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

FORECASTING OIL & GAS ETFS' PRICE MOVEMENTS USING CONVOLUTIONAL NEURAL NETWORKS

44832 - Marc Serafin

Work project carried out under the supervision of:

Patrícia Xufre Gonçalves da Silva Casqueiro

17-12-2021

Abstract

Thanks to advances in processing power, we have seen the revival of artificial intelligence after the 1980s, and algorithmic trading has become quite popular in the last two decades. In this paper, a convolutional neural network for image recognition was constructed. The CNN recognises patterns in 2D images generated from financial data and classifies them as BUY, SELL or HOLD. The analysed ETF, XLE, is from the Oil & Gas sector. The results are evaluated computationally and financially and compared to other industries. Overall, the CNN approach seems promising but generally, it was not possible to outperform the Buy&Hold strategy.

Keywords: Forecasting, Deep Learning, Technical Analysis, Convolutional Neural Networks, Oil & Gas Industry

Table of Content

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
GAF	Gramian Angular Fields
GADF	Gramian Angular Differentiation Field
GASF	Gramian Angular Summation Field
MTF	Markov Transition Fields
NN	Neural Network
PXL (method)	Sezer's feature pixelation (method)
RF	Random Forest
SVR	Support Vector Regression

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 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 consisting of a large variety of companies will be used as assets to forecast on, instead of using individual company shares. Moreover, only ETFs 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 and last part focuses on limitations faced by the taken approach and provides an outlook on potential further research topics.

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 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 demonstared. 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).

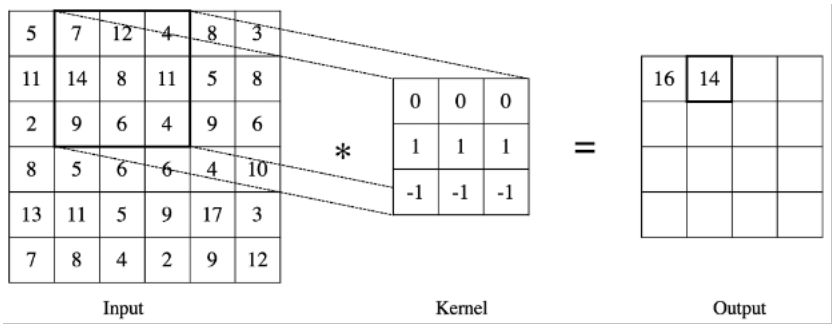


Figure 1 Illustration of the Convolution Operation.

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.

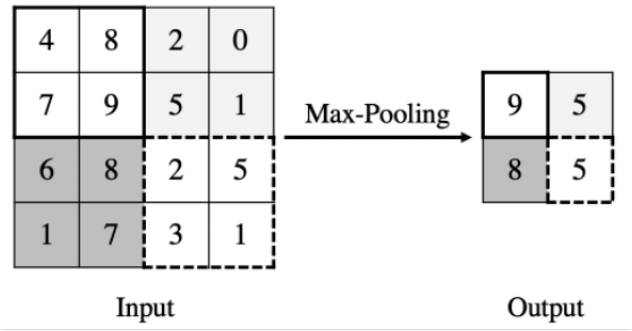


Figure 2 Exemplary Max Pooling Operation

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 = 2) and HOLD (label = 0), based on the price movement during the period after the observation.

3.2.1 Sezer's hill-valley labelling method

Following the approach of Sezer et al. (2018) for financial time series forecasting, the same labelling method will be applied hereafter. The method is directly derived from the pseudocode published in the paper. Essentially, the algorithm considers a sliding window of the close price. Whenever the close price of the middle day of the sliding window (here window = 11 days, middle day = day 6) is at the lowest point - in a valley - the label is classified as BUY. Similarly, if the price of the middle day is on a hill, the label is classified as SELL. Any other data point is classified as HOLD (Sezer and Ozbayoglu 2018, 528).

A notable feature of this labelling approach is that it is more generous than other transaction labelling approaches: On a test dataset, it produced close to 80 % HOLD classifications. That means that the model would trade about once every week. Generally, this is a transaction-intensive strategy and more suitable for swing trading. Similar to day trading, swing trading involves trading strategies that require trades every few days, instead of intra-day trades (Pan 2004, 476). If we extended the length of the sliding window this would most likely change. But the generosity of the algorithm has an advantage. Since there are a few percent more BUY and SELL samples, the class imbalance is not quite as high. A few more samples can help the CNN make a more accurate classification. Also, the CNN should be able to classify easier since the labelling algorithm is relatively simple and therefore should not be quite as difficult for the network to approximate.

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, three trend, sixteen momentum indicators and one volume indicator are combined with different parameter settings. Most technical indicators possess a user defined **interval** as input, affecting the indicators output (Shynkevich et al., 2017, 72). The interval typically refers to the number of raw observations or periods processed by the indicator (Shynkevich et al., 2017, 72). The higher the interval, the more data will be processed. For the three trend indicators, i.e., the moving averages, three different window sizes were chosen respectively. For the sixteen momentum indicators and the volume indicator, one set of parameters was chosen for each. A total of 20 technical indicators are calculated based on the prices 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			No. of Features calculated	Parameters: interval (in days) = [6, 7, 8, …, 27]
	Trend	Momentum	Volume		
Simple moving average (SMA)	x			21	interval
Exponential moving average (EMA)	x			21	interval
Hull moving average (HMA)	x			21	interval
Rate of change (ROC)		x		21	interval
Relative Strength Index (RSI)		x		21	interval
Know Sure Thing Oscillator (KST)		x		21	As defined in appendix.
Williams % Range		x		21	interval
Commodity Channel Index (CCI)		x		21	interval
Directional Movement Index (DMI)		x		21	interval
Stochastic Oscilator (SO)		x		21	interval
Smoothed Relative Strength Index (SRSI)		x		21	interval
Internal Bar Strength (IBS)		x		1	None
Triple exponential average (TRIX)		x		21	interval
Force index (FI)		x		21	interval
Bollinger Bands		x		21	interval
Chaikin Money Flow CMF		x		21	interval
Detrend Price Oscillator (DPO)		x		21	interval
Money Flow Index (MFI)		x		21	interval
Ease of Movement (EOM)			x	21	interval
Chande Momentum Oscillator (CMO)		x		21	interval

Table 2 Technical Indicators and their Parameter Settings

Source: Own illustration

Along with the technical indicators, a set of additional variables are 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 the crude oil price and the exchange rate of Euro and U.S. Dollar.

3.5 Image construction

3.5.1 Sezer's feature pixelation

Following Sezer et al. (2018), the images are constructed according to their pixelation method. Similar to their labelling approach, the concept is quite intuitive. First, the values of the 225 technical indicators are scaled between 0 and 1. Afterwards, they are transformed from a 1-D array per data point to a 2-D array of the form 15 x 15 per data point. Since images have three channels, one for each RGB channel, we copy the array twice and stack the copies along the third axis, resulting in a greyscale image with a shape of 15 x 15 x 3 per data point. Now each feature represents one pixel.

