A Work Project, presented as part of the requirements for the award of a Master's degree in Finance from the Nova School of Business and Economics.

# FORECASTING FINANCIAL ASSET PRICE MOVEMENTS USING CONVOLUTIONAL NEURAL NETWORKS – APPLICATION TO THE U.S. FINANCIAL SERVICES SECTOR AND COMPARISON ACROSS INDUSTRIES

43949 – Jonathan Boße

Work project carried out under the supervision of:

Patrícia Xufre Gonçalves da Silva Casqueiro

17-12-2021

**Abstract:**

This thesis explores the applicability of CNNs as a price movement forecasting tool for ETFs, using a technical analysis approach and three different image encoding techniques. After developing a general methodology, the thesis focuses on the application to the U.S. financial services sector. Subsequently, the research draws comparisons to results obtained for other U.S. sector ETFs using the same model approach.

Overall results show that the CNN models, while proving some potential and exceeding a random model in accuracy, show significant weaknesses for all industries in predicting Buy and Sell signals. Addressing these weaknesses, limitations of the approach are explored to suggest methods for model performance improvements.

**Key words:**

Trading, Technical Analysis, Convolutional Neural Networks, Industry Comparison, Gramian Angular Fields, Markov Transition Fields, Financial Time Series Forecasting, Financial Services Sector

## Table of Content

## List of Figures

## List of Tables

## List of Abbreviations

| Abbreviation | Meaning |
|---|---|
| ADF | Augmented Dickey-Fuller test |
| ANN | Artificial Neural Network |
| B&H | Buy & Hold |
| CNN | Convolutional Neural Network |
| ETF | Exchange Traded Funds |
| FTH (method) | Fixed time-horizon method |
| GAF | Gramian Angular Fields |
| GADF | Gramian Angular Differentiation Field |
| GASF | Gramian Angular Summation Field |
| MTF | Markov Transition Fields |
| NN | Neural Network |
| PCA | Principal Component Analysis |
| PXL (method) | Sezer's feature pixelation (method) |
| RF | Random Forest |
| SVR | Support Vector Regression |
| TB (method) | Triple-barrier (method) |

# 1 Introduction

For technical traders, i.e. practitioners of technical analysis, image analysis plays a vital role in their day-to-day decision-making, given that many decision are based on patterns and trends that can be observed in the stock charts (Drakopoulou 2015, 4). However, when looking at how algorithmic trading, i.e. trading supported by computational resources, is done in practice, one can see very little use of image recognition; instead, other algorithmic trading techniques are primarily in use. Due to various factors, such as the emergence of significantly better hardware and new computational approaches, the last 10 to 15 years have seen critical advances in Deep Learning, especially recently in the field of image recognition and analysis using convolutional neural networks (CNNs). CNNs have proven to be increasingly good at recognising and distinguishing objects.

Thus, a critical question that needs to be asked is how these advances can be leveraged as applications to trading, simulating the trader's decision process based on image analysis with the help of CNNs. There has already been research on the application of CNNs to forecasting stock price movements, however, within a limited scope. The objective of this paper is to apply CNNs to different industries to determine whether there are differences in the performance and usability of CNNs used for stock price predictions across various industries

For this purpose, image recognition with CNNs will be applied to the following six industries and comparisons be made:

- Information Technology
- Healthcare
- Industrials
- Energy
- Oil & gas
- Financial Services

To achieve a high degree of representativeness for each sector and reduce idiosyncratic factors inherent to individual companies, industry ETFs or indices consisting of a large variety of companies will be used as assets to forecast on, instead of using individual company shares. Moreover, only ETFs or indices covering the U.S. market will be used to increase comparability across the industries, avoiding differences in geographic factors as much as possible.

The paper is structured in the following way:

Firstly, an introduction to trading and stock analysis approaches is given to provide context on how CNNs fit into the scope of stock analysis and time-series forecasting.

Secondly, a high-level introduction to CNNs will be given, and the general methodology used in this paper will be explained.

The third part focuses on applying an established methodology to the specific industries, respective adjustments to the methods to account for particular characteristics of the industries and the results obtained for each sector.

In the fourth, the best-performing hyperparameters as well as model performances across the different industries will be compared and discussed and conclusions on the added value of the application of CNNs to price movement forecasts will be drawn.

The fifth focuses on the comparison of the established performance measures across the different industries.

In the sixth and last part, cross-sectoral limitations of the methodology are faced and an outlook on potential further research topics is provided.

## 2 Trading and Time-Series Forecasting

The following section will provide a brief introduction to trading and its two essential stock analysis approaches and a high-level overview of time-series forecasting methods, in order to place CNNs in the context of trading and price forecasting.

### 2.1 Trading

There are several types of trading that can be distinguished based on factors such as the frequency of executed trades, the period of an asset and the underlying method used to determine which assets to buy and sell (Banton 2021). However, regardless of the trading type they are applying, traders have the common key objective of maximising their profits. Traditionally, the most common groups of traders are so-called technical and fundamental traders, based on the stock analysis approach they use: technical and fundamental analysis, the most important general analysis tools in the realm of investing and trading (Petrusheva and Jordanoski 2016, 30). They represent two approaches to determining what shares investors should buy or sell to maximise their profit. Technical analysis also gives indications on the optimal time to execute the transaction (Petrusheva and Jordanoski 2016, 31). Although their overall objective is identical, they differ significantly in the assumptions they are based on, the methods they employ and the time horizons for which they are used (Petrusheva and Jordanoski 2016, 30). While **fundamental analysis** focuses on the economic forces of supply and demand that cause prices to change (Murphy 1999, 5) and aims at determining the fair value of corporate securities by studying company-specific key value-drivers, so-called fundamentals, such as a company's earnings, its risks factors, growth rates and competitive positioning (Lev and Thiagarajan 1993, 190), **technical analysis** focuses solely on the share price and trading volumes as the two key determinants to forecast future price developments (Petrusheva and Jordanoski 2016, 28).

The main premise of fundamental analysis is that each asset has a fair value that it will always converge to in the long run, but it may not always reflect this fair value due to temporary mispricing in the markets (Lev and Thiagarajan 1993, 191). The fair value can be determined by an investor through the analysis of the underlying fundamentals, such as the company's financial statements, the overall economic state of the markets the company operates in as well as developments of the industry the company belongs to. An investor can then generate profits by identifying mispriced assets, capitalising on the eventual price corrections that will take place in the market according to the basic premise of fundamental analysis (Abad, Thore and Laffarga 2004, 231).

The core belief of technical analysis, on the other hand, is that all factors affecting the stock price (fundamentals, political factors, environmental factors, etc.) are already reflected in the price of that stock, which results in the reasoning that only price and volume data need to be analysed to forecast future price movements (Murphy 1999, 2).

A second and third concept essential to technical analysis are the assumptions that prices move in trends and that history repeats itself (Murphy 1999, 2). With these two assumptions in place, an investor can take investment decisions based on patterns that worked well in the past (history repeats itself) and can generate profits by identifying trends in early stages of their development to trade in accordance with the direction of these trends (Murphy 1999, 3).

Regarding the time horizons for which the two methods are used, it can be stated that fundamental analysis commonly uses longer periods when analysing the underlying data and is mostly used for longer-term investment decisions, and as such, is often used by investors focusing on value investing (Petrusheva and Jordanoski 2016, 27). Technical analysis, on the other hand, focuses stronger on short-term data (price and volume data for single a day, few days or few weeks) and is often used for the identification of assets that can be traded to generate profits in the short term, i.e., stocks whose prices will experience significant changes in the near future (Petrusheva and Jordanoski 2016, 28).

Fama's Efficient Market Theory (1970) states that none of the investment analysis approaches will allow an investor to generate returns that exceed the market return, given that any new information entering the market will be immediately included in the asset price. Following this statement, technical analysis, i.e. forecasting future price movements based on past price developments, will not generate excess returns above the market. This paper will analyse to which degree the Efficient Market Theory holds true when applying CNNs to the general technical analysis approach, given that they are potentially able to recognise patterns that traditional technical analysis methods miss.

### 2.2 Introduction Financial Time Series Forecasting

While technical and fundamental analysis have traditionally been the two most widely used approaches to stock price forecasting, emerging technologies have opened up new possibilities to stock price analysis, a type of data that is difficult to predict as financial markets are volatile, representing non-linear, fluctuating, and high noise data (Thakkar & Chaudhari 2021, 1). The use of machine learning and deep learning approaches has gained increasing attention due to their ability to detect localised data features at multiple levels. This trend also opens new possibilities for investment strategies and changes the nature of investing. Relying on deep learning for investment makes trading and investment decisions more rational than investment decisions based on human knowledge and experience, with the latter tending to result in more subjective and biased decisions (Yang et al. 2019, 387). Different forecasting types which might be of prediction interest include either the movement direction of the stock market to predict local extreme values or turning points to recognise the perfect point to either sell or buy (classification problem) or the magnitude of change of the market movement including future prices (regression problem) (Peng et al. 2021, 10).

Before the rise of deep learning applications for financial problems, conservative statistical methods were used. The logistic regression as one popular classification model provides an easy understanding and interpretation of the results. However, these traditional statistic models assume linearity – thus, representing a crucial limitation (Peng et al. 2021, 14).

Deep Artificial neural networks as linear models with pieces of nonlinearity bypass these problems by permitting the learning of more abstract knowledge representations. Nonetheless, by working with more complex structures and hence more features, they are more prone to overfitting. (Peng et al. 2021, 15).

Extensive research has been conducted about possible other approaches for making predictions in trading. Among others, popular approaches include Artificial Neural Networks (ANNs), Support Vector Regressions (SVRs), Logistic regressions and Decision Trees (Huang et al. 2019, 134). Examples of extensive research conducted in this area can be found in several research papers. An overview is presented in Table 1.

Even though all these approaches seem promising, CNN's have a big advantage: They are able to work well with data having a spatial relationship (Brownlee 2018). A necessary requirement to fulfill is the transformation of data into images before being able to make predictions though, as information is retrieved via multi-scale localized spatial features (Chen et al. 2021, 69) (Xu et al. 2015). They have proven themselves to be highly successful for stock predictions, as stock data can be illustrated as a 2D matrix (Chen and He 2018).

| Authors | Goal | Approach | Main Results |
|---|---|---|---|
| Moghaddam and Esfandyari (2016) | Predict daily NASDAQ stock exchange returns | ANN | $R^2$ values above 0.9 |
| Nayak et al. (2016, 441 et sqq.) | Predict daily and monthly movements of the stock (whether they go up or down) | Decision Boosted Tree | Outperformed a SVM and a Logistic Regression Model |
| Henrique et al. (2018, 183) | Predict stock prices from different markets | Support Vector Regression | Performed especially well for market periods with lower market volatility and for a strategy with updating the model periodically |
| Patel et al. (2015, 2171) | Predict Indian Stock market indices | Two-stage fusion approach between ANNs, Random Forest Models and SVRs combined to hybrid models: SVR–ANN, SVR–RF and SVR–SVR. They were afterwards compared to single models | Results of this study have shown ANNs and RFs to better perform in a hybrid model including SVRs rather than as single models. The best overall performance was shown by the SVR-ANN model |
| Vijh et al. (2020, 605) | Forecast next day stock closing prices | Random Forests and an ANN | They indicate strong results. Overall, in this case, the ANN performed better than the RF |

Table 1 - Overview Financial Time Series Research

Source: Own illustration

Within the last years, different approaches to financial time series forecasting with CNNs have been addressed. Cohen, Balch, and Veloso (2020) have created various charts based on open, high, low, and closing prices to forecast trading signals using a CNN. The results demonstrate that the transformation of the time series into images is beneficial for the recognition of trading signals. Sezer and Ozbayoglu (2018) on the other hand create images based on 15 technical indicators over a period of 15 days (15x15 image). Using these images and a CNN-TA architecture, the

research team was able to forecast entry and exit points (Buy, Hold, Sell) comparatively better than with other models. Arratia and Sepúlveda (2020) make use of recurrence plots and data of 12-month periods to predict the direction of the S&P 500 the following month. Their CNN model attains an accuracy of 63 percent. The most promising and cited methods were proposed by Wang and Oates (2015). They used Gramian Angular Fields and Markov Transition Fields to transform time series into images and ran a tiled CNN for classification. Due to the promising results, the method was adapted and further developed in other research papers.

### 2.3 Technical Analysis with CNNs

While there has already been research on the applications of CNNs to stock price prediction, a status review shows that there is still hardly any practical use of this approach. This paper will focus on expanding the state of current research, evaluating if there are differences across industries in terms of computational and financial performance of investment strategies based on CNNs. Before going into details on CNNs and the applied methodology, it is important to understand why CNNs are highly applicable to technical analysis. There are two key factors making the combination of technical analysis with the usage of convolutional neural networks an attractive investment research topic: Firstly, the assumption that no knowledge about factors and trends affecting the markets is necessary as they are already included in the price (Murphy 1999, 4). Technicians know that there are many reasons why markets move, but do not assume it necessary to know these reasons in the forecasting process (Murphy 1999, 4). Based on these assumptions, it is sufficient to use visual representations (such as charts) of past price movements as a base to predict future price developments. Consequently, it appears reasonable to use CNNs to analyse the information contained in these visual representations without having to include further external information that might be difficult to represent in an appropriate visual input for a CNN.

Secondly, experienced technicians increasingly take intuitive decisions based on the patterns they see in the charts (Murphy 1999, 6). They learn to intuitively recognise the meaning of a variety

of patterns, i.e., what price movements tend to be preceded by what type of patterns in the charts. Seen from a high level, CNN's have a very similar approach to learning. Through different layers within the neural network, a CNN learns to recognise patterns in the images it is trained on, giving it the tools to make inferences from these patterns to the classification of that image, in order to be able to classify unknown images. Thus, it seems reasonable to assume that a CNN can be trained to predict future price movements based on patterns in past data in the same way that a human technician would.

# 3 Fundamentals and Methodology

This chapter provides the theoretical and methodological basis for the thesis. First, an understanding of the concepts of neural networks and convolutional neural networks is given. Then, several preprocessing methods are considered, and an overview of the generic model architecture and its evaluation methods are presented. The approach in this chapter is to outline widely established perspectives regarding the concepts presented in the current research. It is continuously reasoned which methodology is used for this work. Definitions that are appropriate for this thesis are also provided.

## 3.1 Introduction to CNNs

The following section provides an introduction to the deep learning algorithms used in this work. The terminology related to neural and convolutional neural networks and their essential structure are described. The associated components are presented to provide a deeper understanding of how the systems operate.

### 3.1.1 Definitions

Definition Neural Network

Neural networks (NNs) are 'computerised intelligent systems' (Thakkar and Chaudhari 2021, 2) that aim to recognise patterns and learn relationships in data by simulating the signal exchange between biological neurons in the human brain. A neural network consists of different layers of artificial neurons, also called **units**, which are interconnected and can be divided into input units, hidden units, and output units (Kröse and Van der Smagt 1993, 15). A set of **input units** receives information and applies certain weights, which are translated into an output by the network through an activation function (Kröse and Van der Smagt 1993, 15). **Output units** signal how the network reacts to the learned and processed information. Between input and output units there are one or more layers of **hidden units**, which perform nonlinear transformations of the

inputs (Kröse and Van der Smagt 1993, 15). A neural network is considered **fully connected** if each hidden unit is connected to each unit in the layers on both sides of the network. Supervised neural networks learn continuously through a feedback process called **backpropagation** (Chollet 2017, 11). In this iterative process, the actual output is compared to the expected output of the network. The difference is used to adjust the weights between the units in the network, that is, the strength of the connections, so that inputs match the correct output (Chollet 2017, 52). Neural networks continuously learn and improve with examples enabling it to respond accordingly to an entirely new set of inputs. They are particularly popular when modeling highly nonlinear systems or when unexpected changes in input data may occur. Many applications have employed neural networks to simulate unknown relationships between various parameters based on a vast set of examples. Classifications of handwritten digits, speech recognition, and stock price prediction are examples of effective neural network applications (Keijsers 2010).

