies -\frac{1}{2}(y - \hat{y})^2$$

The partial derivative of MSE with respect to $\hat{y}$ is:

$$\frac{\partial MSE}{\partial \hat{y}} = y - \hat{y}$$

Then, the output function is sigmoid defined as $\sigma(z_3) = \frac{1}{1+e^{-z_3}}$. With z_3 being the inputted weighted sum in the output neuron, the first derivative is equal to:

$$\frac{\partial \sigma}{\partial z_3} = \sigma(z_3) \cdot (1 - \sigma(z_3))$$

Last, we use the fact that our output function $\hat{y}$ is sigmoid. The partial derivative of MSE with respect to z_3 is thus computed as:

$$\frac{\partial MSE}{\partial z_3} = \frac{\partial MSE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} = (y - \hat{y}) \cdot \sigma(z_3) \cdot (1 - \sigma(z_3))$$

To finally fetch the derivative of MSE with respect to b_3, we use the fact that the derivative of the weighted sum z_3 with respect to b_3 is equal to 1:

$$\frac{\partial MSE}{\partial b_3} = \frac{\partial MSE}{\partial z_3} \cdot \frac{\partial z_3}{\partial b_3} = (y - \hat{y}) \cdot \sigma(z_3) \cdot (1 - \sigma(z_3)) \cdot 1$$

All other parameters, including the kernels during convolution, can be computed analogically using the chain rule (Wythoff (1993)). For example, derivatives of w_5 and w_6 would be of the following expressions:

$$\frac{\partial MSE}{\partial w_5} = (y - \hat{y}) \cdot \sigma(z_3) \cdot (1 - \sigma(z_3)) \cdot h_1$$

$$\frac{\partial MSE}{\partial w_6} = (y - \hat{y}) \cdot \sigma(z_3) \cdot (1 - \sigma(z_3)) \cdot h_2$$

Before, we deep dive into the θ updates, we need to introduce learning rate α, which is positive and determines the magnitude of the jump. Small values might make the updates too slow, while big ones may cause the parameter to oscillate around the minimum and stay too far from it (Moreira and Fiesler (1995)). There is, however, a possibility to change the α in later stages of the training. Regular values in the literature often vary from 0.01 to 0.001.

4.2.2 Stochastic Gradient Descent

The process of effectively approaching the minimum value can be performed by multiple techniques nowadays. Most of them, however, rely on stochastic gradient descent, also known as SGD. This method has the following characteristics.

Input : Learning rate α and initial values of parameters θ

Output : Updated θ

The training procedure is run until the predefined condition is met, which is usually the number of epochs (Amari (1993)). And an epoch is usually a complete pass of the training data through the network. Now, there is a trade-off how frequently to update the θ. On the one hand, the literature works with updating the parameters directly after each observation is processed. This can make the gradients subject to noise and prolong the calculations. On the other, the update can occur after all the considered data are processed. Having many observations reduces noise but is not very efficient since not many updates actually happen. Thus, it would take many epochs to find the global minima for the network. Hence, probably the best way to deal with this trade off is to compromise and set a number of examples, after which the parameters θ update. These examples are sampled from the data usually without replacement and are referred to as batches. The number of examples is preset as a hyperparameter. We can find the detailed method below.

1. Sample a batch of m examples without replacement from the training set $(\mathbf{x}^{(i)}, y^{(i)})$

2. Run these examples through the NN, calculate the loss function and subsequently the parameters' gradients for each of them

3. Average the gradients over the examples $\mathbf{g} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), y^{(i)})$

4. Update the parameters θ by the product of gradients $\mathbf{g}$ and learning rate α

5. $\theta_{t+1} \leftarrow \theta_t - \alpha\mathbf{g}$

As a result, parameters θ update after each batch and theoretically do so $N_{batches} \times N_{epochs}$ times. Nevertheless, the practitioners have come up with several alternations to the simple SGD method. One of the most discussed ones is called adaptive moment estimation with weight decay or AdamW in short. Its advantages consist in that it more effectively updates the parameters by incorporating a weight decay and momentum. The former is a way to avoid overfitting the model by introducing a so called penalty to the loss function (Krogh and Hertz (1991)). Momentum is a feature, which accelerates convergence to the global minima by maintaining the overall direction of the convergence path (Moreira and Fiesler (1995)). This may come in hand when an outlier appears in the batch. On the other hand, this algorithm is more complex and computationally expensive than SGD.

This thesis uses AdamW in the CNN architectures because it has shown improved results in the literature (Tran and Cutkosky (2022)), despite its potentially higher computational costs.

4.2.3 Hyperparameters Choice

This section presents the choice of hyperparameters used to train the models. Since this method was omitted in the world of predicting volatility, it is not straightforward to know how to set those optimally. We, however, followed basic CNN architectures, stemming from the pieces of work of Jiang et al. (2020) and Zeiler and Fergus (2014) and made several versions of the architectures with many alternations. While still facing the uncertainty of the ideal specification, this procedure should decrease the probability of omitting it.

Overall, 96 different types of CNN were chosen for the input bins of five days. The 20-day input had 36 distinct specifications and the 60-day input were trained on 16 models. None of the models can be repeated in the whole pool of 148 combinations since the initial convolution layer is different for the three bin groups. Further, the inverse relationship of the input days and number of models is due to the computation power, which is sharply increasing with introducing complexity by higher number of input days. Therefore, reducing the models' versions was chosen in order to enhance the required deepness. Particular architectures are listed in the appendix.

5 Results

5.1 CNN Performance

In this section, we introduce the main results of the thesis, which is the performance of the image-based method of predicting volatility against the conventional counterparts. The models were evaluated based on both root mean squared error (RMSE) and mean absolute error (MAE). Within the 3 sorts of bins, the top five performing models based on RMSE on testing data were used and average values of these statistics were taken. All models were initially trained on the training set, comprising the first 70% of all observations. Hyperparameters were tuned on the validation set and the results shown here are based on the testing set, featuring the last 15% of available data.

Table 5: Comparison of Forecast Performance

	MAE	RMSE
CNN_5d	0.162	0.216
CNN_20d	0.142	0.200
CNN_60d	0.156	0.214

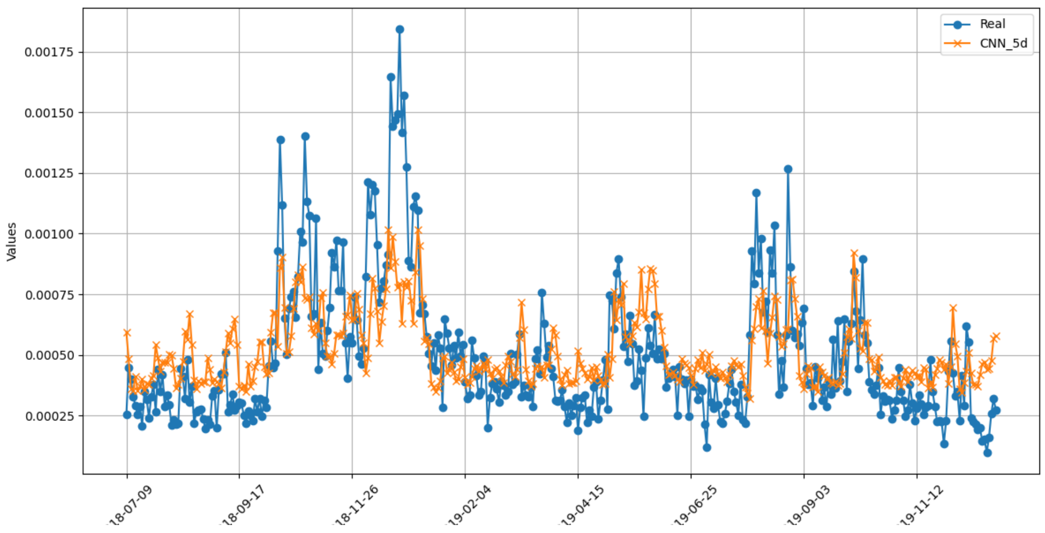
*N*ote: Values are multiplied by 1000
*S*ource: Author's computations

The table 5 shows average RMSE and MAE of the best five models within each type of the input. Both measures showed consistent outcomes. The highest precision was obtained by 20 days, followed by 60 and 5 days, respectively. Further, the winner was significantly better in both aspects, while the two other types always received similar values.