The features must be organised to create interpretable images in the last step. All selected features are arranged side by side according to their name and interval. Moreover, trend indicators are close to each other, momentum and volume indicators likewise. This way, the images can form non-random shapes that the CNN can detect. (Sezer and Ozbayoglu 2018, 529).

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).

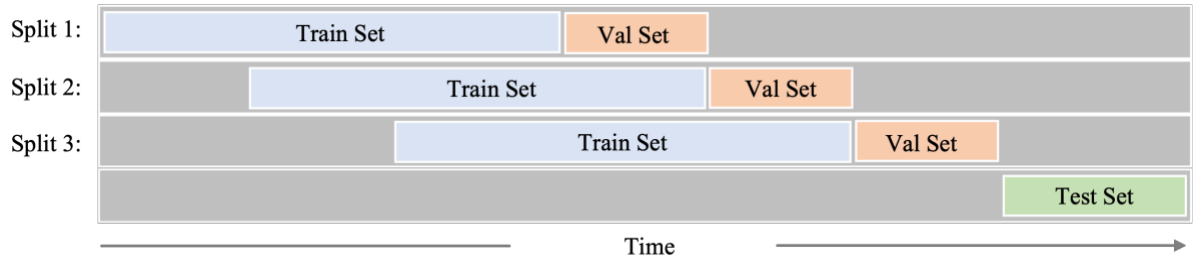


Figure 3 Rolling Forward Cross-Validation

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.

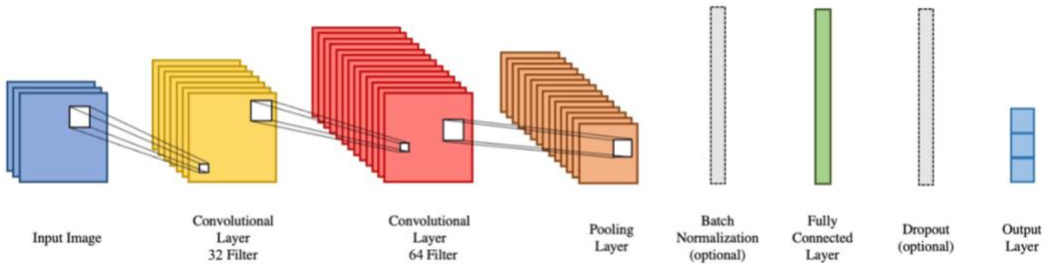


Figure 4 Model Architecture

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 hayperparameter search is utilized. The search includes an optional dropout layer as well as an optional batch normalization layer and optional class weights. 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 use max or average pooling, is used. Lastly, the optimizer, learning rate,

batchsize, the number of epochs and weather balanced class weights should be used 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	100, 200
	Batch Size	16; 32; 64

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	$\mathrm{f(x)} = \max(0, x)$	(1.1)
Sigmoid	$f(x) = \frac{1}{1 + \exp(-x)}$	(1.2)
Tanh	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	(1.3)

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).

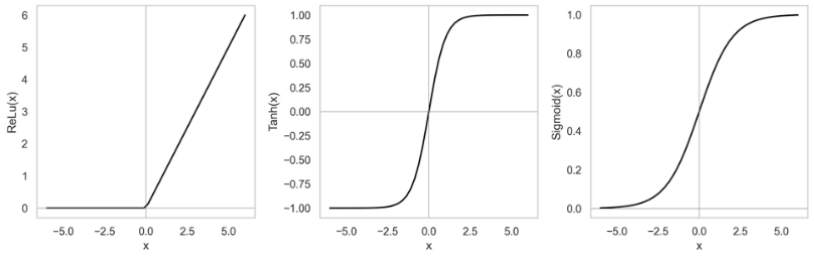


Figure 5 Activation Functions

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).

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)

Activation Function	Formula	
RMSProp	$E(g^2) = \beta E(g^2)_{t-1} + (1-\beta)\left(\frac{C}{w}\right)^2$	(2.1)
	$w_t = w_{t-1} - \frac{\eta}{\sqrt{E(g^2)}}\frac{C}{w}$	(2.2)
	$where\ E(g^2) = Moving\ average\ of\ squared\ gradients$	(2.3)
	$\frac{C}{w}$ = *gradient of cost function with respect to the weight*	(2.4)
	$\eta = learning\ rate$	(2.5)
	$\beta = moving\ average\ parameter$	(2.6)
	$and\ \theta = cost\ function$	(2.7)
Adam	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$	(2.8)
	$v_t = \beta_2 v_{t-1} + (1-\beta_2) g^2{}_t$	(2.9)
	$M_t = \frac{m_t}{1-\beta_1^t}$	(2.10)
	$V_t = \frac{v_t}{1-\beta_2^t}$	(2.11)
	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{V_t}+\epsilon} M_t$	(2.12)
	$With\ \eta = learning\ rate, m = past\ gradient$	(2.13)
	$v = past\ square\ gradient, \beta = decay\ rate$	(2.14)
	$\epsilon = smoothing\ term\ \ and\ \theta = cost\ function$	(2.15)
Stochastic Gradient Descent	$w = w - \eta \Delta Q(w)$	(2.16)
	$Q(w) = \frac{1}{n}\sum_{i=1}^{n} \Delta Q_i(w)$	(2.27)
	$where\ \eta = learning\ rate$	(2.18)
	$(w) = loss\ function$	(2.19)
	$Qw = parameter$	(2.20)

Table 5 Optimisers and Formulas

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

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).

Sample weights

The sample weights are used for computation between layers in a model. The algorithm computes the amounts of true values in the test set and adjusts the weights accordingly. The aim is to find an optimal set of weights ensuring a minimum loss during the network's learning. Sample or class weights are commonly used for imbalanced datasets. (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.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives} \qquad (3.1)$$

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.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3.2}$$

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.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.3}$$

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). 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 Industry analysis

The oil and gas industry (in the following referred to as the O&G) can be described as a sub-sector of the energy industry, which also includes renewable energy. O&G companies are active in various fields of energy production: Exploration, production, refining, shipping, and wholesale.The large companies operate in all areas, while smaller companies usually focus on one part of the chain. For this study we will deal exclusively with American companies to make comparisons across industries uniform (Longwell 2002, 2).

In the U.S., the most significant companies, Exxon Mobil Corp and Chevron Corp, are almost twice the size of the next biggest player in terms of market capitalization (Statista 2021).