Neural networks are usually divided into artificial neural network (ANN) and deep neural network (DNN). A deep neural network is a type of artificial neural network, with multiple hidden layers between the input and output layers (Thakkar and Chaudhari 2021, 2). The increasing volumes of structured and unstructured data cause deep learning systems, i.e., neural networks with many layers, to become increasingly popular.

Definition Convolutional Neural Network

According to Dertat (2017), convolutional neural networks (CNN) are the most popular type of deep neural networks. They are mainly applied in pattern and image recognition problems since they are specifically designed to process pixel data (Sezer and Ozbayoglu 2018). However, they are also useful for natural language processing and prediction purposes. A convolutional neural network comprises five types of layers: **input**, **convolution**, **pooling**, **fully connected**, and **output** layers. Each layer serves a specific purpose and is explained in more detail in Section 3.1.2.

CNNs are generally considered superior to regular NNs due to their automatic feature selection strategy. Using CNNs, it is now possible to build larger models to solve more complex problems, which was infeasible with conventional NNs (Albawi, Mohammed, and Al-Zawi 2017, 1). Their deep learning structure with multiple hidden layers allows them to abstract a larger number of features (Dertat 2017). By analysing the data in greater detail, a higher accuracy of the output can be achieved. The automatic feature extraction of CNNs, achieved by mapping input data to output, is especially useful for extracting complex patterns from non-linear data (Thakkar and Chaudhari 2021, 2). This property is particularly relevant for stock market predictions, since stock-based data is highly complex and non-linear (Thakkar and Chaudhari, 2021, 2,7). A CNN uses convolution to learn the local features of the image, and thus manages to preserve the local connectivity or spatial relationships between pixels, making them particularly suitable for extracting relevant information at low computational cost (Arratia and Sepúlveda, 2020).

### 3.1.2 Key Components of CNNs

Convolutional layer

The convolutional layers are the most important building block in a CNN. Mathematically, convolution refers to an integration function that indicates the amount of overlap of a function shifting over another function. In other words, the convolution describes filters that slide horizontally and vertically over the input array (our picture) and calculate the dot product at each taken step. In this context, the filter, also called kernel, refers to a set of weights, usually a 3*3 matrix, that extracts features (Chollet 2018, 127-128). The so-called stride describes the step size, with which the filter slides over the picture, meaning that increasing the stride will result in a lower-dimensional output (Ghosh et al. 2020, 8). The output of the convolution is a feature map which stores information about the occurrence of features in a matrix along with how well it complements the kernel. In Figure 1 the convolution operation is demonstrated. In this example a 3*3 filter is applied on a

6*6 input array with stride equaling one which results in a 4*4 feature map. Applying zero-padding, i.e., padding the input array with zeros, can be used to further control the size of the output array (O'Shea and Nash 2015, 7).

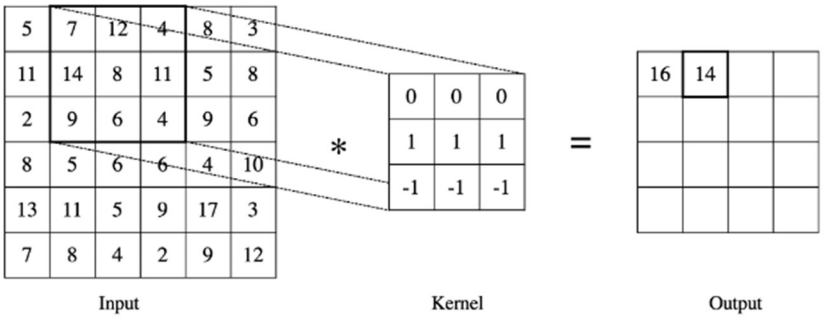


Figure 1 Illustration of the Convolution Operation.

Source: Own illustration

The CNN can contain one or more convolutional layers, each of them allowing through filters to identify local patterns, which can later be recognised all over unseen pictures. The filters behave similarly to the human eye and learn patterns hierarchically. The deeper the convolution layer, i.e., the more convolutional layers applied, the more detailed and higher-level features can be extracted from the image (Tsai, Chen, and Wang 2018, 942).

Pooling Layer

The pooling layer has the purpose to reduce the dimensionality of the convolved feature map. This reduces the number of features and the complexity of the model while persevering the most dominant features. For the pooling operation a kernel, usually of dimensionality 2*2, slides over the feature maps and applies a pooling technique. The most used pooling technique is max pooling, meaning to extract the maximum value for each window. Similar to the convolutional layer, the stride size can be adapted. In the pooling layer the usual stride size is two (Chollet 2018, 127). An example of the max pooling operation with a 2*2 window and stride two is shown in Figure 2.

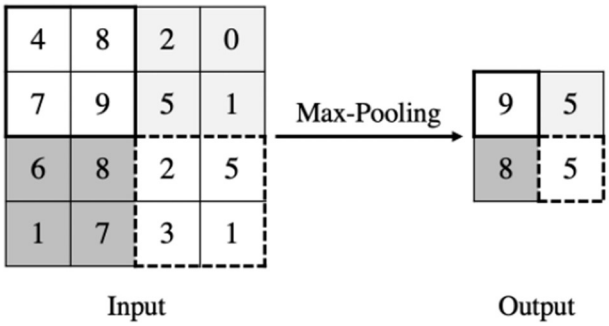


Figure 2 Exemplary Max Pooling Operation

Source: Own illustration

Fully connected layer

Before the created feature can be fed to a fully connected layer, the outputs of the final convolution or pooling operation are flattened. The following fully-connected layer is analogue to a simple feed-forward ANN, meaning that each neuron in this layer is connected with each neuron in the adjacent layers (Ghosh et al. 2020, 9). This step is essential to allow the model to generalise local patterns. The output of the fully connected layer is a representation of the likelihood of an input belonging to a certain class.

Descriptions of hyperparameters used for the CNN in this paper can be found in section 3.6.

### 3.2 Labelling Approach

To train the CNN, labelled training images are required. The approach used in this project opts to frame the predictions as a multi-class classification instead of a regression (i.e., predicting continuous return values). The three classes used to label observations in this project are Buy (label = 1), Sell (label = -1) and Hold (label = 0), based on the price movement during the period after the observation. There are two general labelling approaches in the context of stock price forecasts suggested by different papers: the fixed time-horizon method and the triple-barrier method (Lopez de Prado 2018, 43-48)

### 3.2.1 Fixed Time-Horizon Method

The **fixed time-horizon method** (hereafter called FTH method) is the more commonly used one due to its simplicity (Lopez de Prado 2018, 43). Its basic premise is to compare the price of an asset at the end of an observation period to an upper and lower threshold h that had been previously set. The thresholds are set as relative values to the price at the beginning of the consideration period, e.g., 10% above and below the closing price of the previous period (Lopez de Prado 2018, 43).

$$Y_{i,t} = \begin{cases} Sell \text{ if } p_{i,close\ t+1} \leq (1-h) * p_{i,close\ t} \\ Hold \text{ if } (1-h) * p_{i,close\ t} < p_{i,close\ t+1} < (1+h) * p_{i,close\ t} \\ Buy \text{ if } p_{i,close\ t+1} \geq (1+h) * p_{i,close\ t} \end{cases} \quad (1)$$

The FTH method is straightforward and easy to implement. As it requires relatively little amounts of data, especially compared to the triple-barrier method, it is very suitable for labelling observations from long-term datasets, for which historic high-frequency data, e.g., price on a per-hour base, are not or only very limitedly available. However, the FTH method shows an essential shortcoming: the fixed threshold used in this method does not consider the volatility of the underlying asset and only considers the value of the asset at the end of the observation period, but not during the period; as such, it is unrealistic in practice, since it implicitly assumes that investors would only implement a transaction at the end of the consideration period. In reality, an investor can implement a transaction at any time during trading hours. Moreover, an investor will set limits beforehand based on the volatility (i.e., inherent risk) of an asset. Furthermore, in practice, investment strategies usually have stop-loss limits (i.e., bottom limits) and profit taking targets (e.g., sell when 10% return target is hit) at which they would exit a position as soon as the limit is met. As such, a more realistic labelling approach needs to consider price movements during the consideration period as well as the asset's underlying volatility.

### 3.2.2 Triple-Barrier Method

The **triple-barrier method** (hereafter called TB method) takes into account intra-period price movements and the asset's volatility and solves the main shortcoming of the FTH method (Lopez de Prado 2018, 45). The TB method sets three barriers:

- Two horizontal barriers, representing the profit-taking and stop-loss boundaries. The horizontal barriers are dynamic functions of the estimated volatility experienced by the analysed asset and the limit approach set for the investment.
- One vertical barrier, representing the end of the observation period.

To construct the barriers, upper and lower multipliers need to be set. These multipliers depend on return targets an investor is setting (upper multiplier) and their degree of risk aversion, i.e. the maximum loss they are willing to incur before exiting the position (lower multiplier) and can thus be different across different types of investors. For simplicity, symmetric multipliers of (1 , 1) will be used in this paper.

In the TB method, using a multi-class classification approach with three labels (Buy, Hold, Sell), an observation is labelled based on the first of the three barriers it touches (Lopez de Prado 2018, 45):

- Y = Buy: The observation is labelled as Buy if the upper horizontal barrier is touched first. This means that the asset's price hits the profit-taking target during the consideration period t, and thus, the asset should be bought in period t-1 to realise a positive return.
- Y = Hold: The observation is labelled as Hold if neither the upper nor the lower horizontal barriers are hit. This implies that the asset's price hits neither the profit-taking target nor is stopped out by the stop-loss limit. Thus, it means no transaction is made. Depending on the context of the investment, this either implies not investing (neither long nor short) or holding the asset (in case the asset had already been previously bought).

- Y = Sell: The observation is labelled as Sell if the lower horizontal barrier is touched first. In this case, the asset's price hits the stop-loss limit first and is stopped out. Thus, the asset should be sold in t-1. Depending on the context this implies to either sell the asset to avoid losses or to short-sell to generate a positive return through a shorting strategy.

The TB method is more realistic due to its consideration of intra-period price movements and volatility, but requires significantly more data (Lopez de Prado 2018, 46). This can pose a challenge when analysing long-term data for which higher-frequency data is not sufficiently available. The project described in this paper faces the challenge that it aims at predicting the respective next day's price movements. As such, using the TB method would require intra-day price data to determine which barrier is hit first. However, this intra-day price data could not be obtained for the entire period that is being analysed in this project. To achieve consistency in the labelling approach across the entire data set, a simplified version of the TB method will be applied.

### 3.2.3 Simplified Triple-Barrier Method

The upper and lower horizontal limits will be constructed in the same way as in the normal TB method, with factors *h* calculated based on asset's volatility and the chosen multiplier. However, instead of comparing intra-day price data to the two horizontal limits to create labels on a per-day base, high and low prices will be compared to the limits. The labelling approach is as follows:

(2)

$$Y_{i,t} = \begin{cases} \text{Sell if } p_{i,\,\text{low } t+1} \leq (1\text{-h})*p_{i,\,\text{close } t} \\ \text{Hold if } p_{i,\,\text{low } t+1} > (1\text{-h})*p_{i,\,\text{close } t} \text{ and } p_{i,\,\text{high } t+1} < (1+\text{h})*p_{i,\,\text{close } t} \\ \text{Buy if } p_{i,\,\text{high } t+1} \geq (1+\text{h})*p_{i,\,\text{close } t} \end{cases}$$

- Y = Buy: An observation on day t will be labelled as Buy if the high price on the following day t+1 is higher than or equal to the upper limit at t+1.
- Y = Sell: An observation on day t will be labelled as 2 if the low price on the following day t+1 is lower than or equal to the lower limit at t+1.

- Y = Hold: Should none of the limits be exceeded on day t+1, the observation on day t will be labelled as 0.

This labelling approach assumes that the investor is willing to hold the asset as long as necessary to hit one of the barriers, such that time will have no impact on the position, as long as none of the barriers are hit.

A limitation of this labelling approach is that it is unable to consider a time dimension and thus, the issue of double labelling might arise in case that both conditions are met, i.e. the high price lies above the upper limit and the low price lies below the lower limit. Therefore, when implementing the methodology across the individual industries, the percentage of double labels will be controlled and alternative measures taken should this percentage be above a threshold of 2%.

### 3.3 Feature Engineering

Feature Engineering is essential to improve Machine Learning or AI models. In the following all pre-processing steps are explained and the reasoning for the applied methodologies provided.

#### 3.3.1 Feature Creation

**Technical Analysis** is confined to the analysis of trends and movements in the market (Yang et al. 2019). These indicators are used to predict future stock movements.

In principle, a distinction is made between two categories of technical indicators: leading and lagging indicators. **Leading indicators** lead the price movement as they attempt to predict the trend in a time series (Fernández-Blanco et al. 2008, 1851). **Lagging indicators** are trend-following indicators that provide delayed feedback as they lag the market (Bogullu, Dagli, and Enke 2002, 722).

Indicators from both categories belong to one of four following types of technical indicators (Salkar et al. 2021, 2).

1. **Trend indicators** show the direction in which the market is moving along with the strength of the trend by comparing historical prices to a baseline (Salkar et al. 2021, 2).

They typically move between low and high values. The trend can be either downward (bearish), upward (bullish), or sideways (no clear direction) (Peachavanish 2016, 2).

2. **Momentum indicators** assess the speed of price fluctuations in a time series by comparing current and previous closing prices (Salkar et al. 2021, 2).
3. **Volatility indicators** measure the speed of price movement and provide information on how much the price changes in a given period (Salkar et al. 2021, 2).
4. **Volume indicators** measures the number of shares traded in a stock and thus provide an indication of the strength of the market (Salkar et al. 2021, 2).

The use of technical analysis indicators as input features for neural network systems is established in research (Arratia and Sepúlveda 2020; Sezer, Ozbayoglu, and Dogdu 2017; Sezer and Ozbayoglu 2018; Sim, Kim, and Ahn 2019; Thakkar and Chaudhari 2021). The selection of technical indicators was primarily based on their frequency in related studies as analyzed in literature (Chen et al. 2021, 69; Peng et al. 2021, 5–6; Sezer and Ozbayoglu 2018, 529). In this paper, two trend and seven momentum indicators are combined with different parameter settings. Most technical indicators possess a user defined **window** width as input, affecting the indicators output (Shynkevich et al. 2017, 72). The window size typically refers to the number of raw observations or periods processed by the indicator (Shynkevich et al. 2017, 72). The higher the window width, the more data will be processed. For the two trend indicators, i.e., the moving averages, three different window sizes were chosen respectively. For the seven momentum indicators, one set of parameters was chosen for each. A total of 13 technical indicators are calculated based on the closing price of the used ETF. Table 1 provides an overview of the selected technical indicators. Definitions and calculations for each indicator can be found in Appendix A.

| Technical Indicator | Type | | Number of features | Parameters:<br>n = number of periods processed by the indicator. |
|---|---|---|---|---|
| | Trend | Momentum | | |
| Simple moving average (SMA) | x | | 3 | n = {5, 10, 20} |
| Exponential moving average (EMA) | x | | 3 | n = {5, 10, 20} |
| Rate of change (ROC) | | x | 1 | n = 12 |
| Percentage Price Oscillator (PPO) | | x | 1 | $n_{long} = 26$<br>$n_{short} = 12$ |
| Relative Strength Index (RSI) | | x | 1 | n = 14 |
| Know Sure Thing Oscillator (KST) | | x | 1 | As defined in Appendix A. |
| Williams % Range | | x | 1 | n = 14 |
| Moving Average Convergence Divergence (MACD) | | x | 1 | $n_{long} = 26$<br>$n_{short} = 12$ |
| Commodity Channel Index (CCI) | | x | 1 | n = 20 |

Table 2 - Technical Indicators and their Parameter Settings

Source: Own illustration

Along with the technical indicators, a set of additional variables is included in the set of predictors for the convolutional neural network. Those include the high, low, opening and closing prices along with the volume traded of the respective ETF, the closing prices of S&P 500, gold, and oil futures as well as the exchange rate of Euro and U.S. Dollar.

