MŰEGYETEM 1782

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunication and Media Informatics

Bitcoin Time Series Classification with Modern Deep Learning Methods

Scientific Students' Association Report

Author:

Marcell Schneider

Advisor:

Dr. Bálint Gyires-Tóth

2022

Contents

Kivonat

Idősor osztályozás és előrejelzés a pénzügyi szektorban egy érdekes és folyamatosan növő kutatási terület. Sokféle modelt használnak ilyen idősorok modellezésére, mert az eredmények és információk a területtel kapcsolatban felbecsülhetetlenek. A Transformerek[75] , amik először a természetes nyelvfeldolgozás területén lettek bemutatva, kiválóak adatsorozatok modellezésére. Ennek ellenére, a képességeik pénzügyi szektorban még nincsenek teljesen felmérve.

Munkámban kettő Transformer-alapú modelt tesztelek Bitcoin árfolyam előrejelzéshez és megvizsgálom Transformer Enkóderek hatékonyságát ilyen feladatokban. Az egyik modell egyetlen enkóderből áll, a másik pedig több enkóder egymásra helyezéséből. Ez a kombináció betekintést ad abba, több enkóder mennyivel segíti ezt a fajta osztályozási problémát. Modern és népszerű viszonyítási alapokat használva azt a következtetést vontam le, hogy ezeknek a modelleknek a Bitcoin idősorozat osztályozási képességei alulmaradnak a mai népszerű modellekkel szemben. Azt is megmutatom, hogy, az eredeti publikációval szemben, ezen a területen, Transformer Enkóder modulok egymásra helyezése nem növeli teljesítményt egy modullal szemben.

Bemutatok egy mintaszűrési eljárást is. A szűrés alapja az, hogy kiveszek a tanítóhalmazból olyan a mintákat, amik előre beláthatatlan, külső okoból következtek be. Habár ezek a minták elmélet szerint nem tanulhatóak, ez a szűrési eljárás nem segítette, se nem rontotta a modellek teljesítményét.

Abstract

Time series classification and prediction on financial datasets is an interesting and expanding area of research. There are numerous model types proposed for such tasks because the results and insights are invaluable. Transformers[75], first introduced in the field of Natural Language Processing, excel at modeling sequential data. Despite this, the capabilities of Transformers regarding financial time series tasks are not yet fully explored.
In this work, I present two Transformer-based architectures for Bitcoin time series classification. I investigate the effectiveness of a Transformer Encoder block in such predictions. The other model is a stacked version of such encoders. This combination gives insights into how beneficial it is to stack Transformer Encoders for tasks like this. I evaluate their performance against standard baselines and show that their predictions are inferior to popular time series predicting models. I also show that stacking encoder blocks does not yield further performance, in contrast to the original paper.
I also propose a target filtering method. This technique filters training data so that the model is made to learn fewer random events and patterns which are due to external factors. Although, in theory, such patterns are not learnable, this technique did not significantly boost or degrade performance.

Chapter 1

Introduction

A paper Nakamoto [56] about the invention of Bitcoin and cryptocurrencies introduced a new type of payment system. Up until then electronic payments always had to rely on a trusted third party to carry out and validate the transactions. Not only does this mechanism affect transaction costs but it also increases the need for trust and information from and about the participants. The proposal of the paper is about a public network that can validate and record payments across participants without a third party. The main idea is a blockchain where blocks of recorded transactions are chained together serving as a public history of the network. With this, virtual coins can be possessed by anyone and they can act like currencies between parties.

Three main characteristics of cryptocurrencies are exposed to external events, low signal-to-noise ratio, and high volatility.

Events affecting prices can range from small agreements, social media posts, or as seen more recently, pandemics or wars (Figure 1.1). These can cause jumps in exchange rates or in extreme cases suddenly redefine the trend and actors' trust in the market at any time.

The currency is extremely exposed to every kind of input mentioned above and also is fairly new so the market is not yet fully developed. Partly because of this, its signal-to-noise ratio is even higher than those of regular currencies. This difference is not negligible, the minimum and maximum of the days can differ by tens of percent (Figure 1.2), seemingly without any news that would have such a big impact. This could also mean that trends and future directions may be easier predicted than the next day's exchange rate.

Neural networks are evolving at an unimaginable speed and newer and newer architectures are being invented. One of the recent groundbreaking innovations was the Transformer model. They were first applied to Natural Language Processing (NLP) tasks but reached state-of-art results in image and time series problems.