4.1.2 Influences

The dependency of O&G companies on global oil demand is demonstrated by the negative oil price during the COVID-19 pandemic, when the entire global economy was forced to shut down its production. During this time, oil price had a drop to a negative value in April 2020. Since then, the oil price has recovered and is now at a six-year high. This trend has been influenced primarily by the OPEC, the Organization of Petroleum Exporting Countries, which has curbed its supply throughout the year (EIA 2021). OPEC and other oil producing companies are another factor to consider for the U.S. O&G industry. Besides global demand, global supply also affects oil prices. When prices are high, even fracking is very economical. On the other hand, high prices accelerate the transition to electric vehicles and other energy sources in the near future. Therefore, the industry keeps the price between these certain thresholds (Mittal 2021, 3).

For O&G companies looking to engage in mergers and acquisitions, ESG scores are becoming increasingly important. Investors in the industry prefer low-carbon barrels, but only a small percentage of transactions to date show the ESG trend (Enverus 2021).

Today, O&G business models are evolving to empower the new energy era and diversification of revenue streams. Some oilfield service providers have already moved to cloud computing and other digital services (Peuch 2020, 1). Other companies are diversifying their revenues by up to 40% by serving renewable energy producers (Bagga 2021, 1). However, given decarbonization targets, oil producers need to focus on and expand in the low-emissions sector. This shift could also bring opportunities. In fuel retail, a new generation of motorists is moving away from brand and price toward user-friendly and convenience-oriented retailers (Mittal 2021, 6). These customers happily engage with services beyond refueling and are willing to pay for the added convenience.

A challenge for the industry is attracting talent. After the price drop in 2020, many workers lost their jobs. Today, only 50% of those lost jobs have been filled. These types of developments damages the industry reputation for job security, while the current core workforce blocks positions for younger talent (Mittal 2021, 6).

4.1.3 O&G ETFs

Three exchange-traded funds were selected for further analysis to find the best fit for forecasting. These ETFs are XLE, IYE and VDN. The shortlist was based on country, composition, and sheer size. These exchange traded funds are american, consist of only O&G companies, and are the largest available within these constraints. After high-level shortlisting, the ETF with the best statistical characteristics is selected for forecasting, taking into account

level of asset, trade volume, return, flow and underlying index or asset class. In general, higher numbers for these variables mean better conditions.

4.1.4 ETF Selection

ETFs should have a reasonable amount of assets for other investors to consider them a good choice. If there is limited demand from other investors, liquidity will be tight and large spreads may occur. This is highly undesirable, especially for algorithmic trading, which is best suited for day or swing trading. These types of trading tend demand frequent transactions and require reliable numbers and low transaction costs (Puelz, Carvalho, and Hahn 2015, 2). Therefore, it seems safe to say that the higher the assets of an ETF, the better. As shown in Table 6, XLE has the most assets at $27.8 billion, compared to $3 billion and $6 billion.

In general, higher trading volume in an exchange-traded fund means it is more liquid and more likely to have a tighter bid-ask spread, regardless of asset class. For popular ETFs, trading volume amounts to millions of shares per day. High volume means there are many buyers and sellers and it is easier to exit the desired price. Again, XLE has the highest trading volume.

As depicted in Table 7, the highest average daily return is offered by XLE, although not by a large margin. In terms of standard deviations, minimums and maximums, all ETFs offer very similar performance.

Having a much higher average volume, it is not surprising that XLE has the highest average daily flow. Only the standard deviation is higher, but that is expected with a much higher average. In addition, XLE is also the oldest exchange traded fund with the most data points.

The underlying asset class ideentical for all three ETFs, and in general, the composites are for the most part identical, although their individual proportions vary. Addressing diversification, all ETFs are focused on a specific industry (oil and gas) and a specific country (USA). This is not ideal from an investor's point of view, but necessary for the comparison.

In all statistics, XLE had the best prerequisites for a forecast. Therefore, it is used for the CNN. Since all ETFs offer large volumes and good prerequisites, any of them would be a suitable choice, nevertheless.

Symbol	Name	Net Assets	90-day Avg. Volume	Largest composite	2nd largest composite	3rd largest composite
XLE	Energy Select Sector SPDR Fund	**$27.8B**	**29,498,182**	Exxon Mobil Corp 22.54%	Chevron Corp 20.27%	Schlumberger NV 4.61%
IYE	iShares U.S. Energy ETF	$2.4B	2,394,910	Exxon Mobil Corp 20.29%	Chevron Corp 16.53%	ConocoPhillips 7.74%
VDE	Vanguard Energy ETF	$6.0B	1,183,901	Exxon Mobil Corp 20.30%	Chevron Corp 16.39%	ConocoPhillips 7.02%

Table 6: Overview of possible Oil & Gas ETFs

Symbol	Mean	Std. Dev.	Min	Max	Start year	End year
Flows in 1000s						
XLE	**14816.82**	12870.16	**7.40**	**99356.70**	1998	2021
IYE	713.96	1211.55	0,00	42603.50	2000	2021
VDE	320.29	**440.99**	0.40	5786.70	2004	2021
Returns						
XLE	**0.000239**	**0.018454**	-0.224910	0.152503	1998	2021
IYE	0.000189	0.018530	-0.231433	**0.218133**	2000	2021
VDE	0.000187	0.019400	**-0.220964**	0.157834	2004	2021

Table 7: Summary statistics of Oil and Gas ETFS

daily Flows (daily trading volumes in 1000 shares as flows) and daily log returns of closing prices ($\log P_t - \log P_{t-1}$)