Firstly, this finding suggests that the last month approximation of 20 days might be the best predictor of the next day volatility. The trade-off between volatility clustering feature and introducing noise appear to be best resolved for 20 days, among the three types. This result is in line with the literature where the information from the last 20 days appear to be the best predictor of volatility (Ghysels et al. (2006)).

Secondly, using a longer period of time, like a quarter, on the input might yield slightly more precise forecasts than shorter interval like a week, The clustering phenomenon seems to be more dominant than the presence of noise, although this conclusion is too strong to make.

Figure 16: Comparison of 5-day Input CNN Forecast and Real data



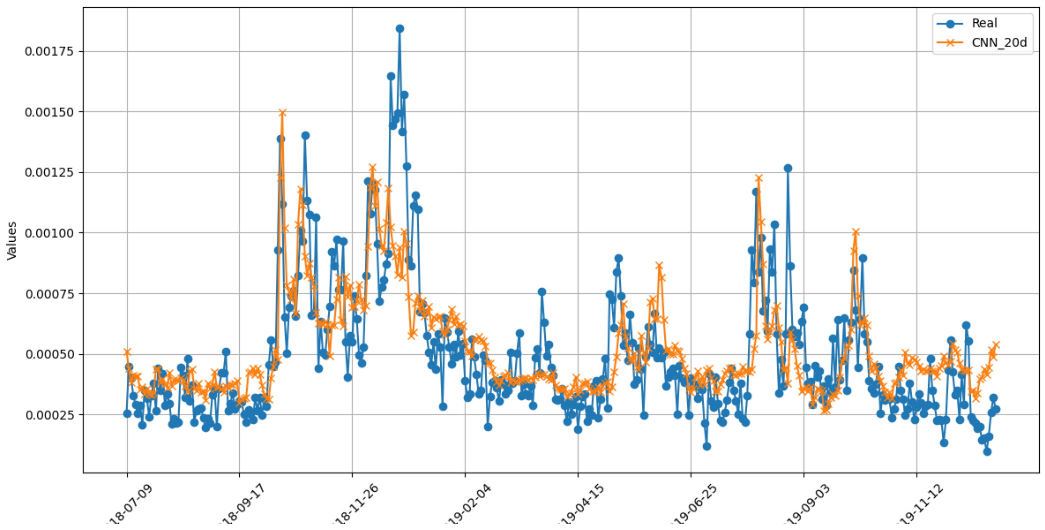
Source: Author's computations

Let us look at the results from the visual perspective. The averaged forecast values within the top five specifications for 5-day input are presented in figure 16. First of all, the values seem to be centred around the mean value and tend not to capture for the several peaks. This is a common feature of volatility forecasting for most of the models. Secondly, the one day-ahead forecasts appear to copy the real values from the overall perspective, but noticeable differences are present as well. The most apparent ones are situated at the beginning and the very end of the series when volatility remained low. On the other hand, mid-high level of volatility experienced a better matching.

Plot 17 shows the same comparison for the 20-day input. Some could argue that the results are in deed more favourable in this scenario, but the improvement does not seem to be strong. Overall, the forecast mood is very similar to the previous graph with better matching spikes.

The 60-day input should represent a quarter period beforehand. The figure 18 tells us there was no significant improvement from the previous two methods. The precision was interestingly low at the beginning but experienced an advance

Figure 17: Comparison of 20-day Input CNN Forecast and Real data



Source: Author's computations

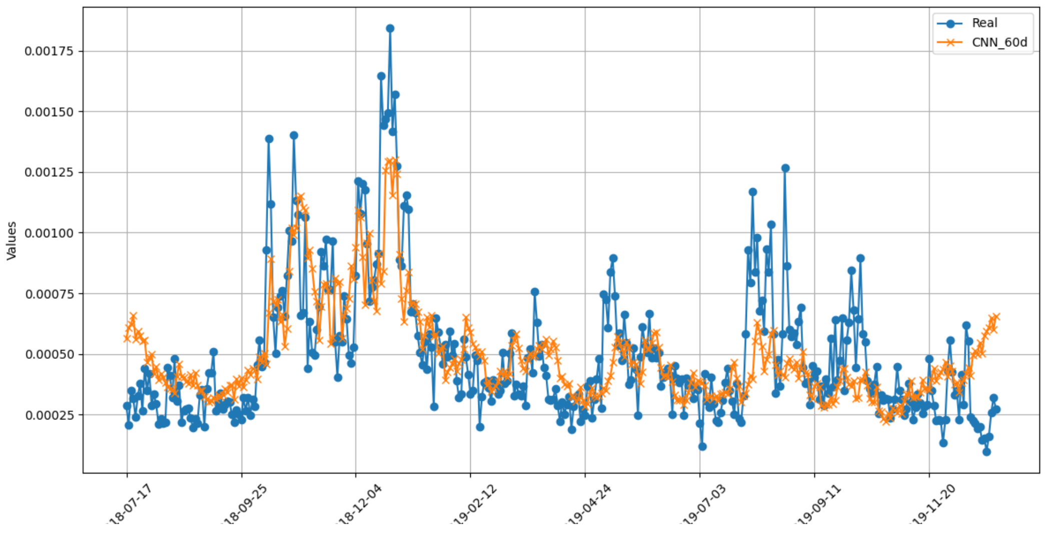
in the period of high volatility right afterwards. The rest is in similar manner to the forecasts discussed before.

In regard to the data science perspective, reasonable pieces of inference can be made about the hyperparameters. Table 6 shows the settings, in which the best five models within the three sorts differed. One can spot a pattern that more days, featuring more complex input images, required more complex models. The most apparent difference is between five days and the rest. 20 and 60 days did not show drastically distinguishable results. All other hyperparameters were identical for the models.

Table 6: Models' Specification

	5 days	20 days	60 days
Convolution Layers	1	2	2
Number of Filters	64	128	128
Batches	40	50	50
Hidden Layers	1	2	2-3

Figure 18: Comparison of 60-day Input CNN Forecast and Real data



Source: Author's computations

5.2 Comparison to Conventional Models

To put the main results into a broader perspective, we decided to fit two conventional models for forecasting volatility on the testing set and compare the results. The former is Heterogeneous Autoregressive (HAR) model. It uses three explanatory variables consisting of 1-period lag and averages of last 5 and 22 days of realised volatility.

The latter is an Autoregressive-realised volatility model (ARVol) of order (1,0,0). Both of the models' specifications are standard in the literature. We know there might exist better fitting hyperparameters, but these should align with the scope of the thesis, which is to have a regular benchmark.

The setup of the two models is identical to the CNNs. That means that we estimate the parameters using the training data and compare the accuracy on the testing set (out-of-sample). Further, the forecast values from these models are one-day ahead predictions, which is the same interval as in the case of CNN. This makes all the approaches directly comparable.