The scientific literature regarding the combinations of time series classification tasks and Transformers is very underdeveloped especially when taking a look at problems regarding

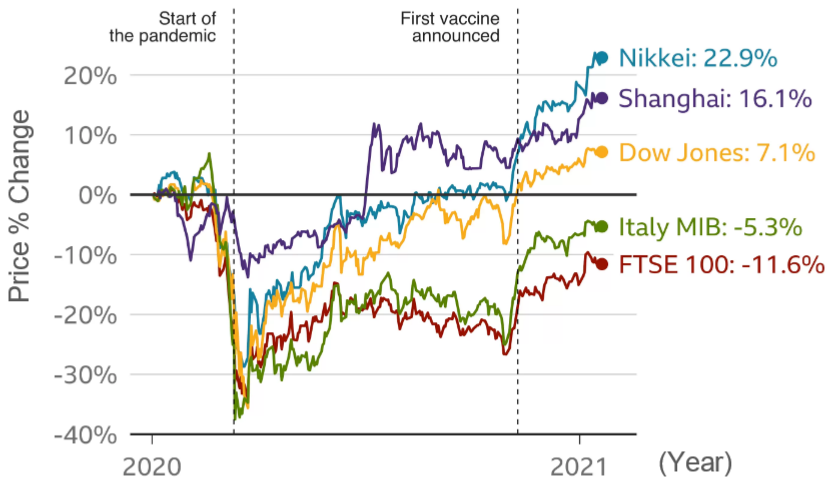

Figure 1.1: Reaction of the stock market to COVID-19 news (adapted from:[37])

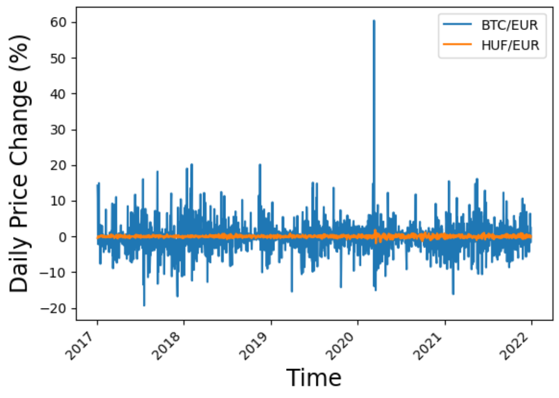

Figure 1.2: Bitcoin and HUF fluctuation of the past few years

the financial domain. In my work, I will test the performance of Transformer-based architectures on Bitcoin data that involve social media aspects as well. During the evaluation, I will also use some of the standard models to provide baselines and reference points. With these, I hope to gain and give better insights into the performance of Transformers in tasks like these.

Chapter 2

Background

This chapter is about the background of cryptocurrencies and time series in general. It gives information about the brief history of cryptocurrencies, the mechanisms of blockchain technology, and the main characteristics of time series. It ends with discussing the main challenges and obstacles of this area of research.

2.1 Bitcoin Blockchain

The goal of the Bitcoin network [56] is to provide an alternative to regular currencies where parties are anonymous and transactions do not require a third party. The backbone of the solution is cryptography and miners. Anybody can be a miner on the network and their job is to provide computing power to the network which is needed to automatically validate transactions. In return, miners get rewards(i.e. Bitcoin) for helping to process payments. In order to record a transaction the following steps have to be completed:

- The payee has to get the unique digital signature[1] of the payer to authenticate the action.
- The new transaction is then broadcast to the network where miners are listening for them.
- The miners of the network collect these unprocessed transactions and solve a cryptographic task related to them.
- The first miner to solve the task can broadcast their easily-verifiable solution to the network trying to add the given block - an unprocessed group of transactions - to the chain. The miner also gets a reward for the work. This transaction is added to the respective block.
- If the broadcast block is correct, everyone appends it to their locally stored chain version.

[1]This signature is unique per payment so it cannot be reused in the future.

The above-mentioned cryptographic task is to find a number that when it is concatenated with information of the given block would yield a SHA256 hash that starts with a given number of zeros. The number of preceding zeros grows with the computing power of the network. The reason for this is to regulate the speed at which the blocks are added to the network - always kept at around ten minutes[2].