4.2 Data Prepocessing, feature engineering and image encoding

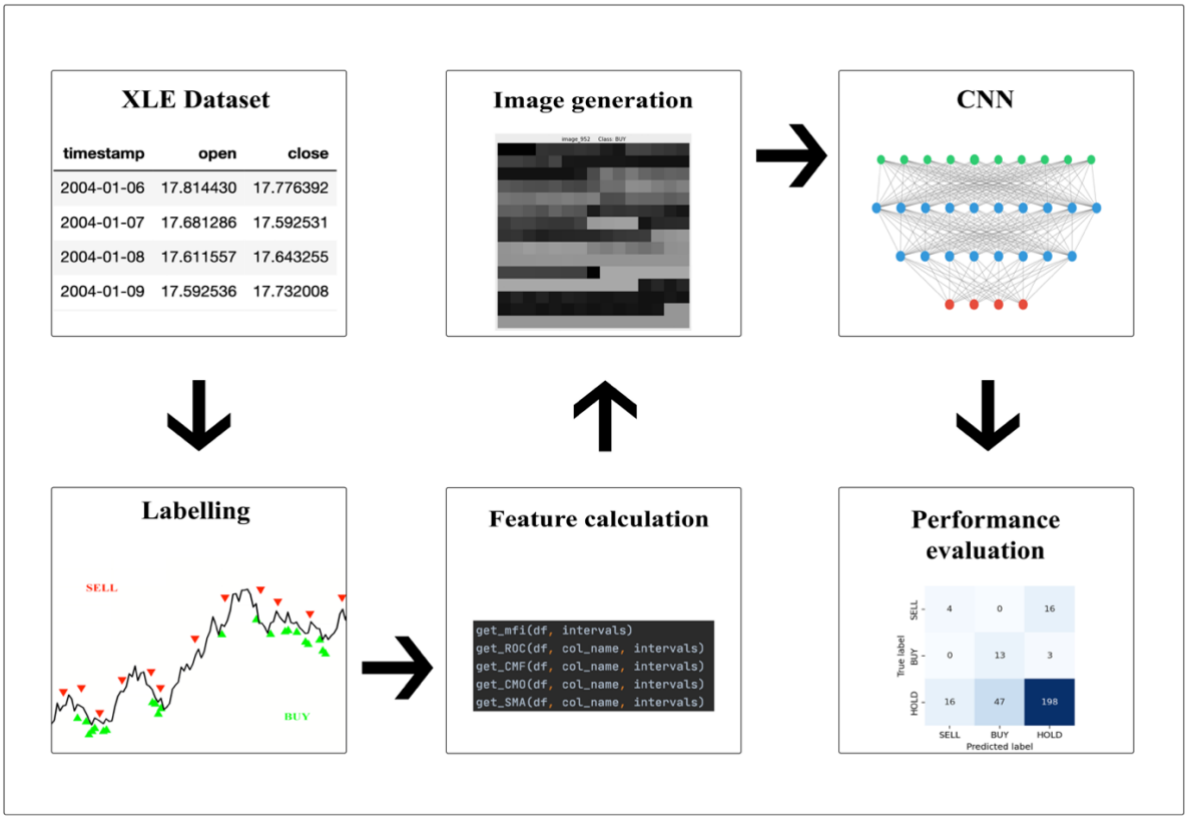


Figure 6: Overview of the implemented methodology

As shown in Figure 6, six steps are performed for the classification model. Retrieving the data, labelling, feature engineering, image generation, training the CNN, and performance evaluation.

4.2.1 XLE Dataset

The XLE data set was retrieved from Fidelity, accessed 11/21/2021. The data points range from the inception date of 1998 to 2021, and consist of daily data points of opening, closing, minimum and maximum prices, and trading volume per day. The data was divided into a training set and a test set. The training data ranged from 2004 to 2018 and was later used for

rolling forward cross validation. The test set covered one year (2019) to provide a reasonable time frame for measuring financial performance.

4.2.2 Labelling

As previously described, Sezer's valley-hill method is used for labelling. Here, valleys of a window of 11 days are classified as SELL, hills as BUY and all other data points as HOLD. One problem with this approach is that the CNN has problems differentiating between the valley data points and points close to the valleys, resulting in many BUY signals when it should be HOLD. This might be specific to the XLE dataset but it is a problem when we want the same amount of SELL and BUY signals. On the test set, the true HOLD signals amount to approximately 90% which seems much but in practice, this still means one transaction every 10 days. Therefore this ratio seems adequate.

4.2.3 Feature engineering

Figure 7: Index Closing prices

Feature calculation

To forecast the XLE ETF, the data was supplemented with the crude oil price CL=F and the Euro/USD exchange rate EURUSD=X. The choice for these indexes was based on the strong dependency of the industry on oil in general, and the strong internationality of the market. The closing prices of the three indices are illustrated in Figure 7. Instead of using only the open and close prices, the same technical indicators were calculated for CL=F and EURUSD=X as for the XLE itself. The full list of indicators can be found in Appendix A. Ultimately, this resulted in a 3% increase in accuracy compared to the model relying only on the XLE data and the open/close prices of the other two indices. The additional data sets are the most notable difference from the approach of Sezer et al. (2018).

Feature selection

The feature selection algorithm mentioned in 3.3.3 selected the best 225 features from the 1212 features for the image, as shown in Table 8. 157 of the 225 features were selected from the XLE dataset. The strongest indicators here are RSI, WR, CCI, FI, and EOM, with all 21 features included in the image. 75 features originate from the CL=F dataset, with FI and EOM being the strongest indicators. For the EURUSD=X dataset, only two DMI features were included in the algorithm. More information can be found in Table 8.

Assuming that an O&G ETF is dependent on the oil price, this distribution is not surprising. Intuitively, O&G stock prices would be more responsive to a decline in oil prices than vice versa, so there may be a slight lag following oil price movements that justifies using this data set for O&G forecasting. Looking at the last data set, it is interesting to see that the exchange rate indicators are almost completely negligible in terms of statistical significance.

After selecting the characteristics, the features are re-sorted. Indicators of the same type but of different intervals are next to each other. Moreover, they are also sorted by their category. This

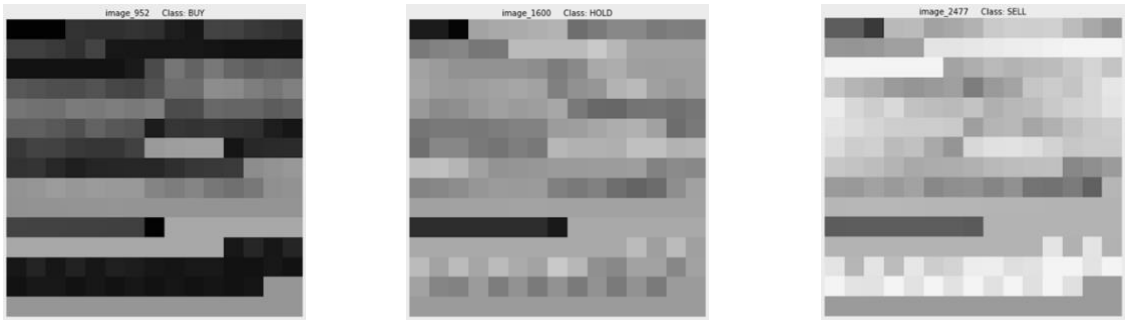
means that trend, momentum and volume indicators are next to their type, e.g. trend next to another trend indicator.