### 3.3.2 Stationarity

When using financial time series, it is common to ensure stationarity as non-stationary time series usually hamper modelling its behaviour (Hyndman und Athanasopoulos 2018). When data are non-stationary, their characteristics, i.e. mean and variance, can change over time, impede the prediction of future values.

To evaluate which variables lack stationarity, the Augmented Dickey-Fuller test (ADF) will be used, one of the most common methods to statistically test for non-stationarity. ADF tests the existence or absence of a unit root. A unit root test can be mathematically represented as

$$y_t = D_t + z_t + \varepsilon_t \tag{3}$$

with $D_t$ representing the deterministic, $z_t$ the stochastic component and $\varepsilon_t$ the stationary error (Verma 2021). The ADF test removes autocorrelation from the time series before testing for stationarity in contrast to the Dickey-Fuller test. The ADF can be represented as

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p} + \varepsilon_t \tag{4}$$

where $\alpha$ denotes a constant, $\beta$ the coefficient over time and $p$ the order of the lag. The null hypothesis, $\gamma = 0$, is tested against the alternative hypothesis of $\gamma > 0$. The test statistic value

$$DF_\tau = \frac{\hat{\gamma}}{SE(\hat{\gamma})} \tag{5}$$

is then compared to the critical value of the ADF test. A 95 percent level is chosen, corresponding to a $DF_\tau$ statistic of -2.86 (Cheung and Lai 1995, 277-279).

In case of non-stationarity **fractional differenciation** will be applied. Unlike integer differencing, a method that simply subtracts a previous value from the current day (Hyndman und Athanasopoulos 2018), fractional differencing finds the optimal balance between zero and maximum differentiation to guarantee stationarity while preserving the maximum amount of memory in the data (Lopez de Prado 2018, 84). More precisely, it ensures that the mean and variance of the time series do not change with time while a high correlation with the original series is maintained. A feature on a current day can be expressed as the sum of all previous days with an assigned weight for each value. The weight is calculated by the fractional derivative. For this purpose, a transformation method is applied that automatically finds the minimum order of fractional differentiation and turns the time-series stationary. Walasek and Gajda (2021) applied fractional differencing to

stock prices before training an ANN model. They confirmed improved performance of the model on stationary data as opposed to non-stationary data.

### 3.3.3 Feature Selection

**Feature Selection** plays a crucial role in the creation of successful prediction models, identifying a final selection of relevant variables (Speiser et al. 2019, 94). If the right features are chosen, it improves the overall prediction performance while reducing computational costs and diminishing the complexity of the model.

Especially the progressive application of Machine Learning and Artificial Intelligence in the field of trading is a driving force for the collection of enormous amounts of data. Special attention should be paid to strongly correlated features (Peng et al. 2021, 5). The creation of technical analysis indicators may lead to highly correlated variables, representing redundant information (Haq et al. 2021, 2). After creating a variety of financial indicators with different parameters in our approach, a special emphasis should be placed on an efficient feature selection approach to avoid this problem of multicollinearity and overfitting (Peng et al. 2021, 10).

Therefore, Principal Component Analysis (PCA) is applied to reduce the features' multicollinearity and thus the dimensionality of the dataset while preserving most of its information. This is achieved by identifying the principal components which are representing new variables as linear combinations of the original features (Rahoma, Imtiaz, and Ahmed 2021, 2).

Mathematically spoken, the eigenvectors and eigenvalues are computed based on the covariance matrix of the feature set, such that $Av = \lambda v$. In this formula A denotes the covariance matrix, v the eigenvector, and $\lambda$ the eigenvalue. The computed eigenvectors describe the direction of the explained variance whereas the eigenvalues express how much variance is captured in the respective component. The components are created such that the first principal component explains the highest percentage of the variance and each additional component captures less information. (Tharwat 2016)

In this work the amount of variance that needs to be explained by the model will be set to 95 percent. This threshold represents a trade-off between capturing as much information of the dataset as possible, and reducing the number of components in order to minimise the computing costs to train the convolutional neural network. Since the algorithm penalises lower variance features, it is necessary to standardise the features before applying PCA (Abdi and Williams 2010, 2).

### 3.5 Image Construction

One of the most common image construction methods used for times series forecasting with CNN's is the transformation of data into Gramian Angular Fields, as proposed by Wang and Oates (2015). The research team proposed another image encoding methodology, called Markov Transition Fields, which will be used in this paper as well.

#### 3.5.1 Gramian Angular Fields

To leverage the advantages of CNNs in the context of trading, the timeseries data must be encoded to images. One approach to this are Gramian Angular Fields (GAFs). GAFs are capturing spectral correlation structures, thus being able to capture temporal dependencies, representing time series in a two-dimensional way. To create a GAF, the first step required is the rescaling of the data points of a time series $X = \{x_1, x_2, \dots, x_n\}$ to a normalisation range of [-1, 1] (Yang et al. 2019, 189).

$$\tilde{x}_i = \frac{(x_i - \max(X)) + (x_i - \min(X))}{\max(X) - \min(X)} \quad (6)$$

GAFs are not using the cartesian coordinate system. Instead, the normalised time series is converted to polar coordinates by computing the angular cosine of the scaled time series. This representation shows the value at a certain timestamp, holding N timestamps t with a value of x. The conducted pairing is of bijective nature, mapping a value represented by the angle uniquely to a point in time, shown by the radius r (Barra et al. 2020, 685).

$$\begin{bmatrix} \phi_i = \arccos(\tilde{x}_i)\,, & \tilde{x}_i \epsilon\, \tilde{X} \\ r_i = \dfrac{i}{N}\,, & with\ t_i\ \epsilon\ \mathbb{N} \end{bmatrix}$$

(7)

After this transformation, the trigonometric sum between the values of the time series in the set is conducted to obtain the correlation (Romero et al. 2020, 16692). Two approaches can be used for turning the vectors into a symmetric Gramian matrix: either the Gramian Angular Summation Field (GASF) or Gramian Angular Differentiation Field (GADF) (Yang et al. 2019, 190). The main diagonal of this final matrix holds the original spectral values. As time moves, the image position moves from the top left to the bottom right corner, representing the time dependencies (Liu et al. 2022, 4).

$$GASF = \left[\cos(\phi_i + \phi_j)\right] = \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2}$$

$$GADF = \left[\sin(\phi_i - \phi_j)\right] = \sqrt{I - \tilde{X}^2}' \cdot \tilde{X} - \tilde{X}' \cdot \sqrt{I - \tilde{X}^2}$$

(8)

(Formula: Yang et al. 2019, 190)

The aggregation of separate GAFs into one image has already been researched. Yang et al. cover this novel approach in their study by stacking images together to feed into the CNN as one (Yang et al. 2019, 190). This aggregation approach raises the question whether the order of images influences the performance of the model. Yang et al. reject this hypothesis by conducting experiments, discovering that the sequence of arrangement has no impact on the results (Yang et al. 2019, 191).

### 3.5.2 Markov Transition Fields

As a third method to transform the dataset into images, Markov Transition Fields (MTFs) will be used – also presented by Wang and Oats in 2015. With this method, information can be preserved in the time sphere of the different features used. As for the Gramian Angular Fields, data from the previous 10 days are used as a reference point for classification.

Given a variable as a time series X, first, the Q quantile bins of the variable will be identified and each value $x_i$ is assigned to one of the bins ($q_i \in [1, Q]$). In a next step, a weighted adjacency matrix W of size a Q * Q is created by counting the conversions of the bins among the time axis conforming to a first order Markov chain. Each value in the Matrix W describes the frequency of a point in a certain quantile which occurs one period after a point in another quantile. The matrix W is normalised such that the sum of each value in the matrix equals one. The values do now present the probability by which one value of a quantile is followed by another value of a specific quantile. (Wang and Oats 2015, 42)

When construting the images for our classification task at hand, a n*n (n refers to the time periods used for each feature image) matrix is created as following based on the weights defined previously (Wang and Oats 2015, 42).

$$M_{i,j} = \begin{bmatrix} w_{ij|x_1 \epsilon q_i, x_1 \epsilon q_j} & \cdots & w_{ij|x_1 \epsilon q_i, x_n \epsilon q_j} \\ w_{ij|x_2 \epsilon q_i, x_1 \epsilon q_j} & \cdots & w_{ij|x_2 \epsilon q_i, x_n \epsilon q_j} \\ \cdots & \cdots & \cdots \\ w_{ij|x_n \epsilon q_i, x_1 \epsilon q_j} & \cdots & w_{ij|x_n \epsilon q_i, x_n \epsilon q_j} \end{bmatrix} \quad (9)$$

For each point in time and each feature a Markov Transition Matrix is calculated. All features matrices of one time stamp are then stacked, similar to the approach used for the GAFs, before fed into the CNN.

### 3.6 Generic Model Architecture

Data set splitting and cross validation for time-series data

An important focus when developing any machine learning model is the generalisation of the model, i.e. how well it deals with data it has not been trained on (Bergmeir and Benítez 2012, 197). To evaluate the performance of a model on unknown data, parts of the available data set will be held back as validation and test sets, such that the model will not be trained on all available data. This produces two problems: firstly, the model would most likely show a better performance if trained on the full data set, and secondly, by just evaluating the performance on sample, this performance measurement might not be representative of the true model performance. To solve these problems, in most cases k-fold cross-validation will be used for training and performance evaluation. All available data is randomly split into k sets. The model training and performance evaluation is carried k times, where every set is used once as the test set, and the other sets being used for model training. This way, the method produces k independent performance measurements, while all available data is used for both training and testing. By averaging the performance measurement across the k iterations, a relatively robust measurement can be obtained, which is much more representative of the true model performance than a single measurement (Bergmeir and Benítez 2012, 197).

However, the standard k-fold cross-validation cannot be applied to time-series data. The data set cannot be split at random into training and validation sets as there is no sense to using data from the future to forecast data from the past (Herman-Safar 2021). In other words, the temporal dependency betweens data points needs to be preserved during training and testing. A solution to this is Rolling Forward Cross-Validation, also referred to as Time Series Split Cross-Validation. The data set is split into k consecutive subsets, while preserving the continuity of the data, i.e. the data set is not split at random, but based on its temporal order. Then, rolling forward cross-validation method will iterate consecutively over the k subsets. In the first iteration, the first subset will be used for training and the second one for validation. In the second iteration, the first subsets

will be used for training and the third one for validation. These iterations continue until the first k-1 subsets are used for training and the k-th subset for validation (Herman-Safar 2021).

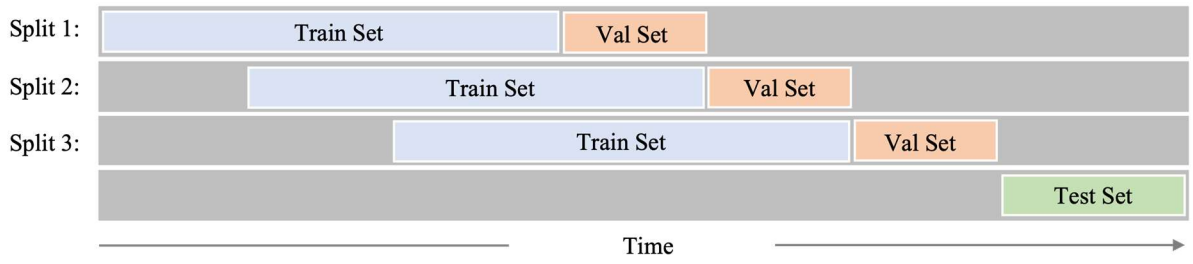


Figure 3 Rolling Forward Cross-Validation

Source: Own illustration

The described cross validation approach is applied to find the best model architecture with the respective optimal hyper-parameters as specified below. After estimating the best model, the chosen model is evaluated with the test set. To retain the temporal dependencies, the test set constitutes consecutive data points like the validation sets used for the cross validation. This test set includes 20% of all data, accounting for approximately the last two years of data.

Model Architecture

As a Convolutional Neural Network this paper proposes a rather simple CNN architecture as displayed in Figure 4. This basic architecture includes the input layer, two convolutional layers with 64 and 128 filters, one pooling layer, one fully connected layer as well as one output layer.

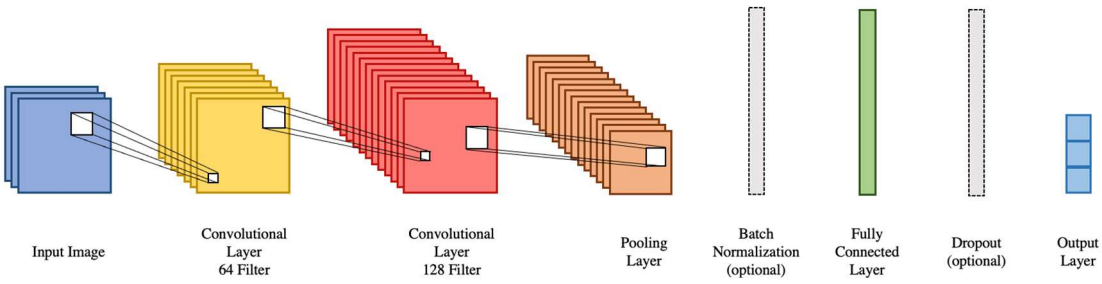


Figure 4 Model Architecture

Source: Own illustration

In order to make the network more flexible to adapt to different ETFs and industries, a hyperparameter search is added. Since a gridsearch would be computationally too expensive, a randomized hyperparameter search is utilized. The search includes an optional dropout layer and batch normalization layer. Regarding the convolutional operation different hyperparameter settings for the kernel size, the activation function (output layer exluded due to multiclass classification problem softmax is used in each model) and padding are included. For the pooling operation a parameter to control the type of pooling, either max or average pooling, is used. Lastly, the optimizer, learning rate, batchsize, the number of epochs and whether class weights should be introduced are included in the randomized search (Table 3). The following section explains the parameters in more detail.

| **Category** | **Hyperparameter** | **Parameter distribution** |
|---|---|---|
| Additional Layer | Batch Normalization | include; exclude |
| | Drop Out (incl. Rate) | exclude; include with rate 0.25; include with rate 0.5 |
| Convolution | Kernel Size | 3*3;  5*5 |
| | Activation Function | relu; sigmoid; softmax |
| | Padding | same; valid |
| Pooling | Pooling Type | max pooling, average pooling |
| Compilation | Optimizer | Adam; RMSprop; SGD |
| | Learning Rate | 0.0001; 0.001; 0.01 |
| Training | Epochs | 5; 10; 25; 50; 75; 100, 150 |
| | Batch Size | 16; 32; 64 |
| | Class Weights | None; Balanced |

Table 3 - Parameter Distribution for Randomized Search

Activation functions

Activation functions in neural networks essentially take a single value and perform a mathematical operation on it. When the function converges to a specific value, the neuron 'triggers' the next one,

hence the name activation function. This concept derives from neurons in the human brain and is also the reason for the framework's name: neural network.

**ReLu** is the most commonly used activation function, introduced by LeCun et al. (1998). Its purpose is to increase the non-linearity of the neural network. Despite being simple, ReLu is a non-linear function. Because there is no parameter inside ReLu (the formula can be seen in Table 4), it also does not require parameter-backpropagation. By setting all negative values to 0, a neuron only actives for images that actually possess the pattern (Wu 2017, 10).

As a result, this particular activation function is well suited for recognising objects and complex patterns. The introduction of ReLu in CNNs significantly reduced the difficulty of learning and improved the accuracy of the networks (Wu 2017, 9).