Since everyone can submit blocks to the network different versions of the blockchain may arise. To combat this everyone stores the longest-known version of the chain. If there is a tie, the community of the network waits for the next block and keeps the longer one. This means that recording false, invalid transactions on the blockchain would require the attackers to be consistently faster in solving these tasks than the others in the network to build a longer blockchain. This is statistically impossible as long as the majority of the community of the network does not consist of attackers.

The reward for a correct solution gets less with time. This means that mining will be less and less profitable. The last Bitcoin is projected to be given out around 2040 with the majority (19 million out of 21 million total) of the coins already in circulation.

2.2 Time Series

Scientific literature differentiates two main types of time series. Univariate time series consists of a single variable and this variable x. When observed through time this variable takes up different values so the series can be described as $x_0, x_1, ..., x_i, .., x_s$ which may be compressed in a single one-dimensional vector $\boldsymbol{x}$ of length s where s is the number of observed samples. A good example of univariate time series modeling may be the work described in Abdel-Aal and Al-Garni [4]. The authors use univariate time series analysis to predict the energy consumption of Saudi Arabia. The task was autoregressive, meaning the past energy consumption values were used to predict future values.

Multivariate time series can be described as multiple univariate time series. Each time steps have n features which give different information about the respective time step. Mathematically, each time step can be described as a $\boldsymbol{t_i}$ vector of length n. The elements of the vector are the values of each feature. This results in a data matrix of size sxn where s is the number of observations. Inherently, each column of this matrix can be viewed as a univariate time series that describes the changes of a feature through time. Multivariate time series usually enable the use of more complex models since the amount of information to be processed by the model is usually larger. Fields of application of multivariate time series are even wider than that of univariates'. A classic example of multivariate time series use is different kinds of weather forecasting. Authors of Qing and Niu [62] use, amongst others, wind speed, temperature, humidity, and calendar information to predict solar irradiance near the equator.

[2]This is a main drawback of the Bitcoin network. Other crypto-currency networks usually process transactions faster.

Giving more information for a model, however, is not always beneficial. If the provided data for example has a lot of non-predictive features (i.e. is not helpful in predicting the target) the performance of the model may decrease (- the model's performance decline is largely dependent on its robustness against useless attributes). Another reason may be that the input data has a lot of predicting features, however, they have really high correlations between them. In this case, giving the model only a subset of features may be preferable. This phenomenon was shown in Du Preez and Witt [20] where the authors found that multivariate models were consistently outperformed by univariate counterparts in predicting monthly tourism numbers.

There is a wide variety of time series-related tasks. The two main directions of scientific literature are classification and regression. Given a time series $\boldsymbol{T}$ the task might be to predict the exact future values of $\boldsymbol{T}$. The above already mentioned work of Qing and Niu [62] falls under this category but so does the recent work of Arora et al. [8]. The authors used deep learning methods to predict the spread of COVID-19 cases in different areas of India during the early stages of the pandemic. This application is also a strong example of real-life use cases of time series predictions.

The other branch of the field, time series classification, is about categorizing a time series based on its values. The use cases range from classifying the current state of a system to predicting the trends of a time series. Luz et al. [53] studied heartbeats of patients to detect heart arrhythmias where the basis of the classification was a multivariate time series where features described different aspects of heartbeats. Zhang et al. [80] generated classes from stock market closing prices which described different trend types - *Up, Down, Flat, Unknown*. After this, a multivariate time series was fed to a model to classify the coming trends of the market. These two papers show the wide variety and different ideas in time series classification.

2.3 Challenges

Multiple challenges need to be overcome to model crypto-currency price changes. This chapter summarizes the main obstacles in the field. Cryptocurrencies share a lot of mutual traits with regular stocks and currencies in terms of behavior. In this chapter when describing a feature of stocks and currencies, they are also meant for crypto-currencies.

2.3.1 Underdeveloped Market

The number of active participants on the market is rapidly increasing, hitting a million addresses in 2018. These participants trade Bitcoin to their own decisions. The market is still underdeveloped with a lot of actors constantly shaping their judgments of the technology. States and national banks have varying opinions of crypto-currencies. Some states already accept Bitcoin as their national currency [38] and other states ban the use of them [39].