Table 7 shows how the best CNN method, which is the 20-day input, stands in comparison to HAR and ARVol. First of all, conventional methods used as the benchmark had the main difference of directly using the explained variable in

Table 7: Comparison of Forecast Accuracy

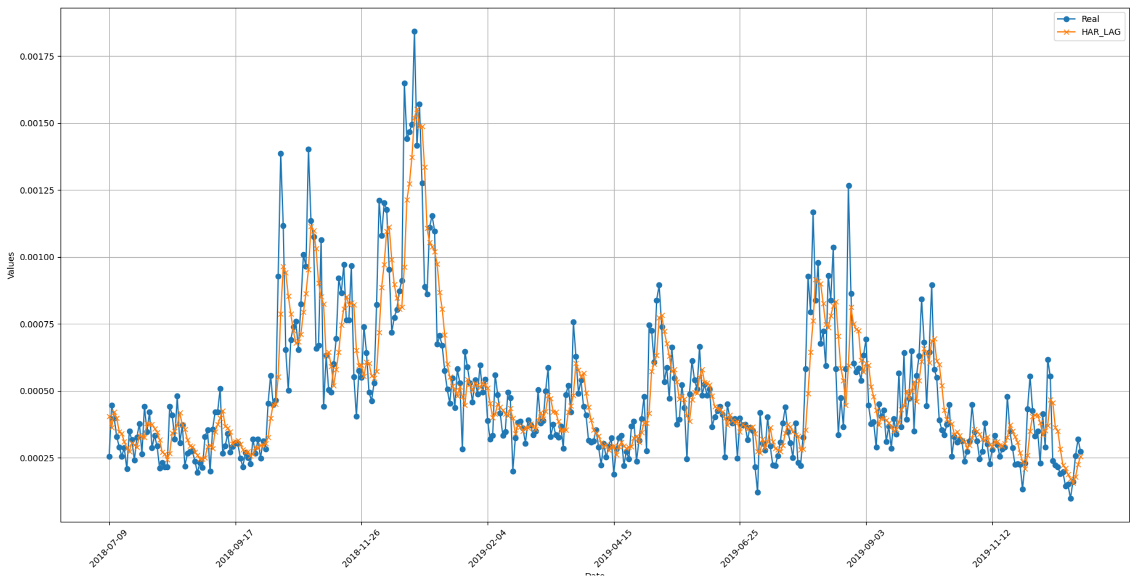
	MAE	RMSE
CNN_20d	0.142	0.200
HAR	0.111	0.138
ARVol	0.136	0.156

*N*ote: Values are multiplied by 1000
*S*ource: Author's computations

some modifications on the input as well. This means the use of the autoregressive function, which CNN did not have. It rather used plain price indicators with volume to forecast the values. The results in the table suggest that conventional models attained significantly favourable values than the best CNN predictor. Both MAE and RMSE were measurably lower in the case of HAR and ARVol than the 20-day CNN.

If we take a look at the forecast comparison in figure 19 of the HAR model and real data, it is apparent that it better predicts the one day forecast. The median values for each epochs are fitted nicely, but the peaks tend to be underestimated.

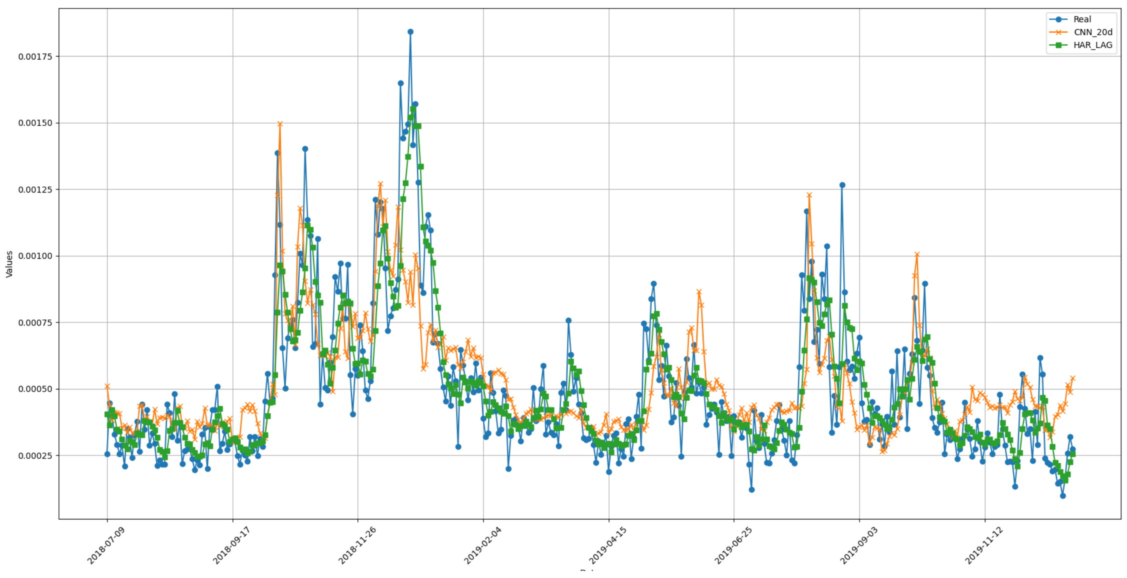
Figure 19: Differences between HAR Forecast and Real data



Source: Author's computations

Figure 20 presents a comparison of the HAR (green), CNN (orange) fits to the real values (blue). The orange colour seems to be indeed a little bit farther from the blue than the green one. Nevertheless, the period of high values at the beginning seems to be captured similarly or even better by the CNN.

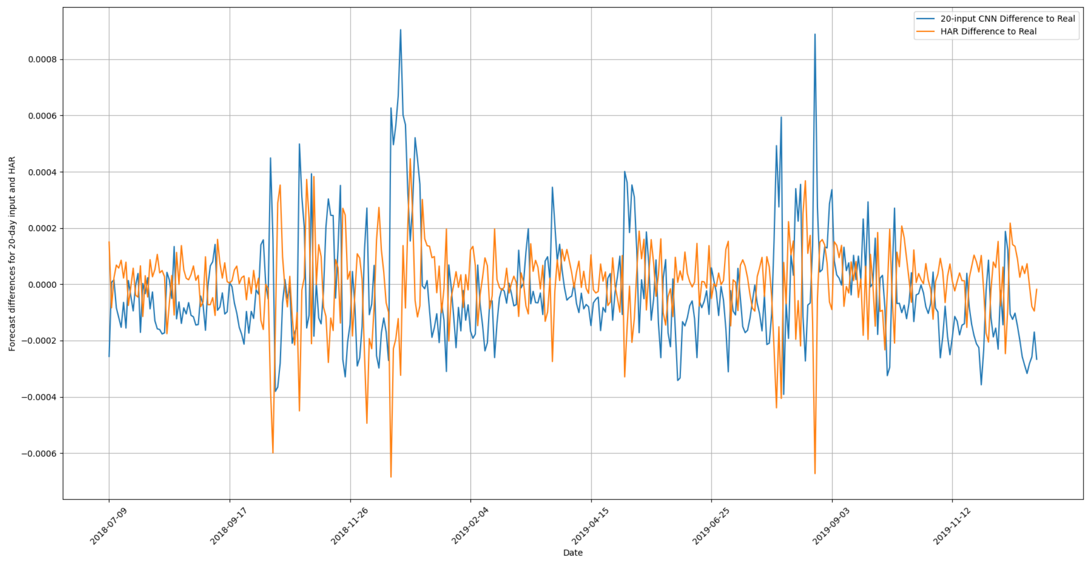
Figure 20: Differences CNN - HAR - Real data



Source: Author's computations

Last, figure 21 presents differences of both 20-day CNN and HAR to the real values. At first glance, the time series do not appear to be significantly better or worse than the other. Both contain some peaks, but remain calmly close to the zero line. Nonetheless, the end of the epoch seems to be a bit more in favour of the orange colour, representing the HAR model.

Figure 21: Differences of CNN & HAR to Real Data



Source: Author's computations

5.2.1 Rolling Window Setup of the Benchmark Models

This subsection compares the results to a more frequently used setup in volatility predicting, which is the rolling window. Analysing this method of estimation makes our CNN results directly comparable to the practical volatility forecasting models. The rolling window specification works with a 500-day interval on the input that shifts one day ahead and re-estimates itself every following step. Since HAR, ARVol and other conventional methods do not require such a long time to be fitted, this approach is indeed attainable. Unfortunately, the calculation power of current computers does not yet enable us to fit the CNN models in a similar way. Therefore, we just present the results for the purposes of practical comparison. The information is presented in table 8.

We can see that both of the errors dropped for HAR and AVOL compared to the main results. This is not illogical since the model's parameters update for the newest information available with every following step. As mentioned in the previous text, the rolling window for the CNN architecture is not feasible so far due to the extreme estimation requirements. Therefore, we cannot put the method into a face-to-face comparison, which makes us unaware of how precisely they would predict.