Before ReLu, **Sigmoid** was one of the most used non-linear transformations. Sigmoid transforms to values between 0 and 1 and is best suited for input data that itself is between 0 and 1 (Ittiyavirah 2013, 312). However in many cases, it performs poorer than ReLu (Wu 2017, 11). A commonly used activation function for the output layer is **Softmax**, which is a combination of many Sigmoid functions. Even in networks with ReLu in the inner layer, this is often the preferred output layer for probabilities or multi-class-classifications. In the latter, probability for each class will be the output (Ittiyavirah 2013, 314).

**Tanh** looks quite similar to sigmoid; however, it is centred around the origin of the coordinate system. That is why it can depict values between -1 and 1 instead of 0 and 1. Its gradient is also steeper in comparison since it has to reach twice as many y values for the same x value. Generally, Tanh is preferred to sigmoid because here, the gradient is not as restricted in one direction and also because it is origin-centred (Sharma 2020, 313). Even though ReLu is the standard in most CNNs nowadays, it can only outperform Tanh in deeper neural networks. That means when there are many layers, and problems such as the vanishing gradients occur (Godin 2018, 8).

| **Activation Function** | **Formula** |
|---|---|

| | |
|---|---|
| ReLu | $\mathrm{f(x)} = \max(0, x)$ |
| Sigmoid | $f(x) = \frac{1}{1 + \exp(-x)}$ |
| Tanh | $f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ |

Table 4 - Activation Functions and Formulas

Source: Sharma 2020, 313

As depicted in Figure 5, sigmoid and tanh both converge towards specific values, either -1, 0 or 1. This convergence leads to 'vanishing gradients' if the absolute values are too large. ReLu, on the other hand, erases all negative values and keeps the positive ones as they are, leading to 'exploding gradients' (Lee and Song 2019, 593).

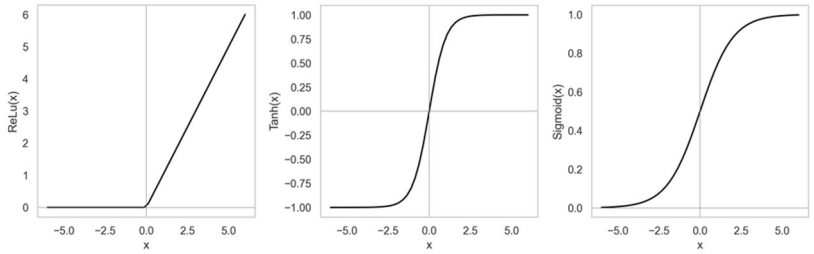


Figure 5 Activation Functions

Source: Own illustration based on Lee and Song 2019, 594

Padding

(Zero) padding allows to control the spatial size of the output of a CNN by adding an appropriate number of pixels (with zero values) to the outer edges of the input feature map before it is processed by the kernel (Chollet 2017, 126). Padding is used when it is desirable to obtain an output feature map with the same spatial dimensions as the input. Therefore, the padding parameter is set to **same** (Chollet 2017, 126; Lee and Song 2019, 608). Otherwise, **valid** means that no padding is performed and that the size of the feature maps gradually decreases along the convolutional layers (Lee and Song 2019, 599). In case the input feature map has a size of (n,n) and the filters have a size of (m,m), then a single output feature map is of size (n-m+1, n-m+1) (Lee and Song 2019, 599).

Pooling

Pooling layers are used to reduce model complexity, limit computation in the network and control issues of overfitting by reducing the spatial size of a feature map. The pooling layer partitions the input into a set of non-overlapping two-dimensional spaces. The pixel values of each subregion are then mapped according to the type of downsampling operator chosen: In **max pooling**, the values are summarized into one maximum value, whereas in case **of average pooling** the mean of the activations in the previous layer is computed for each subregion. (Lee and Song 2019, 598).

Batch Normalization

Normalization methods are used to increase the similarity of samples and hence, to improve generalization, i.e., the models' ability to perform well to unseen data. However, it is insufficient to normalize the data in the preprocessing stage, before feeding it into the model, only. Normalization is not guaranteed for each output after each transformation operated by the CNN since the mean and variance can change over time. (Chollet 2017, 260). The **batch normalization** layer, typically used after a convolutional layer (Chollet 2017, 261), ensures to continuously normalize the data during the training process by standardizing the values in each layer to mean 0 and variance 1 before the activation layers (Ioffe and Szegedy 2015). By making data standardization an integral part of the model architecture, faster and more stable training is possible, allowing the model to improve prediction accuracy (Lee and Song 2019, 609; Santurkar et al. 2018). Due to the implementation of batch normalization layers, higher learning rates can be used (Ioffe and Szegedy 2015; V. Thakkar, Tewary, and Chakraborty 2018, 2) and deeper networks can be built (Chollet 2017, 260).

Dropout

Regularisation is a method that is particularly relevant for preventing overfitting and improving generalization of deep learning models. Dropout is one of the most frequently applied regularisation techniques for CNNs (Srivastava et al. 2014). It randomly drops out input features during the training process, meaning it sets some of the weights connected to a given percentage of nodes in

a CNN to zero (Chollet 2017, 109; 216). The dropout rate refers to the fraction of features that are replaced with zero during training and lies usually between 0.2 and 0.5. For each update in each training epoch, the removed units are not included in the calculations of the current step (Krizhevsky, Sutskever, and Hinton 2017). Dropout is not applied to the test or validation set. In this case, the output of a layer is scaled down by a factor equal to the dropout rate to account for the fact that there are more units than during training. (Chollet 2017, 109).

Epochs

An epoch refers to the one-time training of the CNN with the entire dataset (Sharma 2017). However, since the size of an epoch is usually too large to be fed to the network in a single batch, it is divided into several smaller batches (Chollet 2017, 34). To improve the training process of the model, the number of epochs is increased, i.e., the data is passed to the same CNN multiple times (Sharma 2017). This way, the average loss on the training set is decreased until the optimal curve is met, more precisely, until the network begins to overfit the training data (Wu 2017, 7).

Optimisers

Optimisers are used to tweak the model's parameters during training. In Table 5, the used optimisers and their respective formulas can be inspected.

**Adam**, short for Adaptive Momentum Estimation, is one of the most widely used optimisation algorithms in CNNs. Adam is an iterative algorithm that adapts the model variables. Research has shown that Adam is effective for optimizing large groups of problems (Zhang and Gouza 2018, 1). However, for non-convex objective functions, it has shortcomings as Adam cannot promise to find a global optimum, as its iterative optimization might get stuck in a local optimum. Therefore it cannot be described as a particular robust optimizer for noisy data (Zhang and Gouza 2018, 2).

**Stochastic gradient descent (SGD)** is probably the most widely used optimizer for CNNs (Wu 2017, 7). Generally, it is a fast algorithm that only performs small computations at each descent. As many image recognition problems are based on noisy data, it is a fitting choice. Choosing the

correct learning rate offers a solution to the problem of getting stuck in local optima. When the dataset is heterogenous it can get unstable, and the loss decreases on average. SGD chooses samples at random throughout an epoch, so some samples might get chosen twice and some not at all (Lee and Song 2019, 597).

| **Optimiser** | **Formula** |
|---|---|
| RMSProp | $E(g^2) = \beta E(g^2)_{t-1} + (1-\beta)\left(\frac{\delta C}{\delta w}\right)^2$<br>$w_t = w_{t-1} - \frac{\eta}{\sqrt{E(g^2)}}\frac{\delta C}{\delta w}$<br>$where\ E(g^2) = Moving\ average\ of\ squared\ gradients$<br>$\frac{\delta C}{\delta w}$ =*gradient of cost function with respect to the weight*<br>$\eta = learning\ rate$<br>$\beta = moving\ average\ parameter$<br>$and\ \theta = cost\ function$ |
| Adam | $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$<br>$v_t = \beta_2 v_{t-1} + (1-\beta_2) g^2{}_t$<br>$M_t = \frac{m_t}{1-\beta_1^t}$<br>$V_t = \frac{v_t}{1-\beta_2^t}$<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{V_t}+\epsilon} M_t$<br>$With\ \eta = learning\ rate, m = past\ gradient$<br>$v = past\ square\ gradient, \beta = decay\ rate$<br>$\epsilon = smoothing\ term\ \ and\ \theta = cost\ function$ |
| Stochastic Gradient Descent (SGD) | $w = w - \eta \Delta Q(w)$<br>$Q(w) = \frac{1}{n}\sum_{i=1}^{n} \Delta Q_i(w)$<br>$where\ \eta = learning\ rate$<br>$Q(w) = loss\ function$<br>$w = parameter$ |

Table 5 - Optimisers and Formulas

Source: Zhang and Gouza 2018, 2; Kingma and Ba 2014, 2; Hinton, Srivastava, and Swersky 2012, 20

**RMSProp** or Root Mean Squared Propagation has become one of the more popular gradient algorithms beyond SGD. It has been used for very deep CNNs for computer vision and in some notable cases, outperformed SGD and Adam (Mukkamala and Hein 2017, 3). Even though it was designed for deep neural networks, it performs quite well with noisy data in deep learning and

hence for CNNs. It also offers opportunities like SGD to escape the local optima and contains the Adagrad optimiser when tuned with the correct parameters (Mukkamala and Hein 2017, 2).

Batch size

Batch size denotes the number of input samples in a single batch used for a training iteration (Lee and Song 2019, 595). The choice of batch size affects the batch normalization process as the technique depends on the number of samples in a batch. In general, smaller batch sizes have been found to provide a faster training process and a better generalization compared to larger batch sizes (Shen 2018).

Learning rate

The learning rate describes the extent to how much the model weights are changed during the training process (Brownlee 2019). It takes on a small positive value. The smaller the learning rate, the smaller the changes made at each iteration and thus the higher the number of training epochs necessary. Vice versa, a higher learning rate implies a more rapid adaptation and therefore requires less training epochs. Tuning this hyperparameter is essential as a too high learning rate can cause the model to converge quickly on a suboptimal solution, whereas a too low learning rate can cause the training process to become unstable and time-consuming (Brownlee 2019; Lee and Song 2019, 596).

Kernel size

The kernel_size is a key hyperparameter of the convolutional layer referring to the size of the kernel, a matrix moving over the input data, as explained in section 3.1.2. The input image is separated into sub-regions by the convolutional layer to have a fixed size set by the kernel size. The kernel size refers to the height x width of the filter mask. (Lee and Song 2019, 597 – 598).

Class weights

The weights are used for computation between layers and are updated repeatedly in a model by the algorithm. The aim is to find an optimal set of weights ensuring a minimum loss during the

network's learning. Class weights are commonly used for imbalanced datasets and can be set to 'balanced' to replicate the smaller classes to fit the number of samples in the bigger classes. (Lee and Song 2019, 593).

### 3.7 Performance Evaluation

To evaluate our model, computational and financial performance measures need to be distinguished.

#### 3.7.1 Computational Evaluation

As the stock price movement prediction represents a classification problem, evaluation for computational performance is feasible with the means of common evaluation metrics derived from the confusion matrix (Chen et al. 2021, 77). For assessing and comparing the computational performance of the constructed models, six performance metrics will be considered.

Accuracy

**Accuracy** as the first metric being used represents one of the simplest and most intuitive methods, showing how many classes have been predicted correctly.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + \ True\ Negatives + False\ Positives + False\ Negatives} \quad (10)$$

The accuracy metric can convey false impression of the performance of a model if classes are unbalanced. However, high accuracy is very important in the context of trading since every misclassification should be seen as a wrong trading decision and thus implying loss.

Precision

Precision is the second metric being used. Class-specific precision measures for each class separately the percentage of correct predictions, i.e. the percentage of instances predicted as the respective class that actually belong to the class. Precision values are bound between 0 and 1.

Moreover, the macro-averaged and weighted-averaged precision show the average model precision across all classes.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{11}$$

The type I error is penalized by the precision metric, resulting in lower values with a high type I error. Applied to trading, precision puts more emphasis on risk aversion, showing how many bad investment choices were impeded or how many trading decisions were predicted correctly. For buying transactions to prevent the trader to falsely buy although the asset might not further rise in value, resulting in a loss of value if the price goes down. Falsely predicting to sell will lead to missing out on possible returns if the asset is further rising in value.

Recall

Recall is a measure of how well the model identifies instances of a specific class in the data set.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{12}$$

A high recall means that the model is strong at identifying actual instances of its respective class, whereas a low recall means that the model is only able to identify a small percentage of instances of the class. Recall values are bound between 0 and 1. **Recall** is related to the presence of type II error (Peng et al. 2021, 23). In the context of trading, a higher recall implies not missing out on potentially profitable trading opportunities, indicating how many truly positive instances were marked as such and to decrease the number of false positives (Peng et al. 2021, 23). Related to a real-world trading scenario, a high recall leads to less falsely not-buying decisions although it would have been profitable. In terms of selling triggers, it denotes to not overlooking selling opportunities, preventing to hold the asset when the price will decrease.

F1-score

The F1-score balances precision and recall and provides a harmonic mid-point between recall and precision as it is granting a high value only if both values are performing well (Peng et al. 2021, 23–24).

$$F1 - score = 2\,x\,\frac{Precision\ *\ Recall}{Precision\ +\ Recall} \tag{13}$$

It harmonises indications on how precise the model is as a classifier, i.e. how many instances are correctly predicted, and how robust the model is, i.e. how good it is at identifying instances of the class.This metric can be very useful for strongly unbalanced predictions as the accuracy measure can indicate misleading results (Peng et al. 2021, 24). However, it is less intuitive as it is combining two metrics and is representing a poor resource allocation in this trading context. To gain detailed insights into the quality of the model, precision and recall should be checked separately and relative importance should be placed on recall and precision based on the specific underlying problem (Peng et al. 2021, 24).

Application of the performance measures

In the context of computational efficiency, the focus will lie on accuracy, since each prediction represents a trading decision that results in financial loss if misclassified. Since the datasets are unbalanced (Hold class is dominating each ETF) it is important to make sure that all classes will be predicted while minimizing the false positive rate. Therefore, the precision, recall and F1-score will help to get more insights into the models' prediction behaviour.

To ensure cross-industry comparability, a similar methodology including a similar labelling and model approach is used, except for the Oil and Gas sector. The acquired results will be compared and analysed based on the previously mentioned computational common performance measures, as well as on the basis of financial evaluation approaches which will be discussed in the next part.

### 3.7.2 Financial Evaluation

General approach

As the general approach to the financial evaluation of the model performance, a method suggested by Sezer, Ozbayoglu and Dogdu (2017) will be used. In this approach, the asset is bought, sold or held in accordance with its predicted label:

- If the prediction is Buy, the asset will be bought at current market price.
- If the prediction is Sell, the asset will be sold at current market price. Any existing long position will be closed, i.e. held shares sold, and a short position will be entered, i.e. shares will be short-sold.
- If the prediction is Hold, no operation is performed at that point in time.

Equal to Sezer, Ozbayoglu and Dogdu's approach, a starting capital of 10,000 USD will be used and each transaction (Buy and Sell) will be made using the full capital available at that moment. If the same label is repeated directly after one another in a sequence, only the first label will be considered as a trigger and the respective transaction executed. Repeat labels will be ignored until a new label comes up. At every executed transaction, trading fees will be considered to achieve a near-real scenario.

For the evaluation, the total return over the test period will be used. Given that each individual industry analysis will be applied to the same time period, and as such the test period will be equal, the comparability of industries with this metric is given.

Basic premises and assumptions

For the approach to be consistent, a number of clear assumptions need to be stated:

1. **Trading fees:** Trading fees stay constant during the whole test period.
2. **Execution price:** As the prediction will be based made on the data available at the end of day t for day t +1, the closing price of day t will be used as execution price.