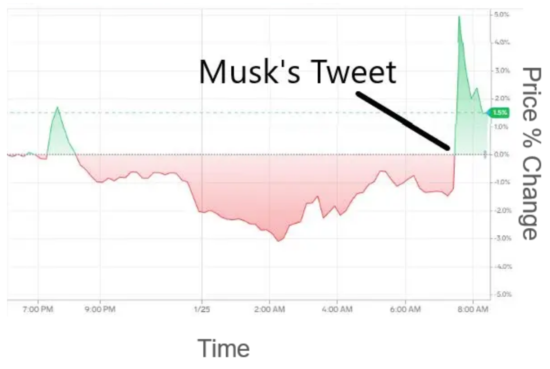

Figure 2.1: Price change of another crypto-currency, DogeCoin, to one of Elon Musk's tweets (adapted from: [24]).

This fact worsens the predictability of the exchange rates and news can greatly impact not just the prices but the future of the currency as well.

2.3.2 High Exposure to External Events

The influence of news from the political and economical domain dominates the stock market trends for obvious reasons. These events are unforeseeable and defining in terms of prices. Events may include social media hype or discussions which also can have a great impact on stocks. A good example of this may be a short squeeze generated by the users of a subreddit during the winter of 2021 [40]. They managed to increase the stock prices of GameStop Corp. by more than 1800%. This caused losses, above 1 billion dollars for hedge funds. The crypto market is even more vulnerable to such events. Prices can change several percents in reaction to tweets or other social media activity from celebrities (Figure 2.1) or other well-known people.

2.3.3 Inherently Imbalanced Classes

In my work, I attempt to classify the coming trends of Bitcoin. Since the price of Bitcoin has an all-time positive trend (i.e. it was worth 0.0 USD and now some p positive amount) it means that positive labels will have a majority over negative samples. As a consequence, the classification task has imbalanced class distribution. This phenomenon could result in the models learning to classify all samples as positives, reaching relatively good accuracy. Not only does this harden training but also precise evaluation.

2.3.4 General Randomness

According to the Efficient-market hypothesis, originating back to Fama [22], there are three types of market efficiency. Under the weak form, the hypothesis means that the market's past prices are all incorporated in the present price and they do not help predict the future. Semi-strong is stronger in the sense that all current public information is encoded in the prices. Furthermore, a market has strong efficiency if all private information is in the stock prices as well. This means no advantage can be gained from *any* information to predict future prices.

The authors in Fama [22] argued that the financial market is unpredictable from past values and the scientific literature categorizes the stock market into semi-strong form, meaning it reacts to new information rapidly.

Since the Bitcoin market is still not fully developed the scientific literature about its market efficiency has mixed opinions. Authors in Burniske and White [13] for example argue that Bitcoin may be a new asset class. This opinion is shared by Dyhrberg [21] who put BTC between USD and gold in terms of Medium of Exchange[3] - Store of Value[4] scale.

The general consensus nevertheless remains that the Bitcoin market is getting more efficient over time, implying harder and harder predictability.

Conducting the Dickey-Fuller test (Dickey and Fuller [19]) on the daily closing prices of BTC/USD results in a p-value of 0.4189. This means that this time series can well be modeled by a random walk and is unpredictable in this sense.

2.3.5 Data Deficiency

Another aspect of the difficulties of this domain is the number of data points. The first crypto-currency was invented in 2008. Reliable data with multiple features is only available since around 2010. If we decide to sample the time series daily that means that we only have approximately 4000 data points. Deep learning methods scale incredibly well with the amount of data that is used for training. Several results are supporting the idea that machine learning in general is a data-driven process.Banko and Brill [12] showed that different algorithms perform very similarly on language processing tasks if enough data is provided. Halevy et al. [29] also argued that the amount of data fed to an algorithm is highly result-defining. The data necessary to train a neural network is heavily dependent on the complexity (e.g. number of parameters) and the dimensionality of the input. The reason for the latter is the curse of dimensionality. The more dimensions the input has, the rarer the data points become in the input space. This implies that fitting a hyperdimensional function to the data points with any algorithm becomes much harder. An average deep learning model requires the magnitude of thousands of data points as the minimum for successful training.

The observations and ideas above in this section mean that crypto-currency prediction and trend classification is an extraordinarily difficult area of research.

[3]An asset type that is widely used for trading

[4]An asset type that keeps its value over time

Chapter 3

Related Work

Since time series analysis is an extremely wide field, numerous models have been proposed to model such data. This chapter provides an overview of often-used models and techniques. I mainly focus on works and methods related to financial time series since Bitcoin predictions fall under this category.