Table 8: Comparison of Forecast Accuracy - Rolling Window

	MAE	RMSE
CNN_20d	0.142	0.200
HAR	0.111	0.138
ARVol	0.136	0.156
HAR_roll	0.097	0.113
ARVol_roll	0.112	0.128

*N*ote: Values are multiplied by 1000
*S*ource: Author's computations

It can be just theoretised how would the convolutional neural networks stand to the rolling window setup of HAR and ARVol if it was more reachable. The rolling window for the traditional methods has shown some significant improvement to the main method of out-of-sample testing. But since the CNN did perform more poorly before, they would need to enhance by a greater margin. The key question whether that is possible thus remains unsolved until further advancements in technology take place. Graphs of the accuracy comparison are in the appendix.

5.3 Training on High vs Low-volatility Data

Further, we aimed to focus on the sensitivity of the forecast performance to the training data. All models utilise 20 day input bins because it proved best and most consistent among the three types.

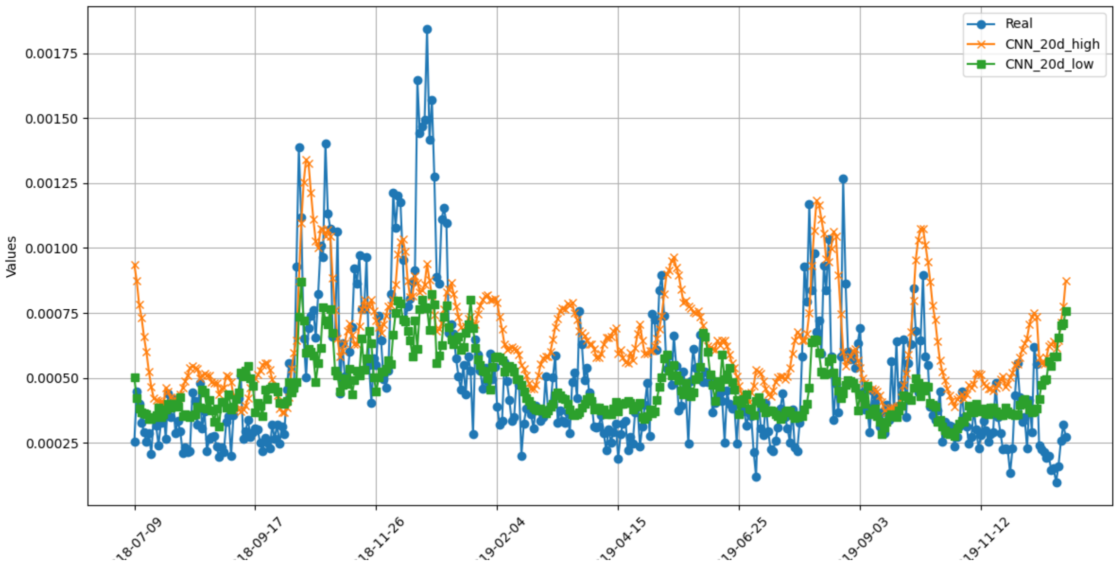
As it was introduced in the data chapter, our volatility ranges from 2010 to 2019. Since the training data capture for 70% of the interval, it corresponds to approximately 7 years. Volatility experienced different phases within this epoch and thus we decided to split it into 2 intervals. The former half encompasses higher fluctuation of volatility and the latter is in the name of constant and low trend.

Due to the fact that this low trend ended after five years, the training data reach up this point. Further on, there is the validation set, which is thus longer than before. The testing interval remained the same for comparison purposes with the initial analysis.

Since the testing data resemble a rather high-volatility behaviour, we expect the high-volatility training set to yield more favourable results. This is based on training the data on a similar set to the testing one, which might use the presence of volatility clustering.

On the contrary, low-volatility epoch is more recent to the testing data and the parameters might be of lower variance since the interval is more stable, thus uninfluenced of the up and down peaks.

Figure 22: Effect of High & Low-volatility Training Data on Test Accuracy



Source: Author's computations

Figure 22 shows average values of the top five performing models for each training set. Orange colour, representing high-volatility training data, appears to be better capturing the spikes than the regular values compared to the green line. This is something that is anticipated based on the preset conditions.

What can be deemed a bit surprising is the overall comparison of the accuracy. Table 9 presents the results with a clear difference. Low volatility training seems to overshadow the latter in both measures RMSE and MAE by a great margin. Nonetheless, it still lacks precision in comparison to the training data being the whole seven years in the main results. Row three shows the average forecast values of the low and high models. It supports the indication from the graph that training in low volatility mood yields lower values than high-volatility.

Table 9: High vs Low-Volatility Training

	CNN-low	CNN-high
RMSE	0.225	0.285
MAE	0.155	0.235
Average For. Value	0.466	0.647

*N*ote: Values are multiplied by 1000
*S*ource: Author's computations

This whole finding suggests that CNN can provide reasonable forecasts, which do not defy the logic. That is, high volatility data on training better matches high volatility on the testing set and vice-versa. Unfortunately, splitting the training dataset does not help fit the models when compared to the whole training interval of seven years.

5.4 Hypotheses Evaluation

This section discusses the overall evaluation of the hypotheses presented in the introduction.

First hypothesis states that the image-based CNN method is able to generate reasonable volatility forecast performance. Even though the approach uses prices and volume on the input, instead of the lags of the dependent variable, and does not use a rolling window, it is able to generate meaningful forecast values. The results suggest there is room for improvement in general, but the overall mood is mostly in favour of supporting this hypothesis, hence failing to reject it.

Further, the second hypothesis claims that when the number of days on the input increases, the forecast precision should rise. As the main results show, the ideal candidate for the input is 20 days. Using a 60-day period does not lead to the enhancement of the accuracy since the duration might be too long to contain only useful information. Moreover, the one week only input experiences similar results to 60 days. This suggests the optimum lies at 20 or at least close to it, forming an inverse U-shape. This hypothesis is thus rejected.

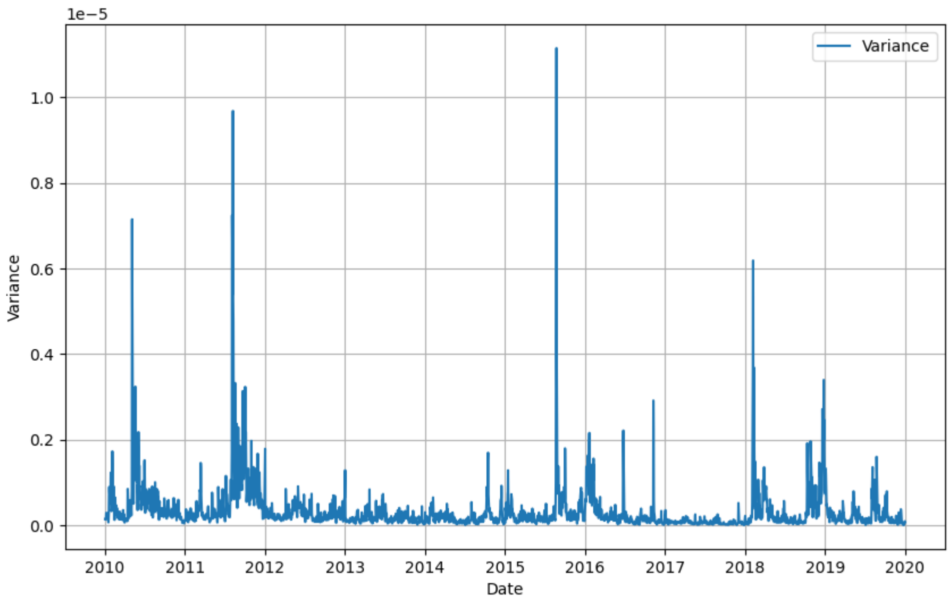
Last, the final hypothesis was that the CNN method does overshadow the traditional methods of forecasting volatility. The identical initial setup of these two types suggests the traditional methods are inclined to yield more favourable results. Nevertheless, the more practical point of view, which is the rolling window, shows the overall improvement for HAR and ARVol. Unfortunately, this cannot be set side by side to CNN since the rolling window is not computationally attainable. This hypothesis is thus rejected on the regular level of the initial setup. The rolling or possibly expanding windows might, however, alter the conclusion. We leave this topic as a motivation for subsequent studies.