Features	Selected Features from index			No. of features calculated per index
	XLE	EURUSD=X	CL=F	
Simple moving average (SMA)	-	-	-	21
Exponential moving average (EMA)	-	-	-	21
Hull moving average (HMA)	-	-	-	21
Rate of change (ROC)	12	-	-	21
Relative Strength Index (RSI)	21	-	11	21
Know Sure Thing Oscillator (KST)	-	-	-	21
Williams % Range (WR)	21	-	11	21
Commodity Channel Index (CCI)	21	-	9	21
Directional Movement Index (DMI)	-	2	-	21
Stochastic Oscilator (SO)	8	-	-	21
Smoothed Relative Strength Index (SRSI)	4	-	-	21
Internal Bar Strength (IBS)	-	-	-	1
Triple exponential average (TRIX)	-	-	-	21
Force index (FI)	21	-	21	21
Bollinger Bands	-	-	-	21
Chaikin Money Flow (CMF)	6	-	2	21
Detrend Price Oscillator (DPO)	5	-	-	21
Money Flow Index (MFI)	13	-	-	21
Ease of Movement (EOM)	21	-	21	21
Chande Momentum Oscillator (CMO)	4	-	-	21
Close	-	-	-	Raw data
Open	-	-	-	Raw data
Low	-	-	-	Raw data
High	-	-	-	Raw data
Volume	-	-	-	Raw data

Table 8: Selected features

Source: Own illustration

4.2.4 Image creation



BUY HOLD SELL

Figure 8: 15 x 15 images with features encoded in the pixels

Source: Own illustration

Sezer's feature pixelation method described in 3.5.1 is used to create the images. Examples can be seen in Figure 8.

The images in Figure 8 are good representations of how each class looks as an image. It is easy to recognise lines and shapes formed by the indicators. For example, BUY usually has many black areas, HOLD is mostly gray, and SELL has many white areas. However, not all data points look like these representations - and this is where the difficulty for the CNN lies.

Another difficulty for the CNN is recognizing the shapes. This is addressed by sorting the pixels in a meaningful order. From experiments with the images, one could observe that the performance of the CNN significantly drops when we break up these shapes by shuffling the order of the indicators.

These images are always in greyscale. This is the case because for every pixel and any given value $\mathbf{r}, \mathbf{g}, \mathbf{b}$ in the RGB channels, if $\boldsymbol{r} = \boldsymbol{g} = \boldsymbol{b}$, the displayed colour is going to be monochrome. Because we are simply copying the red values to the other two channels, that is exactly the case for each pixel. Hence, the hues will not deviate from the monochrome palette.

4.3 Final Model architecture

A randomised search created the model architecture with the parameters from Table 3. The final model with the best parameters had the architecture from Figure 9:

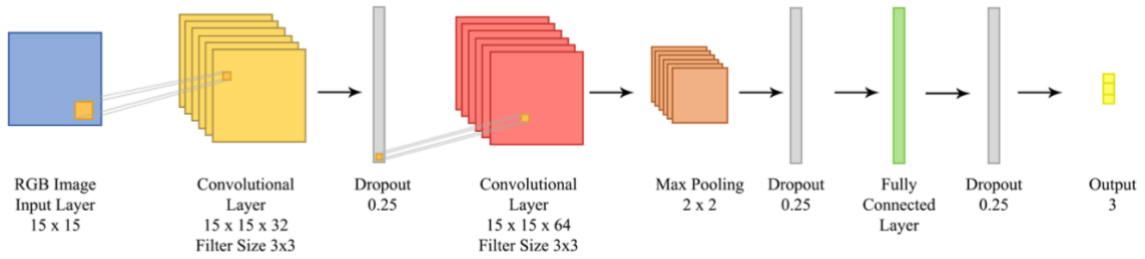


Figure 9: Final model architecture with parameter settings, Source: Own illustration

Two convolutional layers with dimensions 15 x 15 x 32 and 15 x 15 x 64 were implemented for iterations of the grid search. The rate for the dropout layers was 0.25, and the Max method was chosen for pooling. Finally, a fully connected layer was added as the penultimate hidden layer. As an activation function, ‘ReLu’ showed the inner layers' best performance, and ‘Softmax’ was chosen for the output layer. In addition, the learning rate was initialized with a value of 0.001 but can be lowered during training by ReduceLRonPlateau. More information about the final model architecture can be found in Appendix B. The model losses and improvements during training can be seen in Figure 10.

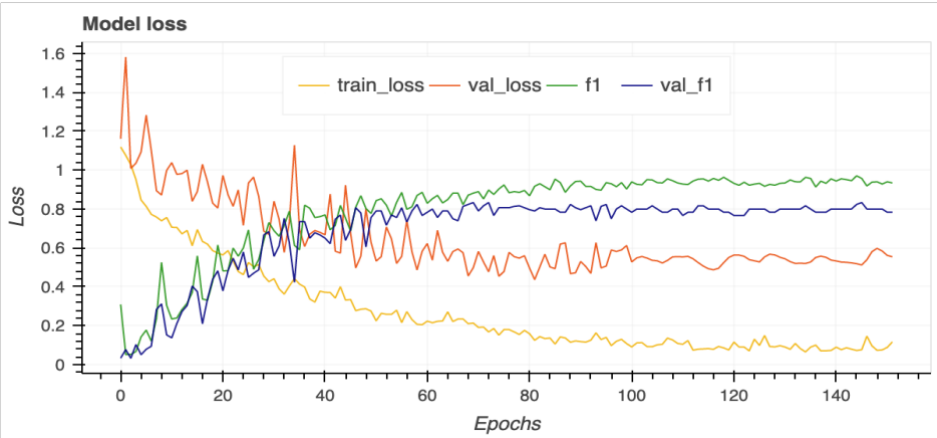


Figure 10 Model loss chart

4.4 Performance evaluation

4.4.1 Test data

The test data consists of the year 2019 with 248 data points for each business day. A period of one year is reasonable for financial performance evaluation. From the financial point of view, it is crucial to have a consecutive dataset. This is due to the fact that we measure the performance within one year. For computational model performance, these constraints would not have been as tight. However, the same test data is used for both performance measures.

4.4.2 Model performance

	Predicted			
		SELL	BUY	HOLD
Actual	SELL	4	0	16
	BUY	0	13	3
	HOLD	16	47	198

Table 9: Confusion Matrix of test data

Total accuracy: 0.72			
	SELL	BUY	HOLD
Recall	0.2	0.81	0.76
Precision	0.2	0.22	0.91
F1 Score	0.2	0.35	0.83

Table 10: Computational evaluation of test data

Analysing the predictive model, the first score to look at is the total accuracy. Although not extremely high, 0.72 (as seen in Table 10) is a useful accuracy that is certainly better than a random distribution. As expected, the model classifies the HOLD labels particularly well. This

is the largest class and thus should be the easiest to predict, especially since the data set is so unbalanced.

The model is also acceptably good at predicting BUY signals. The recall value is very high at 0.81, which means that it almost never misclassifies a true BUY signal. However, it tends to classify a disproportionate number of HOLD signals as BUY with a precision of 0.2. On closer insepection, it seems as if it is hard for the CNN to differentiate the days close to a valley from the valley (as previously explained in 3.5.1). That is the reason for the low precision score.

For the SELL signals, the model performs equally poorly with a value of 0.2 for Recall and Precision. It is however at par with the BUY precision score. Again, the model has difficulty distinguishing between data points close to the hill and the hill. This time it also misclassifies a lot of SELLs as HOLDs, that is why it has three times fewer exit than entry signals.