3. **Fractional shares:** The approach assumes that fractional shares can be purchased. As such, the number of shares purchased or sold in transaction is equal to the total available capital divided by the execution price.
4. **Short-sell limit:** A short-sell limit of 20% of available capital is set, such that in a short-sell transaction, the short position cannot exceed 20% of the total capital available after closing the long position at the moment of a sell signal.

Benchmarks strategies

As benchmarks to compare the financial performance of the model to, the following strategies will be used:

1. **Simple, passive Buy & Hold strategy:** the asset is bought at the beginning of the test period and held until its end. The total return is determined by comparing the value of the investment at the end of the observation period to the start capital.
2. **Simple Moving Average Cross-over Strategy:** One shorter-term simple moving average and one longer-term simple moving average will be applied. In line with technical trading rules, it is considered a buy signal when the shorter-term moving average exceeds, i.e. crosses over, the longer-term moving average (Mitchell 2021). On the other hand, it is considered a sell signal when the shorter-term moving average crosses below the longer-term moving average (Mitchell 2021). For the application in this methodology, in case of a buy signal, the asset will be bought at market price. In case of a sell signal, any existing long position will be closed at market price and a short position in line with the short-selling limit will be entered.

   The best performing moving average combination will be found through a‘simplified randomised search based on the training data set.
3. **Mean-Reversion Strategy:** The mean-reversion trading strategy is built on the premise that prices eventually will revert back towards their mean (Chen 2021). Upper and lower

Bollinger bands are built around the asset price in a distance that is a function of the assets volatility measured as its standard deviation and a simple moving average is constructed (Chen 2021). On the one hand, if the asset price is below the Lower Bollinger Band, the asset is considered oversold and as such undervalued and expected to increase, reverting back towards its mean. This results in a buy signal, meaning that a long position should be built. On the other hand, the if the asset price is above the Upper Bollinger Band, the asset is considered overbought and overvalued and expected to decrease (Chen 2021). This results in a sell signal, meaning that any long position should be exited and a short position opened. In addition, for the strategy approach used in this paper moments where the price crosses the SMA are considered as unclear signals, signalling the investor to go neutral, i.e. to close any long or short position.

# 4 Industry Implementation

## 4.1 Introduction to the industry implementation

The following sub-paper focuses on the application of the previously established methodology to the U.S. financial services sector. For this purpose, a brief analysis of the used ETF will be provided and important market dynamics driving major events will be explored. These dynamics will be presented in a simplified way, by focusing on their key drivers, to avoid extending this analysis, reserving the focus of this sub-paper to the application of the developed CNN forecasting methodology and an analysis of the obtained results. As a next step, a brief description of the chosen features and the application of the established methodology to the chosen ETF for the U.S. financial services sector will be provided. After that, the results of the CNN application will be explored and discussed, comparing them to the results obtained from the other industry applications mentioned in the common part. The industry analysis closes with industry-specific conclusions about and an outlook on the application of CNNs to price movement forecasting in the financial services sector.

## 4.2 Industry Analysis

The following section 2 will provide an overview on the ETF that is used in the subsequent analysis and give brief insights into key market dynamics that drove major events during the ETF observation period.

### 4.2.1 ETF Description

The ETF to be analysed in this part of the paper as representative of its underlying sector is the iShares U.S: Financial Services ETF (IYG). The IYG's benchmark index, i.e. the index whose value the ETF is tracking, is the Dow Jones U.S. Financial Services Index ("Ishares U.S. Financial Services ETF | IYG" 2021). It invests in a market-cap-weighted subset of US stocks exclusively from the financial services industry. The IYG is a physically replicating ETF, meaning that it actually invests in the stocks of its underlying benchmark index instead of synthetically replicating

its returns (ETF.com 2021). Tracking the Dow Jones U.S. Financial Services Index, which is composed of publicly traded stocks of investment and commercial banks, consumer finance institutions, asset managers, credit card companies and securities exchanges, the ETF offers investors pure exposure to the American financial services sector (ETF.com 2021). The Dow Jones U.S. Financial Services Index is reconstituted yearly, meaning that companies are added and removed as needed to ensure that the index reflects the up-to-date status of the sector it is supposed to represent ("Ishares U.S. Financial Services ETF | IYG" 2021). The IYG is composed entirely of US equities ("Ishares U.S. Financial Services ETF | IYG" 2021) and invests across the market-cap spectrum, i.e. is not focused on one market-cap universe only (ETF.com 2021). As of October 31st, 2021, the IYG has a 3-year Equity Beta of 1.31, meaning that on average the index shows a higher volatility than the market.

The IYG is composed of holdings in 108 companies from the US financial services sector, while the 10 largest holdings of it account for more than 54% of the total weight of the index ("Ishares U.S. Financial Services ETF | IYG" 2021).

| Rank | Company | Weight |
|---|---|---|
| 1 | JPMorgan Chase & CO | 11.45% |
| 2 | Visa Inc. (Class A shares) | 8.02% |
| 3 | Bank of America Corp. | 7.86% |
| 4 | Mastercard Inc. (Class A shares) | 6.79% |
| 5 | Wells Fargo | 4.74% |
| 6 | Morgan Stanley | 3.47% |
| 7 | Goldman Sachs Group Inc. | 3.12% |
| 8 | BlackRock Inc. | 3.11% |
| 9 | CitiGroup Inc. | 3.08% |
| 10 | Charles Schwab Corp. | 2.85% |
| | **Total weight top 10 companies** | **54,49%** |

Table 6 - Top ten holdings of the IYG ETF with their respective relative weight

Source: Own illustration

Given its composition and exposure to U.S. financial services, the IYG is an ideal ETF to assess the performance of CNNs in forecasting price movements for the U.S. financial services sector.

### 4.1.2 Exploration of the IYG price data set

In this paper, price data of the IYG in the period from 01.01.2010 to 31.12.2019 will be analysed. This period is chosen to exclude most effects of both the financial crisis in 2007/2008 and the Covid-19 pandemic starting in spring 2020, which both dramatically altered the financial market conditions and dynamics (Grammatikos and Vermeulen 2014; Zhang, Hu and Ji 2020). The objective of this paper is to analyse the general applicability of CNN to forecasting price movement and as such, extremely atypical, long-lasting macro-economic events that significantly alter market structures, i.e. the financial crisis and the Covid-19 pandemic, will be excluded to not falsify the obtained results.

On 01.01.2010, the first day of the analysis period, the price of IYG stands at 54.87 USD. On 31.12.2019, the last day of the analysis period, the price per unit stands at 151.82 USD, showing a price increase of 96.95 USD, 176.7% in relative terms, over the 10-year period.

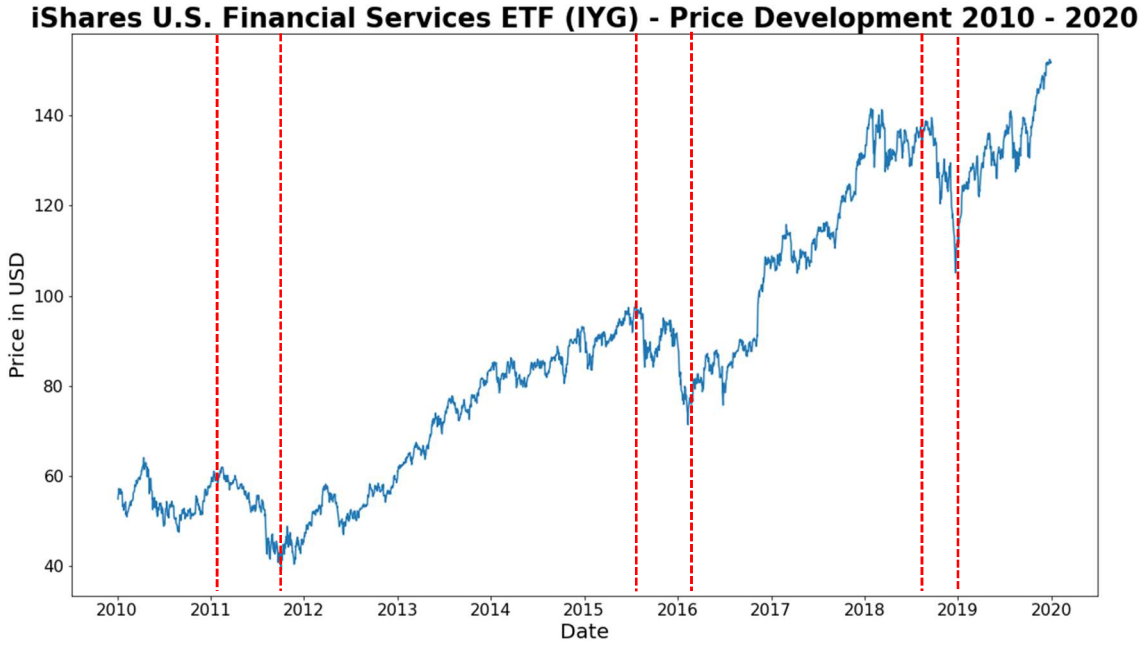


Figure 6 iShares U.S. Financial Services ETF (IYG) – Price development from beginning of 2010 until end of 2019

Source: Own illustration

Although the overall price trend is positive, three significant downturns can be observed at distinct points in time:

1. during 2011,
2. between mid of 2015 until the beginning of 2016
3. starting at the end of 2018 until the beginning of 2019

All three downturns are marked by the red dotted lines in figure 1. The reasons for these downturns will be explored in the section below.

### 4.1.3 Dynamics in the Financial Services sector during the analysis period

**Early 2010s:**

During the financial crisis, equity markets in general and especially financial services stocks had plummeted dramatically (Wehinger 2012). In the early 2010s, supported by large government stimulus packages and dedicated policies, financial markets were still recovering from the aftershock of financial crisis. However, this recovery was mostly policy-driven and there was large uncertainty about future policy developments, inhibiting the recovery (Wehinger 2012). A combination of factors like the Eurozone crisis, the Japanese tsunami and the subsequent meltdown of the Fukushima nuclear plant, the surge in commodity prices, persisting turmoil in the middle east and the U.S. sovereign rating downgrade caused a sharp downturn in the stock markets in 2011, explaining the first downturn that can be observed in figure 1 (Wehinger 2012).

**China stock market crisis 2015:**

The second observed downturn, in 2015, can be attributed to the Chinese stock market turbulence that dramatically affected stock markets in the U.S. A variety of factors, such as falling borrowing costs due to a loosened monetary policy, a legal liberalisation of stock markets and new government regulations on real estate financing, had led to a dramatic increase in investment activities, led especially by Chinese retail investors, which accounted for more than 80% of trading activity

in China (Allen 2015). This sharp increase in stock market investments inflated stock prices and caused a stock market bubble. However, at some point, banks and brokers started to issue margin calls and request paybacks on loans, and a growing number of analysts started warning of a significant overvaluation of stocks. This caused widespread selling of stocks and other assets, leading to a sharp downturn in the Chinese financial markets (Allen 2015). Given China's complex ties with many other countries as well as a variety of dynamics in the financial markets, the Chinese stock market crisis started spilling over to other countries and consequently also affected U.S. financial markets (Allen 2015).

**2018/2019: US-China trade war**

Lastly, the third significant downturn that can be observed in the price data set in 2018/2019 can be attributed to the U.S.-China trade war that had started in 2017. Three main concerns led the U.S. government under Donald J. Trump to initiate this trade war: (1) concerns that job creation in the US was harmed by China's constant trade surplus, (2) suspicions that China applied illegal and unfair strategies to acquire U.S. technology and knowledge and was trying to weaken U.S. national security and (3) fears that China was seeking to replace the U.S. in their international position (Liu and Wing 2019). Although underlying dynamics had existed before, the situation escalated after the Trump office, which had already taken a strong anti-China position during the elections, took office (Ferguson and Xu 2018). The trade war between the two countries expressed itself in a sequence of tariffs and other trade restrictions, which led to disruption in value chains and caused significant uncertainty in both industrial and financial markets.

Given that 20% of the total data set, i.e. roughly two years of price data, will be held back as a test data set, the downturn in 2018/2019 will be part of this test data. It might be argued that this could negatively affect the model performance as it represents an atypical development in the markets that does not reflect general market conditions. However, given that the model will be

trained on data that contains the other two, similar downturns, it is expected that the model can learn to recognise the respective patterns.

### 4.3 Data pre-processing, feature engineering and image encoding

Section 3 will describe the data set used as model input as well as the pre-processing of the data and image encoding in preparation for the model training.

#### 4.3.1 Data and technical indicators

As input for the raw data set, the following variables are included as features:

| | |
|---|---|
| Price and volume data | • Open price<br>• Closing price<br>• High price<br>• Low price<br>• Daily trading volumes |
| Simple Moving Average | • 5-day SMA<br>• 10-day SMA<br>• 20-day SMA |
| Exponential Moving Average | • 5-day EMA<br>• 10-day EMA<br>• 20-day SMA |
| Rate of Change (RIC) | • 12-day rate of change |
| Percentage Price Oscillator | • Percentage Price Oscillator based on closing price<br>• Percentage Price Oscillator based on opening price |
| Relative Strength Index | • 14-day Relative Strength Index |
| Williams % Range | • 14-day Williams % Range |
| Commodity Channel Index (CCI) | • 20-day Commodity Channel Index |
| Foreign Exchange Rate | • USD/EUR exchange rate<br>• USD/GBP exchange rate<br>• USD/JPY exchange rate |
| Know Sure Thing Oscillator | |
| Moving Average Convergence Divergence | |

Table 7 - Variables included in the initial data set

Source: Own illustration

Given that foreign exchange rates have a significant impact on financial institutions in general and banks specifically (Federal Reserve Bank of San Francisco 1996), exchange rates between of the U.S. dollar (USD) and three major currencies are included as explanatory variables, specifically, the Euro (EUR), the Japanese Yen (JPY) and the British pound sterling (GBP). Despite its impact on financial services institutions, an inclusion of the Fed base interest rate was eventually disregarded since those rates are only adjusted infrequently. This is why an inclusion of the base interest rate is not expected to add value to forecasting daily prices.

This selection of price and volume data, technical indicators and exchange rates results in an initial data set with 22 different features, which are used for the further analysis sequence.

### 4.3.2 Stationarity test and differential calculus

As established in section 3.3.2 of the methodology part, the price data as time-series data needs to be tested for its stationarity and, in case of stationarity, be adjusted using fractional differentiation to ensure that it does not hamper the modelling process (Hyndman und Athanasopoulos 2018). To avoid data leakage from potentially transforming the entire data set, i.e. train data set and test data set together, the data is split before the stationarity test and the potential data transformation.

Applying the Augmented Dickey-Fuller test to the train data set, the obtained p-value of 0.9734 indicates non-stationarity of the data, i.e., a trend in the data, which would likely negatively affect the model performance. Therefore, as explained, fractional differentiation is applied to adjust the data. For this purpose, the fractional differentiation is fitted to the training set and then applied separately to the train set and test set. Repeating the stationarity test, a p-value of 0.0000 is obtained, indicating that the data has been successfully adjusted to be stationary.

### 4.3.3 PCA results

After adjusting the data for stationarity, a Principal Component Analysis (PCA) is applied with a threshold of 0.95. In the typical procedure of a PCA, the analysis is fitted to the train data set and

then applied separately to the train and test data set to avoid data leakage. While the initial raw data set contained 22 features, the PCA reduces the number of features to 11.

### 4.3.4 Image encoding and model training

As a next step, the data set obtained from the PCA with 11 features is encoded into three types of images, i.e. Gramian Angular Differentiation Field, Gramian Angular Summation Fields and Markov Transition Fields, with a window of 10 days, meaning that each image contains data from 10 subsequent days. Given the 11 features and a window of 10 days, each image has a 10*10*11 shape.

These images are used to train three different CNNs, each based on one of the image types. The best performing parameters will be found using a randomised search, choosing the model with the highest F1-score and accuracy and making sure that each model predicts all three classes. The results of this model training in terms of performance metrics will be presented and analysed in the following section.