5.5 Robustness Check

This chapter strives to replicate the main results while using realised variance as the output variable. The data are fetched in the same way as in the case of volatility, which results in taking simple squares of the initial values. The input remains identical as in the main process. This exploration allows us to validate the robustness of the models under different conditions, which ensures that the findings are not susceptible to a specific setup but hold true across various scenarios.

Figure 23 presents a plot of variance for the same period as volatility. Y-axis is adjusted for readability by multiplying the values by 10^5. The definition of variance makes the time-series more spread out in comparison to volatility. This increased dispersion is evident in the plot, where the mean value remains relatively stable and low, but the peaks appear more pronounced and significant. The following section shows how this affects the overall forecasting performance of CNN, HAR and ARVol.

Figure 23: Realised Variance



Source: Author's computations

5.5.1 Results

The same approach for fitting the models was conducted for variance as for volatility. That is, we used the same three sets of models' specifications to calculate the results and the best five performing models were then fetched for each type of the input bin. The average RMSE and MAE are introduced in table 10. This schema also includes the same statistics for HAR and RVol as proxies for conventional volatility/variance forecast models. The setup is managed in the same way. That is, it estimates the parameters on the training set and calculates accuracy on the testing one.

In regard to the intra-race of the three types of input, the results do not appear to dramatically differ from the volatility. CNN with 20-day input is still at the top for MAE, but experiences a slight decline in performance in RMSE compared to the 5-day input. On the other hand, 60 days have worsened significantly in both aspects. This could be explained by the amplified presence of sharp jumps in variance time-series, which might influence the longer horizon data input. This

Table 10: Comparison of Forecast Accuracy for Variance

	MAE	RMSE
CNN_5d	0.287	0.444
CNN_20d	0.250	0.474
CNN_60d	0.312	0.535
HAR	0.235	0.382
RVol	0.253	0.395

*N*ote: Values are multiplied by 10^6
*S*ource: Author's computations

could give an explanation for the 5-day input scoring better than for the volatility scenario.

Nonetheless, the overall performance for all CNN models has slightly dropped compared to the traditional techniques like HAR and RVol. Both of those have showed a measurably positive extended difference of precision to CNNs than in the case of volatility. This suggests that the images as input might be a bit more suitable for volatility predicting than for variance, where the data differences are more pronounced.

Altogether, the robustness check's results suggest that the traditional approaches have proven consistently better results in both of the output variables. Regarding the three 5, 20 and 60-day inputs, the 20 days seems to be the best choice for volatility and variance in general. The two other types showed mixed results but did not seem overshadow the 20 days.

6 Conclusion

This thesis investigates whether convolutional neural networks (CNN) featuring images of past price and volume indicators on the input are able to help predict one day-ahead volatility in financial time-series data. The idea is inspired by a paper from Jiang et al. (2020), who managed to use this new technique in predicting the future level of prices. The key question is then whether the process can be applied to the old phenomenon of volatility. Since the answer was never formulated, three distinct input setups were chosen to contain potential best framework. These include converted pictures of the last 5, 20 and 60 days, representing week, month and quarter of the year. Several specifications of the networks were analysed for each of these setups and the top five performing models were always selected. These were then compared to the benchmark traditional models of predicting volatility. The whole analysis was carried out on high-frequency data sampled at 5-minute intervals of the index E-Mini S&P 500 from the CME.

Regarding the intra-CNN performance, the 20-day input appears to be the best of the three. It has shown consistently dominant values both in MAE and RMSE. This finding is in line with the literature, where a period of the last month seems to best predict the volatility in conventional models. The other two types of input experienced similar, but worse, results in both measures, although slightly in favour of 60 days. Thus could suggest that the clustering feature of volatility may be more meaningful for a longer period despite the potential inclusion of noise.

The overall comparison to standard models of volatility showed that the CNN were not able to beat them when tested on the out-of-sample set. The reason might lie in the fact the conventional models operate with lagged values of volatility on the input, while the CNN work with prices and volume. Therefore, the specification is a bit altered and thus the models are not directly comparable. Nevertheless, the goal was to set the CNN, which is based on images representing the index behaviour, side by side to the practical models used in the literature.

Thanks to the relative shallow level of complexity, some of these conventional models can be fitted using the rolling or expanding window, which is the main and mostly used technique. This yielded even lower values of MAE and RMSE than when purely tested on the out-of-sample set. Unfortunately, the CNN are too composite and require enhanced computation power to be fitted this way as well. Thus, the direct comparison can be conducted when the technology advances. The CNN method was then evaluated on different periods of the input and subsequently on realised variance instead of realised volatility on the output. Both of these checks confirmed that the CNN does in deed predict reasonable and not illogical values of volatility. The CNN performance in the case of variance did not suggest any contradiction in relation to the comparison to standard models either.

By all counts the thesis discovered that using past prices and volume converted into images as an input to CNN could produce reasonable volatility forecast. Nevertheless, it does not overshadow the standard models used in the modern literature, which mostly hinge on the autoregressive specification. There are, however, many possible extensions of this thesis since the topic is brand new. These include different conversion of data to images or incorporating the dependent variable on the input as well. The most interesting follow up would, however, be to use a rolling or expanding window for the convolutional neural networks to update for current information.

Bibliography

Ait-Sahalia, Y., Mykland, P. A., and Zhang, L. (2005). How often to sample a continuous-time process in the presence of market microstructure noise. *The review of financial studies*, 18(2):351–416.

Alford, A. W. and Boatsman, J. R. (1995). Predicting long-term stock return volatility: Implications for accounting and valuation of equity derivatives. *Accounting Review*, pages 599–618.

Amari, S.-i. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196.

Andersen, T. G., Bollerslev, T., Diebold, F. X., and Labys, P. (2001). The distribution of realized exchange rate volatility. *Journal of the American statistical association*, 96(453):42–55.

Andersen, T. G., Davis, R. A., Kreiß, J.-P., and Mikosch, T. V. (2009). *Handbook of financial time series*. Springer Science & Business Media.

Bandi, F. M. and Russell, J. R. (2006). Separating microstructure noise from volatility. *Journal of Financial Economics*, 79(3):655–692.

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

Barndorff-Nielsen, O. E. and Shephard, N. (2006). Econometrics of testing for jumps in financial economics using bipower variation. *Journal of financial Econometrics*, 4(1):1–30.

Bollerslev, T. (2008). Glossary to arch (garch). *CREATES Research paper*, 49.

Bollerslev, T., Chou, R. Y., and Kroner, K. F. (1992). Arch modeling in finance: A review of the theory and empirical evidence. *Journal of econometrics*, 52(1-2):5–59.

Brooks, C. and Burke, S. P. (2003). Information criteria for garch model selection. *The European journal of finance*, 9(6):557–580.

Campbell, J. Y., Lo, A. W., MacKinlay, A. C., and Whitelaw, R. F. (1998). The econometrics of financial markets. *Macroeconomic Dynamics*, 2(4):559–562.

Christensen, K., Siggaard, M., and Veliyev, B. (2021). A machine learning approach to volatility forecasting. *Available at SSRN.*

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

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20:273–297.

Davidian, M. and Carroll, R. J. (1987). Variance function estimation. *Journal of the American statistical association*, 82(400):1079–1091.

Di Persio, L. and Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International journal of circuits, systems and signal processing*, 10(2016):403–413.

Ding, X., Zhang, Y., Liu, T., and Duan, J. (2015). Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence.*

Figlewski, S. (1997). Forecasting volatility. *Financial markets, institutions & instruments*, 6(1):1–88.