3.1 Neural Networks For Time Series Analysis

Neural networks have been gaining more and more attention in the past decades. They provide state-of-art performance in several machine learning fields and time series classification is no different. Neural networks are universal approximators [36] meaning they can model a wide variety of functions. If you consider a multidimensional input $\boldsymbol{x}$, in a multivariate time series problem for example, that has a target $\boldsymbol{y}$ then the goal is to approximate this hyperdimensional function $f(x)$ with a model. Neural networks have a huge number of parameters. The main idea behind training a neural network is to configure these parameters so that the function $\boldsymbol{\hat{f}(x)}$ that describes the model, approximates $f(x)$ as precisely as possible. The main method for training such a network is usually backpropagation [67] which uses gradients to learn the right values of the parameters.

3.1.1 Multilayer Perceptrons

Multilayer Perceptrons [26], or MLPs, were the first neural networks. They are layers of perceptrons [66] stacked on each other. The first or input layer is given the inputs of the model. This layer applies a transformation function $t_0(x)$ to the inputs and forwards it to the next layer. Each layer's input is the previous layer's output until the data is forwarded to the last layer. Its output $\boldsymbol{\hat{y}}$ means the model's output for the respective input $\boldsymbol{x}$. This transformation can be described according to the following equation:

$$\boldsymbol{y} = t_d(t_{d-1}(...(t_1(t_0(\boldsymbol{x}))))) \tag{3.1}$$

where $\boldsymbol{y}$ is the model's output for the $\boldsymbol{x}$ vector, d is the number of layers of the network and t_i is the *ith* layer's transformation function.

This architecture is quite powerful however it does not perform at time series-related tasks very well compared to other alternatives. The main reason for this is statelessness. Time series modelling is usually rich in temporal connections (i.e. time steps with varying distances usually correlate) and it requires some kind of memory from the model to remember previous time inputs.

3.1.2 Recurrent and Convolutional Neural Networks

Recurrent Neural Networks (RNNs) are a special type of neural networks that excel at modelling sequential data. To have a kind of memory, the nodes in Recurrent Neural Networks (RNNs) process the input sequentially where at each time step the output or a hidden state of the previous time step is also fed to the nodes. This enables the model to learn temporal connections in the data. This sequentiality also shows if the network is *unrolled* in time which is useful during training - that is why the algorithm to train such a model is called backpropagation through time [77].

Long Short-Term Memory or LSTMs are one of the most widely used RNN architectures. They were first introduced by Hochreiter and Schmidhuber [34] and then the model gradually improved[28] [27] and was tailored for different tasks like speech recognition [70]. LSTMs are applicable to any sequential data-related tasks and stock market analysis is no different.

Chen et al. [14] carried out TSC experiments with LSTM on Chinese stock markets. They used the previous N days to predict the earning rates of the coming 3 days. Their LSTM model was almost twice as accurate (14.3% vs 27.2%) as the random classifier (7 classes total).

Authors in Nelson et al. [57] applied a rolling window model on some stocks of the Brazilian stock exchange. They implemented a trading algorithm that buys 100 stocks if the model predicts the price will go up (only binary classification) and the profit is calculated by the difference between the next day's and today's closing price. The LSTM model brought hypothetical profits in all five stocks consistently.

- Buy and hold: buy on the first day, sell on the last
- Optimistic: buys the stock if this day it went up, and sells tomorrow
- Random trader, trading 100 stocks randomly each day.

The LSTM also had about 54% accuracy in the binary classification of price directions, the highest of the examined models.

McNally et al. [54] compared LSTMs with regular RNNs and ARIMA models on Bitcoin predictions. The task was also to predict the coming trends of Bitcoin closing prices (*Up,* *Down, Flat*). LSTMs and RNNs were the best models out of the three. These two however had similar results in RMSE, with LSTM having slightly better performance. Since the class distributions of labels were imbalanced the authors also calculated specificity, precision and sensitivity. Since RNN results were better in some metrics above, this study suggests LSTMs are not a huge (if at all) improvement over regular RNN architectures in this task. Additionally, LSTM training times were much larger which further decreases the relative benefits of LSTM over regular RNN.

In a study by Chen et al. [16] the authors investigated the predictive power of economical indicators versus pure autoregressive methods when predicting Bitcoin prices. They found that the predictive power of economical statistics is decisively stronger than just future prices. Additionally, their results showed that LSTMs outperformed ARIMA and Support Vector Regression [17][72] models in autoregressive tasks.