## 4.4 Analysis and interpretation of the model results

The following section 4 focuses on the analysis and interpretation of the computational and financial performance of the best performing model for each of the three image types and provides a comparison between the performances of the three models.

### 4.4.1 Comparison of overall model metrics and benchmark to purely random model

The following section will provide a comparison of the overall model metrics across the three constructed model (GADF, GASF and MTF) to draw first conclusions on which model achieves the highest computational performance.

| | GADF | GASF | MTF | Random model |
|---|---|---|---|---|
| **Accuracy** | 0.49 | 0.48 | 0.47 | 0.373 |
| **Macro-averaged F1-score** | 0.36 | 0.26 | 0.31 | 0.370 |
| **Weight-averaged F1-score** | 0.43 | 0.35 | 0.39 | 0.333 |

Table 8 - Comparison of overall model metrics across the three model types and a purely random model
Source: Own illustration

Based on these overall model performance metrics, the GADF model achieves the highest computational performance, outperforming the other two models in all three performance metrics, however, by relatively small margins, especially regarding the key metric accuracy. Regarding the comparison of the GASF model to the MTF model, none of the two is clearly outperforming the other based on their overall model performance metrics, given that they exceed each other in different performance metrics.

Furthermore, to compare the models to a purely random one, in 10,000 iterations, random predictions were generated, using the length of the test data set and the class probabilities of the train data set, and the accuracy, macro-averaged F1-score, and weight-averaged F1-score were calculated for each iteration and averaged. All three models achieve a higher accuracy and weight-averaged F1-score than the purely random model, indicating a better performance in this key metrics. However, all three models' macro-averaged F1-scores is lower than their weight-averaged one and lower than the random model's macro-averaged F1-score. Given their calculation, a macro-averaged F1-score that is lower than the weight-averaged F1-score indicates lower individual F1-scores for at least one of the minority classes and thus a lower precision and/ or recall for this class. Thus, all three models seem to have difficulties with Sell and/ or Buy class predictions, which each make up approximately 25% of the data set.

Thus, in order to develop a better understanding of the performance of these models with respect to each individual class, as a next step, the class-specific metrics of the three models will be analysed.

### 4.4.2 Comparison of class-specific metrics

Given that the overall model performance metrics mentioned in the section above are an aggregation of different class-specific metrics, they do not provide a detailed impression on strengths and weaknesses of the models regarding their predictive power for the different classes. Table 5 contains the class-specific metrics for all three models, providing more details on the model performance for the Hold, Buy and Sell class.

| | **GADF** | **GASF** | **MTF** |
|---|---|---|---|
| **Precision** | | | |
| **Hold class precision** | 0.54 | 0.50 | 0.50 |
| **Buy class precision** | 0.34 | 0.27 | 0.27 |
| **Sell class precision** | 0.37 | 0.26 | 0.37 |
| **Recall** | | | |
| **Recall: Hold** | 0.82 | 0.94 | 0.85 |
| **Recall: Buy** | 0.23 | 0.02 | 0.08 |
| **Recall: Sell** | 0.11 | 0.05 | 0.12 |
| **F1-Score** | | | |
| **F1-score: Hold class** | 0.65 | 0.65 | 0.63 |
| **F1-score: Buy class** | 0.27 | 0.04 | 0.13 |
| **F1-score: Sell class** | 0.17 | 0.08 | 0.19 |

Table 9 - Comparison of class-specific metrics across the three model types

Source: Own illustration

The GADF-based model outperforms both the GASF- and MTF-based in terms of class-specific performance metrics, i.e. class precision, class recall and class F1-score. While in terms of class precisions the three models show similar performances with only small differences, significant differences can be observed for both recall and F1-score values.

While the GADF model has a lower Hold class recall compared to the GASF and MTF model (0.82 v 0.94 v 0.85), it achieves a much better recall for the Buy class, indicating that it is better at identifying Buy class instances than the other two models. The GADF also outperforms the GASF in the Sell class recall, achieving a recall that is more than 2x as high (0.11 v 0.05). However, the MTF model achieves a slightly higher Sell class recall than the GADF model (0.12 vs 0.11), with GASF having the highest recall for the Hold class, but the lowest value for both the Buy and Sell call recall.

In the context of trading, the following conclusions can be drawn on the model performances: Firstly, the GASF model's low recall for Buy and Sell means that is weak at identifying Buy and Sell instances and thus will likely miss a large part of return-generating transactions. Secondly, the GADF's comparatively high recall for the Buy and Sell class show that is better at identifying these classes and indicate that it will thus trigger more Buy and Sell transactions than the other two models and potentially profit from more return-generating opportunities.

### 4.4.3 Financial performance evaluation

Given that, in practice, the main objective of a trading strategy is to generate returns and ideally outperform the market the financial performances need to be assessed.

| | CNN Model | Buy & Hold | SMA | MR |
|---|---|---|---|---|
| **GADF** | 18.78% | | | |
| **GASF** | -12.75% | 16.00% | 16.88% | 21.76% |
| **MTF** | -6.25% | | | |

Table 10 - Comparison of financial performances across the three model types

Source: Own illustration

Only the GADF model is able to generate positive financial returns and to outperform the easiest strategy, i.e. the Buy & Hold strategy. Both the GASF and MTF model generate losses, with the GASF generating the greatest loss of 12.75% over the test period. During the test period, the price of the IYG ETF increases by 16.00%, while the best model achieves a performance of 18.78%, 2.78 percentage points above the Buy & Hold return in absolute terms, 17.38% in relative terms. Based on this finding, a higher recall for the Buy and Sell classes appears to be related to a better financial performance. With the return of 18.78%, the GADF model outperforms the simple moving average cross-over strategy (16.88%) but lies below the return generated by the mean-reversion strategy (21.76%). Based on these findings, it seems that the CNN approach as applied in this paper is unable to disprove the Efficient Market Theory (Fama 1970) and does not offer significant advantages for the financial services sector compared to naïve technical strategies.

### 4.5 Comparison to results from other sectors

As a last step to the result and performance assessment, the model performances obtained for the financial services sector will be compared to the results obtained for the other sectors using the same approach. The purpose of this comparison is to assess if the apparently low performance of the models is specific to the financial services sector or if similar issues can also be identified in other sectors.

### 4.5.1 Computational performance comparison across industries

As a first step in the comparison, the computational performance of the models applied to the financial services sector will be compared to the average computational performance metrics across all industries.

| | **GADF average** | ***GADF FS*** | **GASF average** | ***GASF FS*** | **MTF average** | ***MTF FS*** |
|---|---|---|---|---|---|---|
| **Accuracy** | 0.49 | *0.49* | 0.45 | *0.48* | 0.44 | *0.47* |
| **Macro-averaged F1-score** | 0.35 | *0.36* | 0.31 | *0.26* | 0.31 | *0.31* |
| **Weight-averaged F1-score** | 0.47 | *0.43* | 0.37 | *0.35* | 0.37 | *0.39* |

Table 11 - Average performance metrics across five industries (IT, Healthcare, Energy, Financial Services and Industrials)

Source: Own illustration

The key insight of Table 7 is that there are no major differences in the model performance applied to the financial services sector compared to the average performance metrics across all industries. The average F1-scores show the same pattern as in the case of the financial services sector model: the weight-average F1-scores are higher than the macro-averaged F1-scores, indicating a poorer performance for at least one of the minority classes. Given that for all sectors, Buy and Sell are minority classes compared to the Hold class, this indicates similar weaknesses regarding the prediction of these classes. Overall, the performance obtained for the financial services sector are in line with the average results across all sectors using the same approach.

### 4.5.2 Financial performance comparison across industries

As a second and final step in the industry comparison, the financial performance needs to be evaluated. Since the different industries show significant differences in their price developments over the test data period, ranging from -8.0% for the Energy sector to 51.0% in the Information Technology sector, excess returns calculated as the absolute difference between the model return

and the benchmark strategy are being used to ensure comparability of the obtained results. These excess returns can be seen in Table 8 below.

| | Average excess return compared to Buy & Hold | Financial services excess return compared to Buy & Hold |
|---|---|---|
| **GADF** | 1.30% | 2.78% |
| **GASF** | -6.90% | -28.75% |
| **MTF** | -12.98% | -22.25% |

Table 12 - Financial performance comparison

Source: Own illustration

Considering the averages across all five industries using the same model approach, one can see that the approach performs similarly poorly for those other industries as it does for the financial services sector. On average, only the GADF models outperform the Buy & Hold strategy and thus the price development of the respective ETFs per se, however, only with a small margin of 1.28 percentage points. On average, both the GASF and MTF models perform worse than the Buy & Hold strategy, meaning that an investor would achieve better results by just buying and holding the asset instead of using these two models. As such, the average results across all 5 industries go in line with the results obtained for the financial services sector. The CNN approach taken in this paper appears unable to beat the market performance.

### 4.6 Conclusion and outlook

#### 4.6.1 Limitations

Although all three models achieve a better accuracy than the purely random model, they barely exceed the random model with respect to their F1-scores, showing weaknesses in recall and/ or precision. This weakness becomes more obvious when assessing the class-specific performance metrics. All three models have low recall values for the minority classes Buy and Sell, meaning that they are relatively weak at identifying those classes, implying that the models are rather badly suited to identify these kinds of return-generating opportunities. This impression is confirmed by

assessing the financial performance, which shows that only the GADF model generates positive returns, barely exceeding the Buy & Hold return.

One source of this problem is likely the imbalanced data set: The Hold class dominates the data set, making up roughly 50% of both data sets, which negatively affects model performance. Moreover, considering the complex features, the relatively small train data set of roughly 2000 pictures might be insufficient for a proper model training, causing an issue of overfitting.

However, the relatively poor performance does not seem to be specific to the financial services sector, but an overall problem of the approach, as other industries on average show similar performances. This indicates similar problems for the other industries.

### 4.6.2 Outlook

Further research appears necessary to improve the model performance. As for all industries, the GADF approach achieves the best performance, further research should be built upon this method. To address the issue of the small train data set, shorter time windows for the images and higher-frequency data, e.g. hourly instead of daily data, could be used to increase the number of images available for model training. Furthermore, seeking to solve the problem of imbalanced data, suitable data augmentation techniques and adjusted model training approaches, e.g. the so-called two-phase learning, presented by Wahab, Khan and Lee (2017), should be explored.

Other approach alterations that should be explored in further research include:

- Experimentation with different model architectures that might be better at extracting the necessary patterns
- Experimentation with different initial variables that might have more predictive power
- Adjustments in the labelling, exploring approaches based on characteristics that are more distinguishable in the images

Overall, the CNN approach should not be discarded yet, given that it is a relatively young field and further techniques are expected to develop.

## 5 Performance Comparison and Discussion

In the following section, key findings from the individual analyses conducted in chapter 4 will be summarised, focusing on common findings regarding the model hyperparameters, as well as the computational and the financial performance of the models.

Common findings hyperparameters

Comparing the best-performing model parameters across the three model types (GADF, GASF and MTF) and across the six analysed industries, several findings can be made.

Firstly, for the MTF-based models, a 5*5 kernel achieves the best performance across all industries. For the majority of GAF-based models, i.e. GADF and GASF, a 3*3 kernel leads to the best performance, with the exception of the Energy sector, for which a 5*5 leads to the best performance for all three models. This tendency can be supported by the PXL-based model, which also uses a 3*3 kernel.

Secondly, in the majority of models (17 out of 19), the Softmax and Sigmoid activation function achieve the best performance. The ReLu activation function only leads to the best performance for 2 of the 19 models.

Thirdly, for 5 out of the 6 ETFs applying the proposed image encoding types, average pooling achieves the best performance for the GADF model.

Fourthly, for the majority of analysed ETFs (5 out of 6), including the class weights does not have a positive impact on the model accuracy, i.e. models without class weights achieve a better accuracy for these ETFs. However, this tendency is not supported by the PXL-based model.

Common findings computational performance

For the majority of industries, i.e. Information Technology, Healthcare, Energy and Financial Services, the GADF-based model achieves a better accuracy compared to the GASF- and MTF-based models. Moreover, for 5 out of 7 analysed ETFs, GADF achieves better weight-averaged and macro-averaged F1-scores than both GASF and MTF.

For the Energy industry, it can be noted that GADF performs above the average of the other industries, whereas the GASF and MTF perform poorly compared to the other ETF's in terms of computational performance. The worst model across all industries can be found within the Healthcare models, where the MTF showed the poorest performance from a computational perspective with a weighted average F1-score of 0.34 and an accuracy of 0.3706. Among all models and industries, predictions of the Hold class showed the most promising results, with the only outlier found for the GASF model of the energy sector. It is also worth mentioning that within all industries and ETF's, with the VGT (IT sector) as an exception, class predictions show huge discrepancies in predicting the correct class. Hence it is not possible to conclude that a certain image encoding technique works better to predict a specific signal.

The performance evaluation of the random choice models didn't produce any important insights. For all industries, similar scores can be observed. Moreover, they are less performant than all other models when comparing weighted averages with each other.

Common findings financial performance

For comparing and assessing the financial performances of the models across industries, excess returns calculated as the absolute difference between the model return and the benchmark strategy are being used to ensure comparability of the obtained results. Considering the average of these excess returns, only the GADF models are able to achieve returns that exceed the Buy & Hold strategy, i.e. to beat the return generated by the general price development of the considered ETF. Both the GASF and MTF models have negative excess returns compared to Buy & Hold, leaving the investor with better returns by just buying and holding the asset compared to using a trading strategy based on the models' predictions.

For 4 out of the 6 ETFs to which the common methodology was applied, the GADF models outperform the Buy & Hold return, with the exception of Healthcare and Industrials. The GASF models only outperforms the Buy & Hold return for 2 out the 6 ETFs, i.e. Healthcare and Energy.

Only for the Energy sector, the MTF model outperforms the Buy & Hold return. Despite being a subset of the energy industry, the model used on the Oil & Gas sector cannot outperform the Buy & Hold return. It is also the Energy sector where the model generates the most impact; despite the negative price development of -8% over the test data period, all three models are able to generate positive returns between 3% and 10%. Lastly, the CNN approach shows the poorest performance in the Industrials sector where all three models underperform compared to the Buy & Hold strategy.

# 6. Limitations and Outlook

## 6.1 Limitations

Predictions for the stock market are challenging, as the stock market represents a dynamic, volatile and very complex market based on historical data and influenced by unpredictable events. In this research we face the problem of imbalanced classes, where the largest class is Hold across all sectors. As a result, the predictions are dominated by the largest class - predictions of the minor classes turn out worse, which negatively affects the overall model performance. In addition, a comparatively small train set in combination with complex features further complicates model development. This makes the models prone to overfitting - whereas the inclusion of multiple train data would be advantageous. In the present approach of this research accuracy was chosen as the most important performance measure and model selection criterion. However, there are other evaluation methods that could be considered as primarily evaluation metric, e.g. financial performance, precision or F1-scores. Especially with respect to the financial performance it is important to mention that only the decisions of the next day are considered. Hence, the prediction is related to a very short future period and makes no specific statements about longer term behavior. A further limitation lies in the assessment of the severity in the case of mislabelling. A wrong Buy/Sell decision has more serious negative effects than a wrong Buy/Hold or Sell/Hold decision. In the present research a suitable performance measure is missing - here a suitable loss function would be necessary. A further remark is to be mentioned in the simplification of the labelling approach. If the upper and lower limits are exceeded on the same day, the first labelling trigger decides on the label allocated to the trading day. Another limitation can be found in the Efficient Market Theory (Fama 1970, 383). As mentioned in section 2.1, the theory states that stock prices already reflect and have priced in all relevant information. This would make a deeper analysis with additional features, like technical indicators, redundant, as no investment analysis technique allows investors to generate significant excess returns above the market. However, this is refuted

by the thesis that financial markets in many cases do not react immediately to new information (Cervelló-Royo and Guijarro 2020, 41), which would make returns above the market average still possible through sufficient analysis and the right timing. This would imply that a better performing model could potentially outperform market returns.