Frimpong, J. M. and Oteng-Abayie, E. F. (2006). Modelling and forecasting volatility of returns on the ghana stock exchange using garch models.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

Garman, M. B. and Klass, M. J. (1980). On the estimation of security price volatilities from historical data. *Journal of business*, pages 67–78.

Gavrishchaka, V. V. and Banerjee, S. (2006). Support vector machine as an efficient framework for stock market volatility forecasting. *Computational Management Science*, 3(2):147–160.

Ghysels, E., Santa-Clara, P., and Valkanov, R. (2006). Predicting volatility: getting the most out of return data sampled at different frequencies. *Journal of Econometrics*, 131(1-2):59–95.

Gunduz, H., Yaslan, Y., and Cataltepe, Z. (2017). Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems*, 137:138–148.

Hansen, P. R., Huang, Z., and Shek, H. H. (2012). Realized garch: a joint model for returns and realized measures of volatility. *Journal of Applied Econometrics*, 27(6):877–906.

Hautsch, N. and Podolskij, M. (2013). Preaveraging-based estimation of quadratic variation in the presence of noise and jumps: theory, implementation, and empirical evidence. *Journal of Business & Economic Statistics*, 31(2):165–183.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hoseinzade, E. and Haratizadeh, S. (2019). Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285.

Huang, Z., Liu, H., and Wang, T. (2016). Modeling long memory volatility using realized measures of volatility: A realized har garch model. *Economic Modelling*, 52:812–821.

Jeong, J. (2019). The most intuitive and easiest guide for convolutional neural network. *Towards Science.*

Jiang, J., Kelly, B. T., and Xiu, D. (2020). (re-) imag (in) ing price trends. *Chicago Booth Research Paper*, (21-01).

Kambouroudis, D. S., McMillan, D. G., and Tsakou, K. (2016). Forecasting stock return volatility: A comparison of garch, implied volatility, and realized volatility models. *Journal of Futures Markets*, 36(12):1127–1163.

Koyuncu, H. (2020). Loss function selection in nn based classifiers: Try-outs with a novel method. In *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6. IEEE.

Krogh, A. and Hertz, J. (1991). A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lek, S. and Park, Y. (2008). Artificial neural networks. In *Encyclopedia of Ecology, Five-Volume Set*, pages 237–245. Elsevier Inc.

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

Maciel, L., Ballini, R., and Gomide, F. (2016). Evolving possibilistic fuzzy modeling for realized volatility forecasting with jumps. *IEEE Transactions on Fuzzy Systems*, 25(2):302–314.

McKenzie, M. D. (1999). Power transformation and forecasting the magnitude of exchange rate changes. *International Journal of Forecasting*, 15(1):49–55.

McMillan, D., Speight, A., and Apgwilym, O. (2000). Forecasting uk stock market volatility. *Applied Financial Economics*, 10(4):435–448.

Meddahi, N. and Renault, E. (2004). Temporal aggregation of volatility models. *Journal of Econometrics*, 119(2):355–379.

Miron, D. and Tudor, C. (2010). Asymmetric conditional volatility models: Empirical estimation and comparison of forecasting accuracy. *Romanian Journal of Economic Forecasting*, 13(3):74–92.

Moreira, M. and Fiesler, E. (1995). Neural networks with adaptive learning rate and momentum terms.

Nagyfi, R. (2018). The differences between Artificial and Biological Neural Networks — towardsdatascience.com. `https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7`. [Accessed 07-03-2024].

Neufeld, D. (2023). The $109 trillion global stock market in one chart. *Visual Capitalist*.

Nguyen, K. H. (2022). Anatomy, Head and Neck: Eye Retina. https://www.ncbi.nlm.nih.gov/books/NBK542332/.

O'shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Parkinson, M. (1980). The extreme value method for estimating the variance of the rate of return. *Journal of business*, pages 61–65.

Patton, A. J. (2006). Modelling asymmetric exchange rate dependence. *International economic review*, 47(2):527–556.

Patton, A. J. (2011). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, 160(1):246–256.

Petrozziello, A., Troiano, L., Serra, A., Jordanov, I., Storti, G., Tagliaferri, R., and La Rocca, M. (2022). Deep learning for volatility forecasting in asset management. *Soft Computing*, 26(17):8553–8574.

Poon, S.-H. and Granger, C. W. J. (2003). Forecasting volatility in financial markets: A review. *Journal of economic literature*, 41(2):478–539.

Pratiwi, H., Windarto, A. P., Susliansyah, S., Aria, R. R., Susilowati, S., Rahayu, L. K., Fitriani, Y., Merdekawati, A., and Rahadjeng, I. R. (2020). Sigmoid activation function in selecting the best model of artificial neural networks. In *Journal of Physics: Conference Series*, volume 1471, page 012010. IOP Publishing.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Rastogi, S., Don, J., and Nithya, V. (2018). Volatility estimation using garch family of models: Comparison with option pricing. *Pacific Business Review International*, 10(8):54–60.

Rogers, L. C. G. and Satchell, S. E. (1991). Estimating variance from high, low and closing prices. *The Annals of Applied Probability*, pages 504–512.

Ross, S. A. (1977). The capital asset pricing model (capm), short-sale restrictions and related issues. *The Journal of Finance*, 32(1):177–183.

Sheta, A. F., Ahmed, S. E. M., and Faris, H. (2015). A comparison between regression, artificial neural networks and support vector machines for predicting stock market index. *Soft Computing*, 7(8):2.

Tran, H. and Cutkosky, A. (2022). Better sgd using second-order momentum. *Advances in Neural Information Processing Systems*, 35:3530–3541.

Tyuleubekov, S. (2019). Can model combination improve volatility forecasting?

Udacity (2024). Deep learning pytorch. `https://github.com/udacity/deep-learning-v2-pytorch`. Accessed on March 29, 2024.

Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495.

Wythoff, B. J. (1993). Backpropagation neural networks: a tutorial. *Chemometrics and Intelligent Laboratory Systems*, 18(2):115–155.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer.

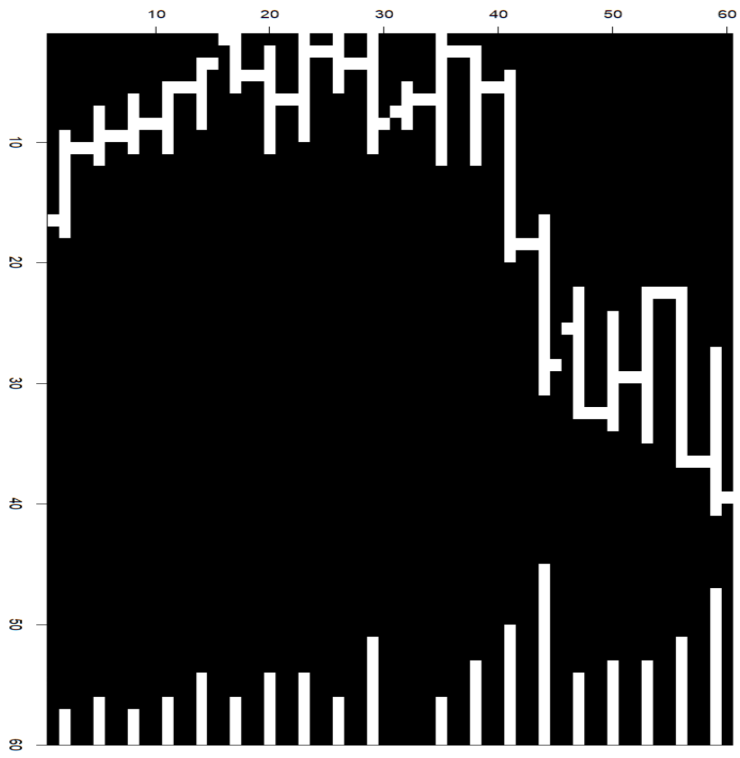
Zhang, Q. (2018). Convolutional neural networks. In *Proceedings of the 3rd International Conference on Electromechanical Control Technology and Transportation*, pages 434–439.