Convolutional Neural Networks[49], (CNNs), are incredibly powerful models in the field of image processing. Convolutional layers slide a 2D kernel over an image learning local patterns while producing weighted sums of the values under the kernel. The weights have to be learnt during training and a 2D convolutional layer produces a 2D output of the results. Stacking multiple convolutional layers on top of each other can enable the model to learn larger and larger patterns on the image. This simple technique held state-of-art results[48][31][73] in image processing for a long time.

CNNs however can also be used for time series tasks. 1D convolution can be performed on univariate time series. The difference is a 1D kernel which produces 1D outputs. This technique was popularized in a famous paper by Oord et al. [58] in speech generation. Multivariate time series can be represented by an $s\text{x}f$ matrix where s and f are the number of samples and features respectively. This enables the use of 2D convolutional layers similar to image processing.

Tsantekidis et al. [74] applied a model consisting of 2D and 1D convolutional layers to a stock market TSC problem. The multivariate stock data was about nordic companies. The authors created three classes - *Up, Down, Flat* - for future prices. The CNN model had better results than its baselines, namely SVMs[17] and MLPs.

Chen and He [15] experimented with 1D convolutional networks on several stocks. Stocks included GOOGL, APPL, IBM, and several others from the United States. Their target was also binary predicting whether the prices will go up or down the next day. They used 1D convolution for each feature in the multivariate input vector. Their result showed that such architecture can compete with LSTMs. Their model has better results in accuracy and recall (i.e. $\frac{\#\text{TruePositives}}{\#\text{TruePositives}+\#\text{FalseNegatives}}$) in the majority of stocks and equal results regarding F1[1] scores.

Hybrid LSTM and CNN models are more and more common in the literature. Combining the strengths of the two often outperform the individual model types. Alonso-Monsalve et al. [6] implemented a CNN-LSTM hybrid model and evaluated its performance on 6

[1]harmonic mean of precision and recall where precision is $\frac{\#\text{TruePositives}}{\#\text{TruePositives}+\#\text{FalsePositives}}$

crypto-currencies. In a binary classification task regarding the coming trends of the respective currencies, Hybrid LSTM-CNN outperformed the CNN baseline. Evaluating the hybrid model's performance against an LSTM baseline as well would have been beneficial for exact comparison but the hybrid model's 5% accuracy increase over the CNN baseline is useful information nevertheless. Livieris et al. [52] also carried out experiments with different hybrid architectures over multiple crypto-currencies. They found that hybrid models with different structures perform statistically the same.

3.1.3 Transformers

Transformers were introduced in a groundbreaking paper by Vaswani et al. [75]. The Transformer is an encoder-decoder-based model. The idea behind encoder-decoder architectures is to make the encoder encode the input in a meaningful way that can be processed by the decoder later. Transformer encoders and decoders consist of similar blocks put on each other. These blocks are relatively simple stacks of attention, normalization, and feed-forward layers with skip connections[2]. The attention mechanism measures similarity (e.g. cosine similarity) between two vectors and this lets the model focus on relevant (similar) areas of the input while processing parts of it. This implies that it works best with embeddings where vector representations can be learned. Since it only focuses on vector similarity it loses the sense of time and positions of the vectors. For this reason, positional encoding has to be applied where vectors are added terms that encode the relative positions of the respective vectors in sequential data. The architecture of the model can be seen in Figure 3.1. The encoder block is highlighted with red because it is a key component of the tested Transformer-based model in my work.

Models based on this work reached state-of-art performance in neural machine translation (NMT) [18][61] and image generation with natural language processing (NLP) [69][64]. The main characteristic of these models is the efficient use of the attention mechanism invented by Bahdanau et al. [10]. Transformers work very well with sequential data, however, the limits of this architecture regarding time series-related tasks have not been fully explored by the scientific community. Although LSTMs with attention mechanism is widely used by the time series community [50][63], well-documented Transformer experiments in the financial domain are quite rare, especially regarding BTC TSC.

Zerveas et al. [79] investigated transformers with other baselines including LSTMs and CNNs. They evaluated the models on several TSC eleven datasets and experimented with an unsupervised pretraining step before training to learn useful latent representations of features. Transformers both with and without a pretraining step outperformed the baselines on average ranking on the datasets.

Zhou et al. [81] introduced an informer network which is a transformer based architecture designed for long sequence time series forecasting. Not only is it faster at inference

[2]Connecting layers directly with skipping layers in-between.