For algorithmic trading, it is important to have accurate entry points, i.e. BUY and SELL. Here it is especially important that these two are not confused with each other. Fortunately, the model is very good at distinguishing between BUY and SELL. A look at the confusion matrix in Figure 9 shows that it can distinguish these two without any error. Thinking about real-world applications, this fact would prevent many of the worst-case scenarios.

If we tried to catch more true positives of the BUY signals (and improve our Recall), we have to accept that additional HOLD signals would also be classified as entry points. The problem is that the true entry points are so few. This makes them extremely difficult to capture accurately without creating false alarms, reducing the precision score (Sezer and Ozbayoglu 2018, 534).

Considering that stock data is predicted, the model has decent performance overall.

4.4.3 Financial Performance

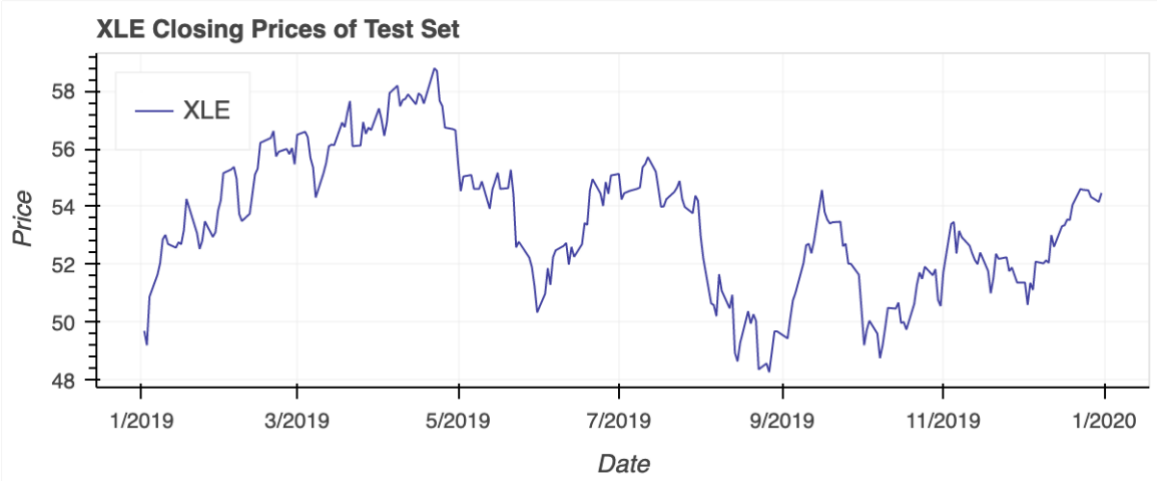


Figure 11: Closing Prices of test set data

To measure the financial performance of the CNN, the algorithm explained in 3.7.2 is applied to the test set, the year 2019. Figure 11 above shows the closing prices of XLE for this year.

In Table 11, we see that the Buy & Hold strategy is the clear winner with a return of 10.0%. Typically, this strategy is hard to beat in the long run. However, over the medium term, e.g., one year, other strategies usually have a chance to outperform it. With SMA and MR's returns below 5%, one could argue that this year may have been a tough year for swing trading and non-B&H strategies. CNN came in second, and while not the best strategy, a 5.2% return is still a good return when the asset is not on an uptrend.

The problem with this test is that the start and end dates are critical. For example, if we had started in April 2019, Buy and Hold would have a slightly negative return and the other algorithms, including CNN, would have a positive return.

To really test the model's performance, we would need to test many years separately and use summary statistics on those measures to get a better idea of how robust the CNN is compared to regular algorithms. The dilemma is that we do not have enough data to test for many years and still have sufficient data for training.

CNN‘s Financial Performance	Buy & Hold Return	SMA Return	MR Return
5.2%	**10.0%**	4.8%	0,0%

Table 11: Financial performance comparison

4.5 Limitations and Implications

This approach, inspired largely by Sezer et al. (2018), shows potential but has some shortcomings in the current implementation. The model itself needs to be optimized or trained with additional data to improve the recall value for the SELL signals. Of course, other improvements would also be desirable. In addition, one could argue that the underlying labelling algorithm could also be improved. It seems to work for the data used in the original paper by Sezer et al. (2018), but for the XLE test set, it was not the best. This could also be due to the model making incorrect predictions. Unfortunately, this cannot be said with certainty because there are two sources of error in this chain: the labelling algorithm itself and the model that replicates the labels.

Furthermore, we could think about adding more datasets to the features. Using the oil price has worked remarkably well, and perhaps other assets can help make predictions, e.g. gold. Also, it would be helpful to train the model with more than one ETF, so we could also use more years for backtesting the strategy. Another option would be to use minute data, which is usually not as readily available. The XLE ETF seems to be a good choice for predictions. It is not very volatile, and the model's accuracy suggests that modelling the labels for the CNN is a manageable task. It would be interesting to see how this model performs on other ETFs, or how it would perform if we trained it with data from other ETFs and tested it again on XLE. IYE or VDE would be obvious choices, but perhaps ETFs from other sectors could benefit from this

model. However, the auxiliary data from the oil price and exchange rate would need to be adjusted.

In addition, the deep learning approach to technical analysis could be combined with other strategies. For example, natural language processing models from tweets or Reddit forums could help identify the sentiment of other traders. Since the stock market is very psychologically driven nowadays, this aspect should not be neglected in a successful trading strategy (Yao and Luo 2009, 669).

Evaluating financial performance still seems to be one of the most challenging parts, so adding more years to the test would be of limited advantage. A more advanced test strategy that includes both long and short positions would further improve the evaluation of the model (Sezer and Ozbayoglu 2018, 535).

5 Performance Comparison and Discussion

In the following section, key findings from the individual analyses conducted in part 4. will be summarised, focusing on common findings regarding the model hyperparameters, the computational performance of the models, 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 worst 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. Returns are also positive in the Oil & Gas sector, where the PXL-based method is applied. 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 return 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, which is also used for model selection. However, there are other evaluation methods that could be considered. In particular, it is important to consider in which cases more emphasis should be placed on either computational or financial performance. 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 mislabeling. 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 labeling approach. If the upper and lower limits are exceeded on the same day, the first labeling 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, such as 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 most cases do not react immediately to changes and new information which in turn makes profits above the market average still possible through sufficient analysis.

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.

First, 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 different 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 suggest 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 to transform 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 since 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.

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.

Bagga, Binny. 2021. 'Rystad Energy: Emerging opportunities and challenges for oilfieldservice players.'

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.

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/.

EIA. 2021. ‘Cushing, OK WTI Spot Prices FOB.’

Enverus. 2021. ‘Enverus intelligence.’

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

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

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.

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, 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.