Zupan, J. (1994). Introduction to artificial neural network (ann) methods: what they are and how to use them. *Acta Chimica Slovenica*, 41(3):327.

Appendices

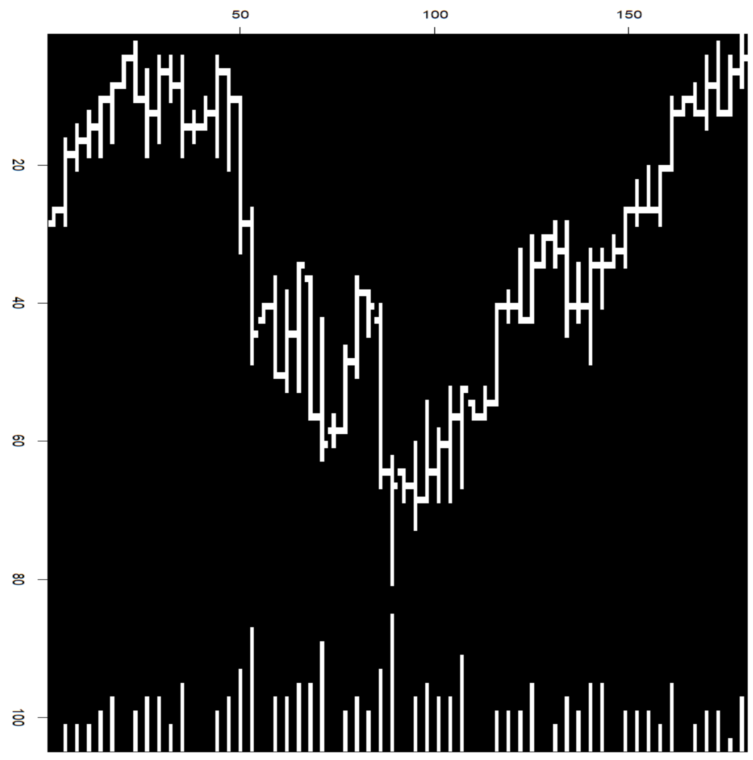
Appendix A Transformed Input Matrices - 20 & 60 Days

Figure 24: Input Data 20 Days - Transformed Matrix



Source: Author's computations

Figure 25: Input Data 60 Days - Transformed Matrix



Source: Author's computations

Appendix B Extended Configuration of CNNs

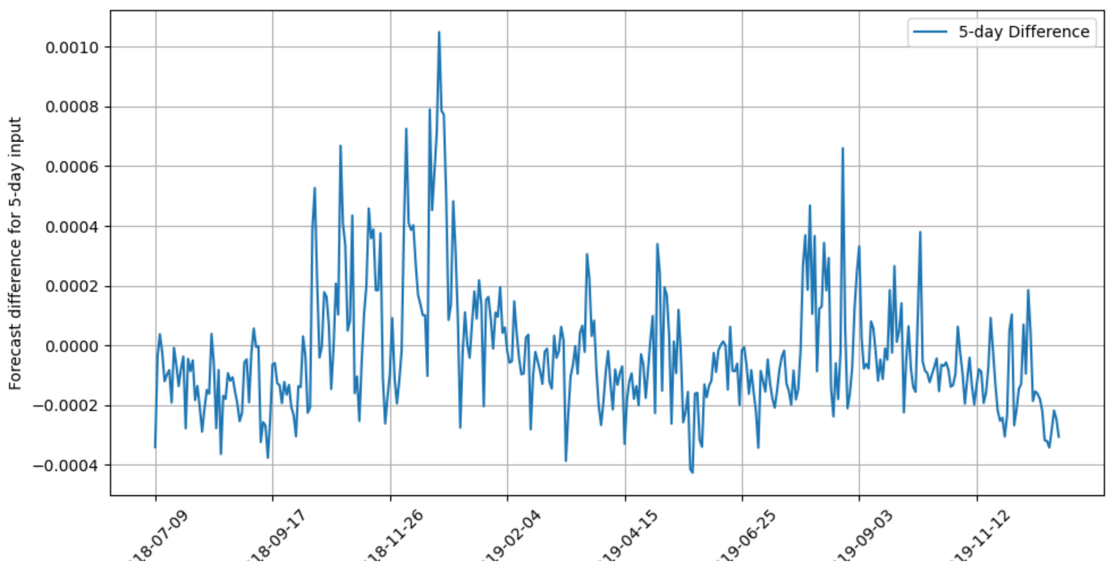
Table 11: CNN Configuration

Parameters	**5 days**	**20 days**	**60 days**
Conv. Layer with Relu Activation Function	✓	✓	✓
Max Pooling (2x2)	✓	✓	✓
Conv. Layer with Relu Activation Function		✓	✓
Max Pooling (2x2)		✓	✓
Flatten Layer	✓	✓	✓
# of Hidden Layers - ReLU Activation Function	1	2	2-3
Output Layer - Sigmoid Function	✓	✓	✓
# of Filters	64	128	128
Convolution Layers	1	2	2
Batches	40	50	50
Optimiser	AdamW	AdamW	AdamW
Learning Rate	0.001	0.001	0.001
Loss Function	MSE	MSE	MSE

*S*ource: Author's computations

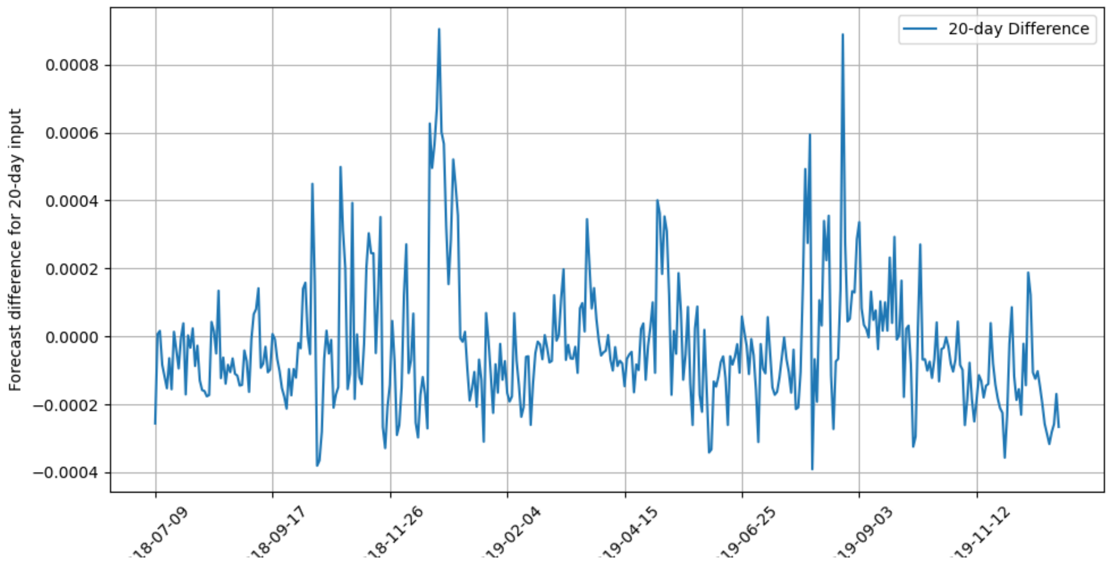
Appendix C Extended Results for CNN Performance