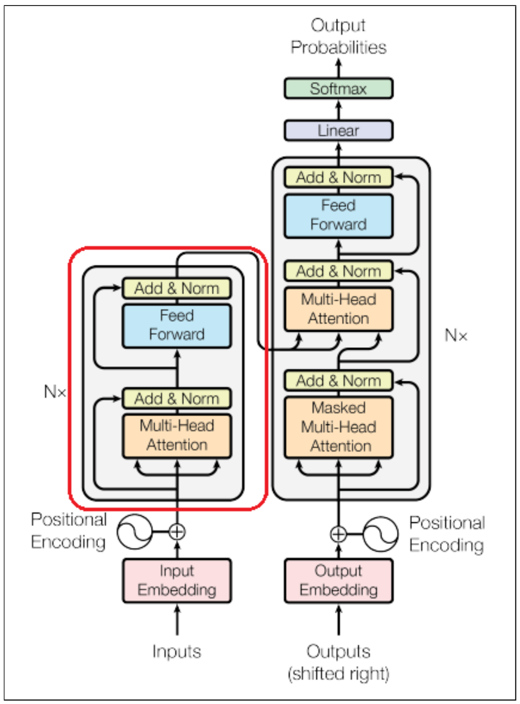

Figure 3.1: The Transformer architecture. The encoder block is highlighted in red. (adapted from: Vaswani et al. [75])

time than original transformers with less memory usage but it also outperformed LSTM baselines (and others) on the tested four datasets.

Liu et al. [51] also experimented with Transformer based models in TSC. They used various datasets and several baselines and their results also showed Transformer superiority on the majority of the thirteen datasets.

The problem however remains the same - none of the authors in the previously mentioned two papers used a dataset from the financial domain, not mentioning the crypto-currencies. This is supported by the fact that in review by Ismail Fawaz et al. [43] from 2019 where the authors summarize the directions and results of the TSC field, Transformers are never mentioned.

There are studies using Transformers for sentiment analysis in this domain [59] [23]. The authors in these papers use Transformers to create input features for other deep learning models (such as LSTMs) via news/social media sentiment analysis. However, in this work, I try to use Transformer-based architectures for BTC TSC and evaluate this against other networks.

Recent work from Ji et al. [44] compared several techniques and models for Bitcoin TSC. Although Transformers were not present in the tested models, the authors wrote at the end of the paper that it would be an interesting idea to explore the performance of Transformers in this area.

3.2 Statistical Models

Moving Average (MA) models are simpler yet powerful tools for univariate time series. They work best for time series that have short-term autocorrelations without changing mean and variance. They model a time series according to Equation 3.2.

$$\begin{aligned} y_t &= \eta + \sum_{i=1}^{p} \phi_i * \varepsilon_{t-i} + \varepsilon \\ \hat{y_t} &= \eta + \sum_{i=1}^{o} \phi_i * \varepsilon_{t-i} \end{aligned} \tag{3.2}$$

where the $\hat{y_t}$ is the predicted value, y_t is the real value of the series, ε_j is the error term (white noise) at the jth prediction, p is the order of the process, η is the mean of the series and finally ϕ_i is the weight of the $(t-i)th$ error term.

Autoregressive (AR) models represent the series as a linear combination of past values plus an error term, i.e. in Equation 3.3:

$$\begin{aligned} y_t &= \sum_{i=1}^{q} \phi_i * x_{t-i} + \varepsilon \\ \hat{y_t} &= \sum_{i=1}^{o} \phi_i * x_{t-i} \end{aligned} \tag{3.3}$$

where x_j is the jth value of the series and q is the order of the autoregressive model.

Autoregressive Moving Average (ARMA) is a combination of the latter two models with two parameters (p, q) where p and q define the AR and MA part of the model respectively. The time series can be created according to this model as in Equation 3.4:

$$\begin{aligned} y_t &= \sum_{i=1}^{q} \phi_i * x_{t-i} + \sum_{i=1}^{p} \phi_i * \varepsilon_{t-i} + \varepsilon \\ \hat{y_t} &= \sum_{i=1}^{q} \phi_i * x_{t-i} + \sum_{i=1}^{p} \phi_i * \varepsilon_{t-i} \end{aligned} \tag{3.4}$$

where the $\hat{y_t}$ is the predicted value, y_t is the real value of the series and the other symbols are the same as in Equation 3.2 and Equation 3.3. These three models require the time series to be stationary - that is to have a constant mean and variance over time without any seasonality. Financial time series usually do not possess this property.