### 6.2 Outlook

Forecasting Financial Time Series Movements using CNNs is a recent research field. For this reason many different topics can be addressed in future research.

Firstly, it would be interesting to test if the proposed methodology can achieve better results with regard to different prediction horizons. These could include the prediction of price movements within the next week or month, alternatively intraday data can be used for short-term forecasting. This work focuses on using technical indicators along with foreign exchange, commodity and indices as features to feed into the CNN. However, future work could incorporate other types of features. These could, among others, include data from the news, social media and market segments. Moreover, machine-learning-based fundamental analysis approaches as suggested by Cao and You (2020), e.g. for forecasting company earnings, could be included to provide a more holistic impression on the underlying companies' situation.

Furthermore, within the current research not all papers propose transforming the data into stationary time series. Therefore, research regarding the necessity of stationary time series in the context of forecasting financial time series with CNNs can be conducted. This is particularly interesting as methods to transform non-stationary data imply information loss within the used variables.

## References


Abad, Cristina, Sten A. Thore, and Joaquina Laffarga. 2004. 'Fundamental Analysis Of Stocks By Two-Stage DEA'. Managerial And Decision Economics 25 (5): 231-241. doi:10.1002/mde.1145.

Abdi, Hervé, and Lynne J. Williams. 2010. 'Principal Component Analysis'. Wiley interdisciplinary reviews: computational statistics 2(4): 433-459.

Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. 'Understanding of a Convolutional Neural Network'. In 2017 International Conference on Engineering and Technology (ICET), 1–6.

Allen, Katie. 2015. "Why Is China's Stock Market In Crisis?". *The Guardian*, 2015.https://www.theguardian.com/business/2015/jul/08/china-stock-market-crisis-explained.

Arratia, Argimiro, and Eduardo Sepúlveda. 2020. 'Convolutional Neural Networks, Image Recognition and Financial Time Series Forecasting'. In Mining Data for Financial Applications, 60–69. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-37720-5_5.

Banton, Caroline. 2021. 'An Introduction To Trading Types: Fundamental Traders'. Investopedia. Accessed December 10, 2021. https://www.investopedia.com/articles/trading/02/100102.asp.

Barra, Silvio, Salvatore Mario Carta, Andrea Corriga, Alessandro Sebastian Podda, and Diego Reforgiato Recupero. 2020. 'Deep learning and time series-to-image encoding for financial forecasting'. IEEE/CAA Journal of Automatica Sinica 7 (3): 683–692. https://doi.org/10.1109/JAS.2020.1003132.

Bergmeir, Christoph, and José M. Benítez. 2012. 'On The Use Of Cross-Validation For Time Series Predictor Evaluation'. Information Sciences 191: 192-213. doi:10.1016/j.ins.2011.12.028.

Bogullu, Vamsi Krishna, Cihan H. Dagli, and David Lee Enke. 2002. 'Using Neural Networks and Technical Indicators for Generating Stock Trading Signals'. Intelligent Engineering Systems Through Artificial Neural Networks 12: 721–726.

Brownlee, Jason. 2018. ‘When to Use MLP, CNN, and RNN Neural Networks’. Machine Learning Mastery. Accessed December 10, 2021. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/.

Brownlee, Jason. 2019. ‘Understand the Impact of Learning Rate on Neural Network Performance’. Machine Learning Mastery. Accessed December 10, 2021. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

Cao, Kai, and Haifeng You. 2020. ‘Fundamental Analysis Via Machine Learning’. SSRN Electronic Journal 2020 (009). doi:10.2139/ssrn.3706532.

Cervelló-Royo, R., and F. Guijarro. 2020. "Forecasting Stock Market Trend: A Comparison Of Machine Learning Algorithms". Finance, Markets And Valuation 6 (1): 37-49.

Chen, Sheng, and Hongxiang He. 2018. ‘Stock Prediction Using Convolutional Neural Network’. IOP Conference Series: Materials Science and Engineering 435 (1). https://doi.org/10.1088/1757-899X/435/1/012026.

Chen, Wei, Manrui Jiang, Wei-Guo Zhang, und Zhensong Chen. 2021. ‘A Novel Graph Convolutional Feature Based Convolutional Neural Network for Stock Trend Prediction’. Information Sciences 556 (May): 67–94. https://doi.org/10.1016/j.ins.2020.12.068.

Cheung, Yin-Wong, and Kon S. Lai. 1995. ‘Lag Order and Critical Values of the Augmented Dickey–Fuller Test’. Journal of Business & Economic Statistics 13 (3): 277–280. https://doi.org/10.1080/07350015.1995.10524601.

Chollet, Francois. 2017. Deep Learning with Python. New York, NY: Manning Publications.

Chollet, François. 2018. Deep Learning with Python. Shelter Island, New York: Manning Publications Co.

Cohen, Naftali, Tucker Balch, and Manuela Veloso. 2020. ‘Trading via Image Classification’. In Proceedings of the First ACM International Conference on AI in Finance, 1–6. https://doi.org/10.1145/3383455.3422544.

Drakopoulou, Veliota. 2016. ‘A Review Of Fundamental And Technical Stock Analysis Techniques’. Journal Of Stock & Forex Trading 05 (01): 1-8.

Dertat, Arden. 2017. ‘Applied Deep Learning - Part 4: Convolutional Neural Networks’. Towards Data Science. Accessed November 8, 2021. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

Desconfio, Josh. 2018. ‘A Beginner's Guide to Technical Indicators’. Scanz.com. Accessed December 3, 2021. https://scanz.com/technical-indicators-guide/.

Fama, Eugene F. 1970. ‘Efficient Capital Markets: A Review of Theory and Empirical Work’. The Journal of Finance, 25(2), 383–417. https://doi.org/10.2307/2325486

Federal Reserve Bank of San Francisco. 1996. Banks and Foreign Exchange Exposure. Eco nomic Letter, San Francisco: FRBSF.

Ferguson, Niall, and Xiang Xu. 2018. "Trump And The 'Chimerica' Crisis". *Wall Street Jour nal*, 2018. https://www.wsj.com/articles/trump-and-the-chimerica-crisis-1525635323.

Fernández-Blanco, Pablo, Diego J. Bodas-Sagi, Francisco J. Soltero, and J. Ignacio Hidalgo. 2008. ‘Technical Market Indicators Optimization Using Evolutionary Algorithms’. In Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation.

Ghosh, Anirudha, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. 2020. ‘Fundamental Concepts of Convolutional Neural Network’. In Recent Trends and Advances in Artificial Intelligence and Internet of Things, 172:519–67. https://doi.org/10.1007/978-3-030-32644-9_36.

Godin, Fréderic, Jonas Degrave, Joni Dambre, and Wesley De Neve. 2018. ‘Dual Rectified Linear Units (DReLUs): A Replacement for Tanh Activation Functions in Quasi-Recurrent Neural Networks’. Pattern Recognition Letters. 10.1016/j.patrec.2018.09.006

Grammatikos, Theoharry, and Robert Vermeulen. 2014. "The 2007-2009 Financial Crisis: Changing Market Dynamics And The Impact Of Credit Supply And Aggregate Demand Sensitivity". Applied Economics 46 (8): 895-911.

Haq, Anwar Ul, Adnan Zeb, Zhenfeng Lei, and Defu Zhang. 2021. ‘Forecasting Daily Stock Trend Using Multi-Filter Feature Selection and Deep Learning‘. Expert Systems with Applications 168 (April): 114444. https://doi.org/10.1016/j.eswa.2020.114444.

Hayes, Adam. 2021. ‘Know Sure Thing (KST)’. StockCharts. Accessed December 10, 2021. https://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:know_sure_thing_kst.

Henrique, Bruno Miranda, Vinicius Amorim Sobreiro, and Herbert Kimura. 2018. ‘Stock Price Prediction Using Support Vector Regression on Daily and up to the Minute Prices’. Journal of Finance and Data Science 4 (3): 183–201. https://doi.org/10.1016/j.jfds.2018.04.003.

Herman-Safar, Or. 2021. ‘Time Based Cross Validation’. Blog. Towards Data Science. https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8.

Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky. 2012. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." Neural Networks for Machine Learning 14.

Huang, Boming, Yuxiang Huan, Li Da Xu, Lirong Zheng, and Zhuo Zou. 2019. ‘Automated trading systems statistical and machine learning methods and hardware implementation: a survey’. Enterprise Information Systems 13 (1): 132–144. https://doi.org/10.1080/17517575.2018.1493145.

Hyndman, Rob J., and George Athanasopoulos. 2018. Forecasting: Principles and Practice. OTexts.

Ioffe, Sergey, and Christian Szegedy. 2015. ‘Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift’. In: International conference on machine learning. PMLR, 2015. S. 448-456.

"Ishares U.S. Financial Services ETF | IYG". 2021. *Blackrock*. https://www.ishares.com/us/products/239509/ishares-us-financial-services-etf.

Ittiyavirah, Sibi, S. Jones and P. Siddarth. 2013. ‘Analysis of different activation functions using Backpropagation Neural Networks’. Journal of Theoretical and Applied Information Technology 47: 1344-1348.

Keijsers, N. L. W. 2010. ‘Neural Networks’. In Encyclopedia of Movement Disorders, 257–259. Elsevier.

Kingma, Diederik P, and Jimmy Ba. 2014. ‘Adam: A method for stochastic optimization’. arXiv preprint arXiv:1412.6980.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2017. 'ImageNet Classification with Deep Convolutional Neural Networks'. Communications of the ACM 60 (6): 84–90.

Kröse, Ben, and Patrick Van der Smagt. 1993. 'An Introduction to Neural Networks'. Journal of Computer Science 48 (January).

Lee, Hagyeong, and Jongwoo Song. 2019. 'Introduction to Convolutional Neural Network Using Keras; an Understanding from a Statistician'. Communications for Statistical Applications and Methods 26 (6): 591–610.

Lev, Baruch, and S. Ramu Thiagarajan. 1993. 'Fundamental Information Analysis'. Journal Of Accounting Research 31 (2): 190. doi:10.2307/2491270.

Liu, Tao, and Wing Thye Woo. 2019. "Understanding the U.S.-China Trade War." *China Economic Journal*, 1-22.

Liu, Shiyu, Shutao Wang, Chunhai Hu, and Weihong Bi. 2022. 'Determination of Alcohols-Diesel Oil by near Infrared Spectroscopy Based on Gramian Angular Field Image Coding and Deep Learning'. Fuel 309 (February): 122121. https://doi.org/10.1016/j.fuel.2021.122121.

Lopez de Prado, Marcos. 2018. Advances In Financial Machine Learning. 2nd ed. New Jersey: John Wiley & Sons.

Mitchell, Cory. 2021. 'How To Use A Moving Average To Buy Stocks'. Investopedia. https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp.

Moghaddam, Amin Hedayati, Moein Hedayati Moghaddam, and Morteza Esfandyari. 2016. 'Stock Market Index Prediction Using Artificial Neural Network'. Journal of Economics, Finance and Administrative Science 21 (41): 89–93. https://doi.org/10.1016/j.jefas.2016.07.002.

Murphy, John J. 1999. Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. New York: New York Institute of Finance.

Nayak, Aparna, M. M.Manohara Pai, and Radhika M. Pai. 2016. 'Prediction Models for Indian Stock Market'. Procedia Computer Science 89: 441–449. https://doi.org/10.1016/j.procs.2016.06.096.

O'Shea, Keiron, and Ryan Nash. 2015. 'An Introduction to Convolutional Neural Networks'. arXiv preprint arXiv:1511.08458.

Patel, Jigar, Sahil Shah, Priyank Thakkar, and K. Kotecha. 2015. 'Predicting Stock and Stock Price Index Movement Using Trend Deterministic Data Preparation and Machine Learning Techniques'. Expert Systems with Applications 42 (1): 259–268. https://doi.org/10.1016/j.eswa.2014.07.040.

Peachavanish, Ratchata. 2016. 'Stock Selection and Trading Based on Cluster Analysis of Trend and Momentum Indicators'. In Proceedings of the International MultiConference of Engineers and Computer Scientists 2016. Vol. 1. IMECS 2016. http://www.iaeng.org/publication/IMECS2016/IMECS2016_pp317-321.pdf.

Peng, Yaohao, Pedro Henrique Melo Albuquerque, Herbert Kimura, and Cayan Atreio Portela Bárcena Saavedra. 2021. 'Feature Selection and Deep Neural Networks for Stock Price Direction Forecasting Using Technical Analysis Indicators '. Machine Learning with Applications 5 (September): 100060. https://doi.org/10.1016/j.mlwa.2021.100060.

Petrusheva, Nada, and Igor Jordanoski. 2016. 'Comparative Analysis between the Fundamental and Technical Analysis of Stocks'. Journal of Process Management. New Technologies 4: 26–31. https://doi.org/10.5937/JPMNT1602026P.

Rahoma, Abdalhamid, Syed Imtiaz, and Salim Ahmed. 2021. 'Sparse Principal Component Analysis Using Bootstrap Method'. Chemical Engineering Science 246: 116890. https://doi.org/10.1016/j.ces.2021.116890.

Romero, Luis, Joaquim Blesa, Vicenç Puig, Gabriela Cembrano, and Carlos Trapiello. 2020. 'First Results in Leak Localization in Water Distribution Networks Using Graph-Based Clustering and Deep Learning'. IFAC-PapersOnLine, 21st IFAC World Congress, 53 (2): 16691–96. https://doi.org/10.1016/j.ifacol.2020.12.1104

Salkar, Tanishq, Aditya Shinde, Neelaya Tamhankar, and Narendra Bhagat. 2021. 'Algorithmic Trading Using Technical Indicators'. In 2021 International Conference on Communication Information and Computing Technology (ICCICT), 1–6. https://doi.org/10.1109/ICCICT50803.2021.9510135.

Santurkar, Shibani, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. ‘How Does Batch Normalization Help Optimization?’. Proceedings of the 32nd international conference on neural information processing systems: 2488-2498.

Sezer, Omer Berat, and Ahmet Murat Ozbayoglu. 2018. ‘Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach’. Applied Soft Computing 70: 525–538. https://doi.org/10.1016/j.asoc.2018.04.024.

Sezer, Omer Berat, Murat Ozbayoglu, and Erdogan Dogdu. 2017. ‘A Deep Neural-Network Based Stock Trading System Based on Evolutionary Optimized Technical Analysis Parameters’. Procedia Computer Science, 114: 473–80. https://doi.org/10.1016/j.procs.2017.09.031.

Sharma, Sagar. 2017. ‘Epoch vs Batch Size vs Iterations - towards Data Science’. Towards Data Science. Accessed December 10, 2021. https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

Sharma, Siddharth & Sharma, Simone & Athaiya, Anidhya. . 2020. ‘Activation Functions In Neural Networks’. International Journal of Engineering Applied Sciences and Technology: 310-316. 10.33564/IJEAST.2020.v04i12.054.

Shen, Kevin. 2018. ‘Effect of Batch Size on Training Dynamics.’ Mini Distill. Accessed Dezember 3, 2021. https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e.

Shynkevich, Yauheniya, T. M. McGinnity, Sonya A. Coleman, Ammar Belatreche, and Yuhua Li. 2017. ‘Forecasting Price Movements Using Technical Indicators: Investigating the Impact of Varying Input Window Length’. Neurocomputing, Machine learning in finance, 264: 71–88. https://doi.org/10.1016/j.neucom.2016.11.095.

Sim, Hyun, Hae Kim, and Jae Ahn. 2019. ‘Is Deep Learning for Image Recognition Applicable to Stock Market Prediction?’ Complexity 2019: 1–10. https://doi.org/10.1155/2019/4324878.