Longwell, Harry J. 2002. "The future of the oil and gas industry: past approaches, new challenges." World Energy 5 (3): 100-104.

Mittal, Anshu. 2021. ‘Deloitte: 2022 oil and gas industry outlook.’

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.

Mukkamala, Mahesh Chandra, and Matthias Hein. 2017. "Variants of RMSProp and Adagrad with Logarithmic Regret Bounds." Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research. https://proceedings.mlr.press/v70/mukkamala17a.html.

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.

Pan, HP. 2004. "A Swingtum theory of intelligent finance for swing trading and momentum trading." 2004). A revised version submitted to a finance journal.

Peuch, Oliver Le. 2020. "JP Morgan 2020 Energy, Power, and Renewables Conference." Schlumberger.

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.

Puelz, David, Carlos M Carvalho, and P Richard Hahn. 2015. 'Optimal ETF selection for passive investing.'

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?'.

https://proceedings.neurips.cc/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.

Scott, Gordon. 2021. "Technical Indicators." Investopedia US.

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.

Statista. 2021. ‘Leading oil and gas companies in the United States based on market capitalization as of October 2021 (in billion U.S. dollars)’ Financial Times: p. 1.

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.

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.

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.

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

Yao, Shujie, and Dan Luo. 2009. 'The economic psychology of stock market bubbles in China.' World Economy 32 (5): 667-691.

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

Appendix

Apendix A

The table below displays the technical indicators used in the Oil & Gas industry. 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).	$SMA = \frac{C_1 + C_2 + \cdots + C_n}{n}$	(4.1)
		where:	
		C_i = price of an asset at period i	(4.2)
		n = the number of periods used for moving average	(4.3)
Trend	**Exponential moving average (EMA)** calculates a moving average such that greater weights are assigned to more recent values (Sezer and Ozbayoglu 2018, 535).	$EMA = C_t * k + EMA(y) * (1 - k)$	(5.1)
		where:	
		k = 2÷(n+1)	(5.2)
		n = number of days in EMA	(5.3)
		C_t = closing price of an asset today	(5.4)
		y = yesterday	(5.5)
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.	$ROC = \frac{(C_t - C_{t-n})}{(C_{t-n})} * 100$	(6.1)
		where:	
		C_t = closing price of an asset today	(6.2)
		n = number of periods	(6.3)
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	$RSI = 100 - \frac{100}{1 + (\frac{g_n}{l_n})}$	(7.1)
		where:	
		n = number of periods	(7.2)
		g_n = average percentage gain during a period of length n	(7.3)
		l_n = average percentage loss during a period of length n	(7.4)

	stock prices are in the "overbought" or "oversold" region (Sezer, Ozbayoglu, and Dogdu 2017a,2; Corporate Finance Institute 2020, 4)		
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)* **where:** RCMA #1 = 10-period SMA of 10-period ROC RCMA #2 = 10-period SMA of 15-period ROC RCMA #3 = 10-period SMA of 20-period ROC RCMA #4 = 15-period SMA of 30-period ROC	(8.1) (8.2) (8.3) (8.4) (8.5)
Momentum	**Williams % Range** is a momentum-based indicator determining overbought and oversold conditions for stock prices **(**Sezer and Ozbayoglu 2018, 535).	$R = \frac{\max(H) - C}{\max(H) - \min(L)} * -100$ **where:** C = Closing price today. max(H) = Highest price in the lookback period n. min(L) = Lowest price in the lookback period n. n = number of periods	(9.1) (9.2) (9.3) (9.4) (9.5)
Momentum	**Commodity Channel Index (CCI)** compare the current price with the average price over a given period of time **(**Sezer and Ozbayoglu 2018, 536).	$CCI = \frac{Typical\ Price - MA}{0.015 * Mean\ Deviation}$ **where:** Typical Price = $\sum_{i=1}^{n}(\frac{(H+L+C)}{3})$ n= number of periods H = High price today L = Low price today C = Closing price today MA = $\frac{(\sum_{i=1}^{n} Typical\ Price)}{n}$ Mean Deviation = $\frac{(\sum_{i=1}^{n} \lvert Typical\ Price - MA \rvert)}{n}$	(10.1) (10.2) (10.3) (10.4) (10.5) (10.6) (10.7) (10.8)
Momentum	Directional Movement Index **(DMI)** is a momentum indicator that shows in which direction the price is moving. It does this by comparing prior highs and	$+DI = \frac{Smoothed + DM}{ATR} * 100$ $-DI = \frac{Smoothe - +DM}{ATR} * 100$ $DX = \frac{\lvert +DI - -DI \rvert}{\lvert +DI + -DI \rvert} * 100$ **where:**	(11.1) (11.2) (11.3)

	lows and drawing two lines, a positive and a negative.	+DM (Directional Movement) = Current High – PH	(11.4)
		PH = Previous high	(11.5)
		-DM = Previous Low – Current Low	(11.6)
		Smoothed +/-DM = $\sum_{t=1}^{14} DM - \left(\frac{\sum_{t=1}^{14} DM}{14}\right) + CDM$	(11.7)
		CDM = Current DM	(11.8)
		ATR = Average True Range	(11.9)
Momentum	The **Stochastic Oscillator (%K)** is a momentum indicator that compares the closing price of a security to a range of its prices over a certain period of time.	$\%K = 100 * \frac{(C - L14)}{(H14 - L14)}$	(12.1)
		where:	
		C = The most recent closing price	(12.2)
		L14 = The lowest price traded of the 14 previous trading sessions	(12.3)
		H14 = The highest price traded during the same 14-day period	(12.4)
		%K = The current value of the stochastic indicator	(12.5)
Momentum	**Smoothed Relative Strength Index (SRSI)** is a momentum indicator that is essentially a simple moving average of the RSI.	$SRSI = \frac{RSI_1 + RSI_2 + \cdots + RSI_n}{n}$	(13.1)
		where:	
		C_i = RSI at period i	(13.2)
		n = the number of periods used for moving average	(13.3)
Momentum	**Internal Bar Strength (IBS)** is a momentum oscillator that measure the position of the close price relative to the highs and lows.	$IBS = \frac{Close - Low}{High - Low}$	(14.1)
Momentum	**Triple exponential average (TRIX)** is a momentum indicator that shows the change in a moving average (in %) that has been triple-smoothed exponentially. The smoothing is supposed to filter out insignificant price movements.	$EMA1(i) = EMA(Price, N, 1)$	(15.1)
		$TRIX(i) = EMA3(i-1)EMA3(i) - EMA3(i-1)$	(15.2)
		where:	
		Price(i) = Current price	(15.3)
		EMA1(i) = The current value of the Exponential Moving Average	(15.4)
Momentum	**Force Index (FI)** is a momentum oscillator and uses price and volume to determine the amount of strength behind a price	$FI(1) = (CCP - PCP) * VFI(13) = 13 - Period\ EMA\ of\ FI(1)$	(16.1)
		where:	(16.2)
		FI = Force index	(16.3)