Figure 26: Differences between 5-day Input CNN Forecast and Real data



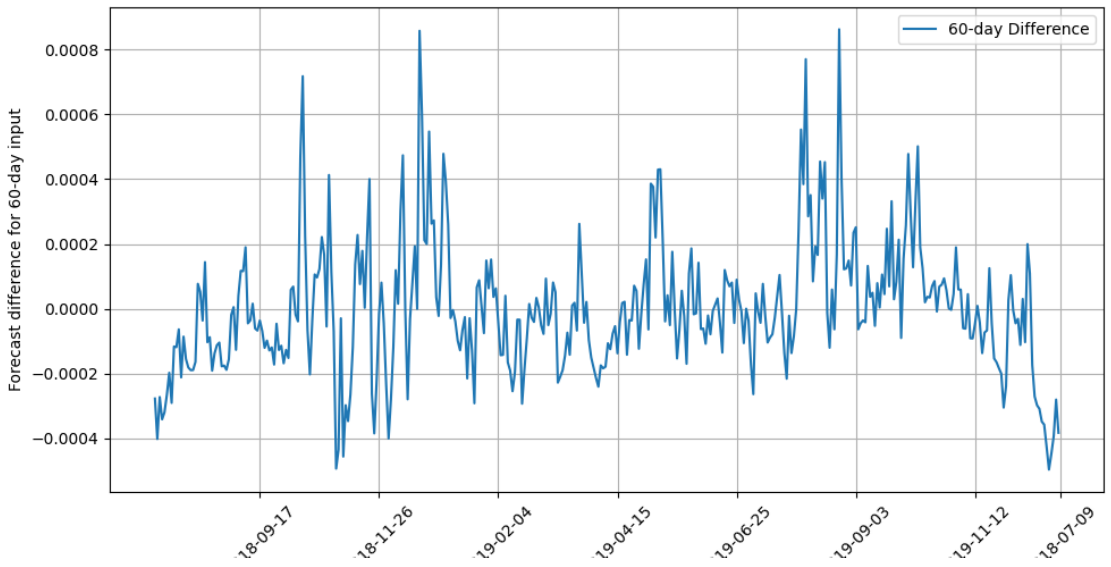
Source: Author's computations

Figure 27: Differences between 20-day Input CNN Forecast and Real data



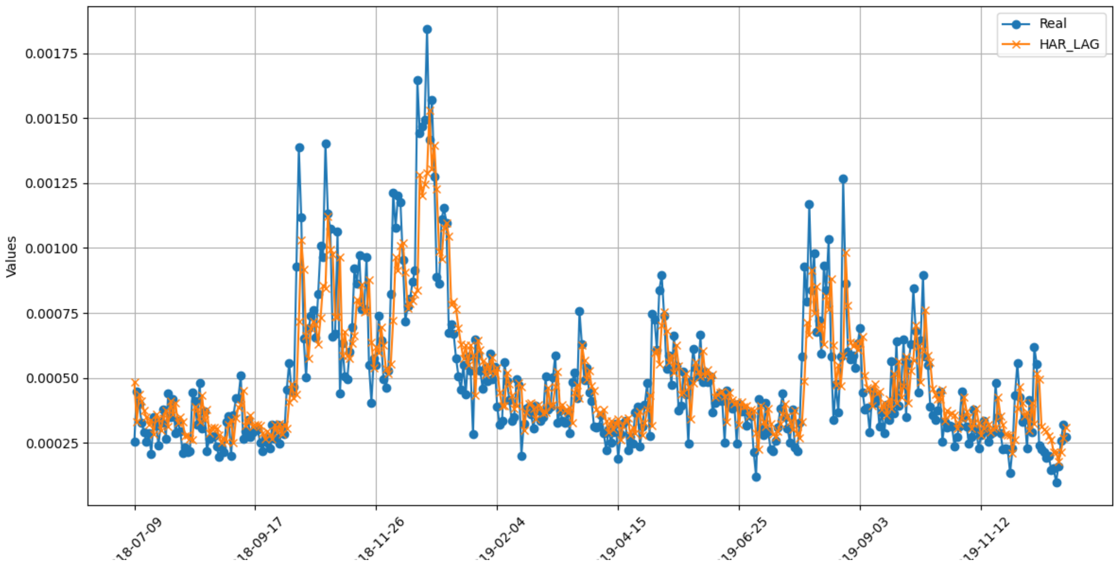
Source: Author's computations

Figure 28: Differences between 60-day Input CNN Forecast and Real data



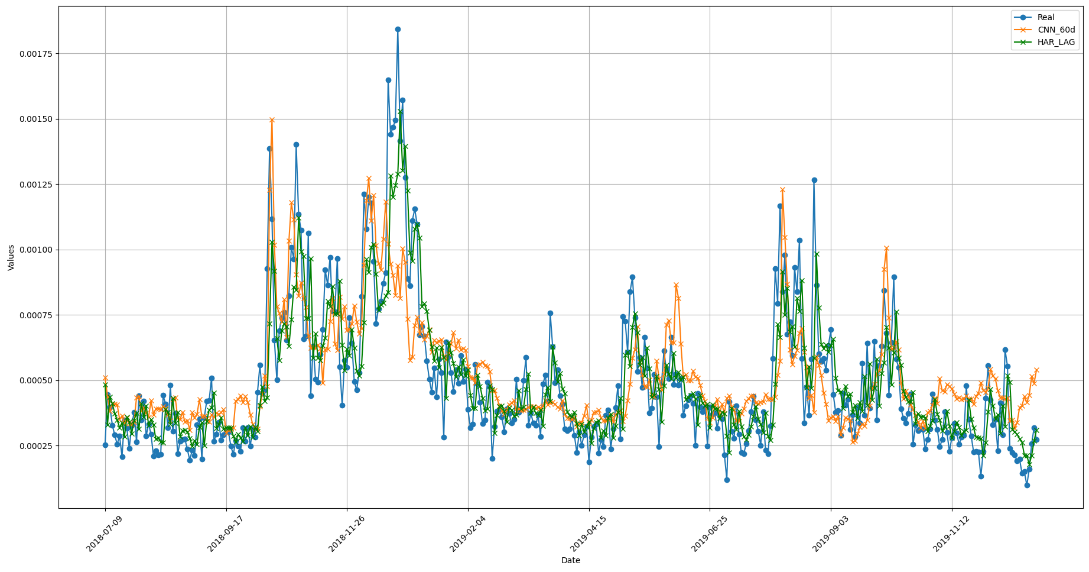
Source: Author's computations

Figure 29: Differences between HAR Forecast and Real data - Rolling Window



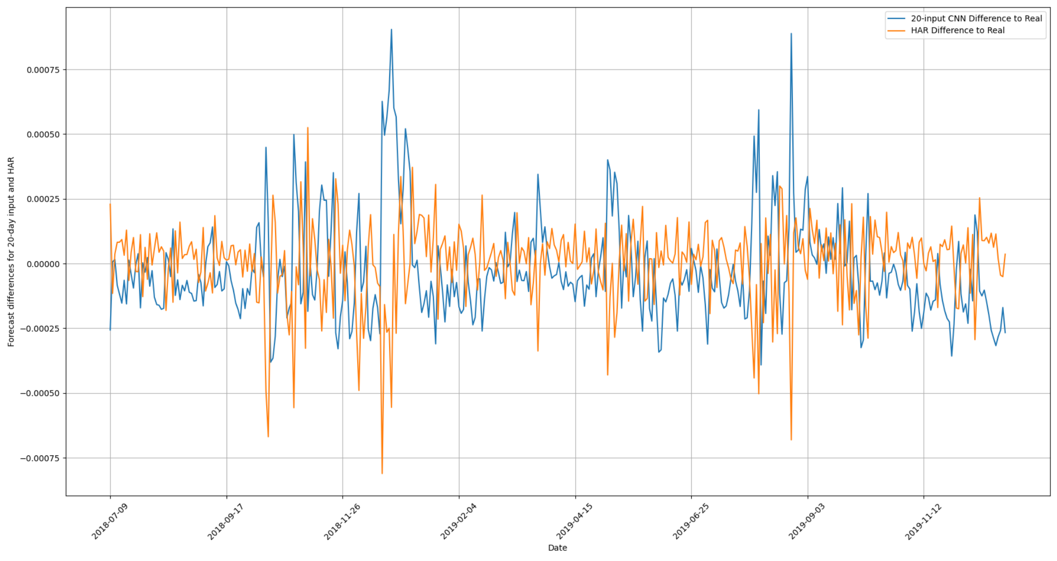
Source: Author's computations

Figure 30: Differences CNN - HAR - Real Data - Rolling Window



Source: Author's computations

Figure 31: Differences of CNN & HAR to Real Data - Rolling Window



Source: Author's computations