Speiser, Jaime Lynn, Michael E. Miller, Janet Tooze, and Edward Ip. 2019. ‘A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling‘. Expert Systems with Applications 134: 93–101. https://doi.org/10.1016/j.eswa.2019.05.028.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'. Journal of Machine Learning Research: JMLR 15 (56): 1929–1258.

Thakkar, Ankit, and Kinjal Chaudhari. 2021. 'A Comprehensive Survey on Deep Neural Networks for Stock Market: The Need, Challenges, and Future Directions'. Expert Systems with Applications 177: 114800. https://doi.org/10.1016/j.eswa.2021.114800.

Thakkar, Vignesh, Suman Tewary, and Chandan Chakraborty. 2018. 'Batch Normalization in Convolutional Neural Networks — A Comparative Study with CIFAR-10 Data'. In 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), 1–5. https://doi.org/10.1109/EAIT.2018.8470438.

Tharwat, Alaa. 2016. 'Principal Component Analysis - a Tutorial'. International Journal of Applied Pattern Recognition 3: 197. https://doi.org/10.1504/IJAPR.2016.079733

Tsai, Yun-Cheng, Jun-Hao Chen, and Jun-Jie Wang. 2018. 'Predict Forex Trend via Convolutional Neural Networks'. Journal of Intelligent Systems 29 (1): 941–958. https://doi.org/10.1515/jisys-2018-0074.

Verma, Yugesh. 2021. 'Complete Guide To Dickey-Fuller Test In Time-Series Analysis'. Analytics India Magazine. Accessed December 1, 2021. https://analyticsindiamag.com/complete-guide-to-dickey-fuller-test-in-time-series-analysis/.

Vijh, Mehar, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar. 2020. 'Stock Closing Price Prediction Using Machine Learning Techniques'. Procedia Computer Science 167 (2019): 599–606. https://doi.org/10.1016/j.procs.2020.03.326.

Walasek, Rafał, and Janusz Gajda. 2021. 'Fractional Differentiation and Its Use in Machine Learning'. International Journal of Advances in Engineering Sciences and Applied Mathematics 13 (2–3): 270–277.

Wahab, Noorul, Asifullah Khan, and Yeon Soo Lee. 2017. "Two-Phase Deep Convolutional Neural Network For Reducing Class Skewness In Histopathological Images Based Breast Cancer Detection". Computers In Biology And Medicine 85: 86-97. doi:10.1016/j.compbiomed.2017.

Wang, Zhiguang, and Tim Oates. 2015. 'Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks'. Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, 40–46.

Wehinger, Gert. 2012. "The Financial Industry In The New Regulatory Landscape". *OECD Journal: Financial Market Trends* 2011 (2): 225-249. doi:10.1787/fmt-2011-5k9cswmzqp7d.

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

Xu, Kelvin, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. 'Show, Attend and Tell: Neural Image Caption Generation with Visual Attention'. 32nd International Conference on Machine Learning, ICML 2015 3: 2048–2057.

Yang, Chao-Lung, Chen-Yi Yang, Zhi-Xuan Chen, and Nai-Wei Lo. 2019. 'Multivariate Time Series Data Transformation for Convolutional Neural Network'. In 2019 IEEE/SICE International Symposium on System Integration (SII), 188–192. Paris, France: IEEE. https://doi.org/10.1109/SII.2019.8700425.

Yang, Zhenhua, Kuangrong Hao, Xin Cai, Lei Chen, and Lihong Ren. 2019. 'Prediction of Stock Trading Signal Based on Multi-Indicator Channel Convolutional Neural Networks'. In 2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS), 912–917. IEEE.

Zhang, Dayong, Min Hu, and Qiang Ji. 2020. "Financial Markets Under The Global Pandemic Of COVID-19". Finance Research Letters 36: 1-6. doi:10.1016/j.frl.2020.101528.

Zhang, Jiawei, and Fisher B Gouza. 2018. 'GADAM: genetic-evolutionary ADAM for deep neural network optimization'. arXiv preprint arXiv:1805.07500.

2021. https://www.etf.com/IYG#overview.

# Appendix

## Apendix A: Technical indicators

The table below displays the technical indicators used cross-sectoral. Along with a description, the formulas for calculationg the indicator is provided.

| Type | Technical Indicator | Formula (Sezer and Ozbayoglu 2018, 535; Sim, Kim, and Ahn 2019, 7) |
|---|---|---|
| **Trend** | **Simple moving average (SMA)** calculates the average price over a given period. The indicator is widely used to detmine price trends (Sezer and Ozbayoglu 2018, 535). | $SMA = \frac{C_1 + C_2 + \cdots + C_n}{n}$ **where:** $C_i$ = price of an asset at period i n = the number of periods used for moving average |
| **Trend** | **Exponential moving average (EMA)** calculates a moving average such that greater weights are assigned to more recent values (Sezer and Ozbayoglu 2018, 535). | $EMA = C_t * k + EMA(y) * (1 - k)$ **where:** k = 2÷(n+1) n = number of days in EMA $C_t$ = closing price of an asset today y = yesterday |
| **Momentum** | **Rate of change (ROC)** is a momentum oscillator measuring the speed of changes in price over a given period (Sezer and Ozbayoglu 2018, 536). The indicator is calculated by comparing the current closing price with the closing price *n* periods ago. | $ROC = \frac{(C_t - C_{t-n})}{(C_{t-n})} * 100$ **where:** $C_t$ = closing price of an asset today n = number of periods |
| **Momentum** | **Percentage Price Oscillator (PPO)** is a technical momentum indicator similar to MACD (Sezer and Ozbayoglu 2018, 536). It exhibits the relation of two moving averages in percentage, usually a 26-period and 12-period **EMA**. | $PPO = \frac{(EMA_{n_{short}} - EMA_{n_{long}})}{EMA_{n_{long}}} * 100$ **where:** EMA = Exponential moving average as defined before n = number of periods |
| **Momentum** | The **Relative Strength Index (RSI)** is an oscillating indicator measuring the strength and weaknesses of stock prices or the magnitude of historical price changes, indicating whether stock prices are in the 'overbought' or 'oversold' region (Sezer, Ozbayoglu, and Dogdu 2017a,2; Corporate Finance Institute 2020, 4) | $RSI = 100 - \frac{100}{1 + (\frac{g_n}{l_n})}$ **where:** n = number of periods $g_n$ = average percentage gain during a period of length n $l_n$ = average percentage loss during a period of length n |

| Type | Technical Indicator | **Formula (**Sezer and Ozbayoglu 2018, 535; Sim, Kim, and Ahn 2019, 7) |
|---|---|---|
| **Momentum** | **Know Sure Thing Oscillator (KST)** is a momentum oscillator to make rate-of-change readings easier for traders to interpret (Hayes 2021). | *KST = (RCMA #1×1) + (RCMA #2×2) + (RCMA #3×3) + (RCMA #4×4)*<br>**where:**<br>RCMA #1 = 10-period SMA of 10-period ROC<br>RCMA #2 = 10-period SMA of 15-period ROC<br>RCMA #3 = 10-period SMA of 20-period ROC<br>RCMA #4 = 15-period SMA of 30-period ROC |
| **Momentum** | **Williams % Range** is a momentum-based indicator determining overbought and oversold conditions for stock prices **(**Sezer and Ozbayoglu 2018, 535). | $R = \frac{\max(H) - C}{\max(H) - \min(L)} * -100$<br>**where:**<br>C = Closing price today.<br>max(H) = Highest price in the lookback period n.<br>min(L) = Lowest price in the lookback period n.<br>n = number of periods |
| **Momentum** | **Moving Average Convergence Divergence (MACD)** is a momentum indicator showing the trend of stock prices by representing the relationship between two moving averages of prices. Usually a 26-period and 12-period EMA is applied **(**Sezer and Ozbayoglu 2018, 535). | $MACD = EMA_{n_{long}} - EMA_{n_{short}}$<br>**where:**<br>EMA = Exponnetial moving average<br>n = number of periods |
| **Momentum** | **Commodity Channel Index (CCI)** compare the current price with the average price over a given period of time **(**Sezer and Ozbayoglu 2018, 536). | $CCI = \frac{Typical\ Price - MA}{0.015 * Mean\ Deviation}$<br>**where:**<br>Typical Price = $\sum_{i=1}^{n}(\frac{(H+L+C)}{3})$<br>n= number of periods<br>H = High price today<br>L = Low price today<br>C = Closing price today<br>MA = $\frac{(\sum_{i=1}^{n} Typical\ Price)}{n}$<br>Mean Deviation = $\frac{(\sum_{i=1}^{n} \lvert Typical\ Price - MA \rvert)}{n}$ |

**Appendix B: Classification report of the best performing GADF model**

The following appendix shows the classification report for the best performing GADF model obtained for the Financial Services ETF (IYG).

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Hold | 0.54 | 0.82 | 0.65 | 242 |
| Buy | 0.34 | 0.23 | 0.27 | 131 |
| Sell | 0.37 | 0.11 | 0.17 | 121 |
| accuracy | | | 0.49 | 494 |
| macro avg | 0.41 | 0.38 | 0.36 | 494 |
| weighted avg | 0.44 | 0.49 | 0.43 | 494 |

**Appendix C: Classification report of the best performing GASF model**

The following appendix shows the classification report for the best performing GASF model obtained for the Financial Services ETF (IYG).

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Hold | 0.50 | 0.94 | 0.65 | 242 |
| Buy | 0.27 | 0.02 | 0.04 | 131 |
| Sell | 0.26 | 0.05 | 0.08 | 121 |
| accuracy | | | 0.48 | 494 |
| macro avg | 0.34 | 0.34 | 0.26 | 494 |
| weighted avg | 0.38 | 0.48 | 0.35 | 494 |

**Appendix D: Classification report of the best performing MTF model**

The following appendix shows the classification report for the best performing MTF model obtained for the Financial Services ETF (IYG).

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Hold | 0.50 | 0.85 | 0.63 | 242 |
| Buy | 0.27 | 0.08 | 0.13 | 131 |
| Sell | 0.37 | 0.12 | 0.19 | 121 |
| accuracy | | | 0.47 | 494 |
| macro avg | 0.38 | 0.35 | 0.31 | 494 |
| weighted avg | 0.41 | 0.47 | 0.39 | 494 |

## Appendix E: Model hyperparameters

The table below shows the selected model hyperparameters chosen through the randomised search for each ETF and image encoding type.

| Sector | ETF | Image type | Batch Norm. | Drop-out | Activation | Kernel | Padding | Pooling | Optimizer | Learning rate | Epochs | Batch size | Class weight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Information Technology | VGT | GADF | True | 0.25 | softmax | 3,3 | valid | average | RMSprop | 0.0001 | 150 | 16 | None |
| | | GASF | True | None | sigmoid | 3,3 | valid | max | SGD | 0.001 | 10 | 16 | None |
| | | MTF | True | 0.25 | softmax | 5,5 | same | average | RMSprop | 0.0001 | 100 | 16 | None |
| | XSD | GADF | True | 0.5 | softmax | 3,3 | valid | average | Adam | 0.001 | 50 | 32 | None |
| | | GASF | False | None | sigmoid | 3,3 | same | average | RMSprop | 0.0001 | 75 | 64 | None |
| | | MTF | True | 0.25 | sigmoid | 5,5 | valid | max | SGD | 0.001 | 50 | 16 | None |
| Healthcare | IYH | GADF | False | None | softmax | 3,3 | same | average | RMSprop | 0.001 | 75 | 64 | balanced |
| | | GASF | False | None | relu | 3,3 | valid | max | Adam | 0.0001 | 100 | 16 | balanced |
| | | MTF | True | None | sigmoid | 5,5 | same | average | SGD | 0.01 | 10 | 32 | balanced |
| Energy | S&P 500 Energy | GADF | False | None | sigmoid | 5,5 | same | average | RMSprop | 0.0001 | 100 | 64 | None |
| | | GASF | True | None | sigmoid | 5,5 | same | max | SGD | 0.001 | 50 | 16 | None |
| | | MTF | True | None | softmax | 5,5 | valid | max | Adam | 0.001 | 10 | 32 | balanced |
| Financial Services | IYG | GADF | True | 0.25 | softmax | 3,3 | valid | average | RMSprop | 0.0001 | 100 | 16 | None |
| | | GASF | True | None | sigmoid | 3,3 | valid | max | RMSprop | 0.0001 | 50 | 16 | None |
| | | MTF | True | None | sigmoid | 5,5 | valid | average | SGD | 0.001 | 100 | 16 | None |
| Industrials | VIS | GADF | True | None | softmax | 3,3 | same | max | RMSprop | 0.001 | 25 | 16 | None |
| | | GASF | False | 0.25 | softmax | 3,3 | valid | max | Adam | 0.0001 | 75 | 16 | None |
| | | MTF | False | 0.25 | softmax | 5,5 | same | average | RMSprop | 0.0001 | 100 | 16 | None |
| Oil & Gas | XLE | PXL | False | 0.25 | relu | 3,3 | same | max | Adam | 0.001 | 200 | 64 | balanced |

## Appendix F: Computational and financial performances

The table below summarises the computational and financial performance on the test set for each ETF and image encoding type.

| | | | | | | | Benchmark | | | Labelling (on test set) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Sector** | **ETF** | **Image type** | **Accuracy** | **Macro F1** | **Weighted F1** | **Financial Performance** | **Buy & Hold Return** | **SMA Return** | **MR Return** | **% Buy** | **% Hold** | **% Sell** |
| Information Technology | VGT | GADF | 0.51 | 0.34 | 0.42 | 55.48% | 50.0% | 27.57% | -8.17% | 26.29% | 50.19% | 23.53% |
| | | GASF | 0.49 | 0.31 | 0.40 | 36.03% | | | | | | |
| | | MTF | 0.49 | 0.34 | 0.42 | 16.11% | | | | | | |
| | XSD | GADF | 0.50 | 0.28 | 0.38 | 53.43% | 51.0% | 6.66% | 6.1% | 26.10% | 49.45% | 24.45% |
| | | GASF | 0.44 | 0.34 | 0.40 | 40.23% | | | | | | |
| | | MTF | 0.48 | 0.28 | 0.37 | 19.72% | | | | | | |
| Healthcare | IYH | GADF | 0.50 | 0.28 | 0.37 | 07.45% | 25.00% | 20.51% | 11.08% | 25.31% | 47.77% | 26.92% |
| | | GASF | 0.47 | 0.26 | 0.35 | 28.59% | | | | | | |
| | | MTF | 0.37 | 0.29 | 0.34 | 24,64% | | | | | | |
| Energy | S&P 500 Energy | GADF | 0.51 | 0.46 | 0.49 | 10.66% | -8,00% | -3,90% | 0.06% | 29,00% | 44,00% | 27,00% |
| | | GASF | 0.41 | 0.34 | 0.37 | 2.84% | | | | | | |
| | | MTF | 0.37 | 0.31 | 0.34 | 3.35% | | | | | | |
| Financial Services | IYG | GADF | 0.49 | 0.36 | 0.43 | 18.78% | 16.0% | 16.88% | 21.76% | 26.52% | 48.99% | 24.49% |
| | | GASF | 0.48 | 0.26 | 0.35 | -12.75% | | | | | | |
| | | MTF | 0.47 | 0.31 | 0.39 | -6.25% | | | | | | |
| Industrials | VIS | GADF | 0.41 | 0.38 | 0.40 | 2.98% | 7,00% | 3.76% | 19.61% | 27.54% | 47.16% | 25.30% |
| | | GASF | 0.42 | 0.32 | 0.37 | 4.64% | | | | | | |
| | | MTF | 0.47 | 0.31 | 0.37 | 5.53% | | | | | | |
| Oil & Gas | XLE | PXL | 0.72 | 0.46 | 0.76 | 5.2% | 10.0% | 4.8% | 0,00% | 5,38% | 88,88% | 6,74% |