	move. The index fluctuatwa between positive and negative territory. It is unbounded meaning the index can go up or down indefinitely.	CCP = Current close price PCP = Prior close price VFI = Volume force index EMA = Exponential moving average	(16.4) (16.5) (16.6)
Momentum	**Chaikin Money Flow (CMF)** is a momentum indicator that measures Money Flow Volume over a set period of time. Money Flow Volume is a metric used to measure the buying and selling pressure of a security.	$Multiplier = \frac{((Close - Low) - (High - close))}{(High - Low)}$ $Money\ Flow\ Volume\ (MFV) = Volume * Multiplier$ $21\ Period\ CMF = \frac{21\ Period\ Sum\ of\ MFV}{21\ Period\ Sum\ of\ Volume}$	(17.1) (17.2) (17.3)
Tren	**Detrended Price Oscillator (DPO)** is an trend oscillator that attempts to estimate the length of cycles in price movements from hills to hills and valleys to valleys. It highlights peaks and troughs in price, which are used to estimate buy and sell points in line with the historical cycle.	$DPO = Price\ from \frac{X}{2} + 1\ periods\ ago - X\ period\ SMA$ **where:** X = Number of periods used for the look-back period SMA = Simple Moving Average	(18.1) (18.2) (18.3)
Momentum	**Money Flow Index (MFI)** is a technical oscillator that uses price and volume data for identifying overbought or oversold signals in an asset. It can also be used to spot divergences which warn of a trend change in price. The oscillator moves between 0 and 100.	$MFI = 100 - \frac{100}{1 + Money\ Flow\ Ratio\ (MFR)}$ **where:** $MFR = \frac{14\ Period\ Positive\ Money\ Flow}{14\ Period\ Negative\ Money\ Flow}$ $Raw\ Money\ Flow = Typical\ Price * Volume$ $Typical\ Price = \frac{High + Low + Close}{3}$	(19.1) (19.2) (19.3) (19.4)

Volume	**Ease of movement (EoM, EMV)** is a volume indicator that attempts to merge a mix of momentum and volume data into one value. The intention is to use this value to discern whether prices are able to rise, or fall, with little resistance in the directional movement.	$Distance\ moved = \left(\frac{High + Low}{2} - \frac{High + Low}{2}\right)$	(20.1)
		$Box\ Ratio = \frac{\frac{Volume}{Scale}}{High - Low}$	(20.2)
		$1 - Period\ EMV = \frac{Distance\ moved}{Box\ Ratio}$	(20.3)
Momentum	**Chande Momentum Oscillator (CMO)** is a momentum oscillator that is calculates a volume weighted moving average of higher and lower closing prices.	$CMO = \frac{sH - sL}{sH + sL} * 100$	(21.1)
		where:	
		$sH = sum\ of\ highter\ closes\ over\ n\ periods$	(21.2)
		$sL = sum\ of\ lower\ closes\ over\ n\ periods$	(21.3)
	Bollinger Bands is a momentum indicator that consists of a set of trendlines located two std. dev. (positively and negatively) away from a SMA of a security's price. They can be adjusted to preference, however.	$BOLU = MA(TP, n) + m * \sigma[TP, n]$	(22.1)
		$BOLD = MA(TP, n) - m * \sigma[TP, n]$	(22.2)
		where:	
		BOLU = Upper Bollinger Band	(22.3)
		BOLD = Lower Bollinger Band	(22.4)
		MA = Moving average	(22.5)
		TP (typical price) = (High+Low+Close)÷3	(22.6)
		n = Number of days in smoothing period (typically 20)	(22.7)
		m = Number of standard deviations (typically 2)	(22.8)
		σ[TP,n] = Standard Deviation over last n periods of TP	(22.9)

Source: (Scott 2021)

Appendix B

The table below shows the selected model parameters chosen through the randomized 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 C

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

							Benchmark			Labeling (on test set)		
Sector	ETF	Image Type	Accuracy	Macro Average (F1)	Weighted Average (F1)	Financial Performance	Buy & Hold Return	SMA Return	MR Return	% Buy	% Hold	% Sell
Information Technology	VGT	GADF	0.5055	0.34	0.42	55.48%	50.0%	27.57%	-8.17%	26.29%	50.19%	23.53%
		GASF	0.4890	0.31	0.39	36.03%	50.0%	27.57%	-8.17%	26.29%	50.19%	23.53%
		MTF	0.4945	0.33	0.41	16.11%	50.0%	27.57%	-8.17%	26.29%	50.19%	23.53%
	XSD	GADF	0.5018	0.28	0.37	53.43%	51.00%	6.66%	6.10%	26.10%	49.45%	24.45%
		GASF	0.4412	0.33	0.39	40.23%	51.00%	6.66%	6.10%	26.10%	49.45%	24.45%
		MTF	0.4816	0.28	0.36	19.72%	51.00%	6.66%	6.10%	26.10%	49.45%	24.45%
Healthcare	IYH	GADF	0.5028	0.28	0.37	07.45%	25.00%	20.51%	11.08%	25.31%	47.77%	26.92%
		GASF	0.4655	0.26	0.35	28.59%	25.00%	20.51%	11.08%	25.31%	47.77%	26.92%
		MTF	0.3706	0.29	0.34	24,64%	25.00%	20.51%	11.08%	25.31%	47.77%	26.92%
Energy	S&P 500 Energy	GADF	0.5102	0.46	0.49	10.66%	-8.00%	-3.90%	0.06%	29.00%	44.00%	27.00%
		GASF	0.4119	0.34	0.37	2.84%	-8.00%	-3.90%	0.06%	29.00%	44.00%	27.00%
		MTF	0.3655	0.31	0.34	3.35%	-8.00%	-3.90%	0.06%	29.00%	44.00%	27.00%
Financial Services	iShares U.S. Financial Services ETF (IYG)	GADF	0.49	0.36	0.43	18.78%						
		GASF	0.48	0.26	0.35	-12.75%	16.0%	16.88%	21.76%	26.52%	48.99%	24.49%
		MTF	0.47	0.31	0.39	-6.25%						
Industrials	VIS	GADF	0.41	0.38	0.40	2.98%						
		GASF	0.42	0.32	0.37	4.64%	7.00%	3.76%	19.61%	27.54%	47.16%	25.30%
		MTF	0.47	0.31	0.37	5.53%						
Oil & Gas	XLE	PXL	0.72	0.46	0.76	5.20%	10.00%	4.80%	0,00%	5,38%	88,88%	6,74%
Average 5 industries		GADF	0.48		0,41							
		GASF	0.44		0,36							
		MTF	0.43		0,36							