Autoregressive Integrated Moving Average (ARIMA) model tries to solve this issue by first differencing the time series to make its mean stationary (but it does not affect the variance). Differencing creates a new time series z from the original by taking the differences from it as shown in Equation 3.5:

$$z_i = y_{i+1} - y_i \tag{3.5}$$

This process is quite similar to taking a discrete derivative. The transformation is repeated until the mean is constant over time. The model parameter d describes the number or order of differencing that the model carries out. After this transformation ARIMA models the time series z as an ARMA model would. This is why an ARIMA model consists of three parameters (p, d, q) where p and q equal the contained ARMA(p, q) parameters.

Several works have studied the performance of ARIMA models on financial datasets. [7] and [55] experimented with New York Stock Exchange and Indian stocks respectively. Adebiyi et al. [5] found that the performance of simpler neural networks was comparable to ARIMA models and also concluded that there was no statistical difference between the two models. Worth noting that the test set was, unfortunately, limited[3] and the neural network baseline was an MLP which is reported to be outperformed by simple CNN or LSTM baselines on datasets like these.

Karakoyun and Cibikdiken [46] compared LSTMs and ARIMA models on Bitcoin price predictions and found that LSTMs heavily outperformed ARIMA models. Despite this, the general consensus of the scientific literature is that ARIMA is not only a powerful baseline but can be really competitive in random domains such as financial prices. There are a lot of results in the scientific community where neural networks' predictions were similar to a simple moving average [68][71][78][65]. Since neural networks are prone to learn such patterns, ARIMA can help notice this as well because the performance of the neural networks will be very similar to some kind of autoregressive moving average.

[3]which is a common problem in financial data related tasks

Chapter 4

Goal

As I detailed in Section 3.1, financial time series dataset evaluations are usually left out from complete, comparative TSC surveys. Furthermore, partly mentioned in Section 3.1.3, Transformers in general are less explored in the field of time series analysis than they are in other fields (e.g. NLP) and works in financial TSC domain are even rarer.

To explore a combination of the two, the goal of this work is to provide a well-documented, precisely evaluated baseline model to the scientific community regarding Transformer-based architectures and the Bitcoin TSC task.

To achieve this, I compare the investigated models against several well-known baselines-types to give a better picture of the results. This work may be viewed in the future as an outlining survey of the limits of basic Transformers in this regard and might mean a useful baseline and a well-defined starting point for future studies.

Chapter 5

Data

Table 5.1 details the data that was collected for preprocessing. I used financial indicators that describe the current trading statistics, and technical indicators that are about the state of the Bitcoin blockchain. Additionally, social media data was also used because of its potential impact, detailed in Section 2.3.2. The following table gives an overview of the features that were collected for the preprocessing stage and their short definitions.

Table 5.1: Used indicators

Cat.	**ID**	**Name**	**Description**
Misc.	1	Time	unix timestamp of the day
Financial	2	High	Highest daily exchange rate of BTC/USD
	3	Low	lowest daily exchange rate of BTC/USD
	4	Open	opening price of BTC/USD
	5	Close	closing price of BTC/USD
	6	VolumeFrom[1]	BTC traded to USD that day
	7	VolumeTo	USD traded to BTC that day
Technical	8	new_addresses	#new addresses registered on the Bitcoin blockchain
	9	active addresses	#active addresses on the Bitcoin blockchain
	10	transaction_count	#transactions carried out on the respective day
	11	large_transaction_count	#large transactions carried out the respective day
	12	block_height	length of blockchain
	13	hashrate	computer power allocated to the network[2]
	14	difficulty	average difficulty of mining a block
	15	block_time	average time to produce a new block[3]
	16	block_size	average block size
Social	17	reddit_subscribers	#subscribers to the Bitcoin subreddit [2]
	18	reddit_active_users	#active users on the Bitcoin subreddit
	19	reddit_posts_per_day	average #posts per hour on the BTC subreddit
	20	reddit_posts_per_hour	#posts that day on the BTC subreddit
	21	reddit_comments_per_hour	average #comments per hour on the BTC subreddit
	22	reddit_comments_per_day	#comments that day on the BTC subreddit

[1]Intentionally camelCased due to the data source that was used for evaluation. Details in Section 7.1.
[2]e.g. scales with the number of miners
[3]i.e. a hash is found

Chapter 6

Proposed Methods

6.1 Data Preparation

Data preprocessing and preparation is a crucial step of machine learning pipelines. Efficient data representation can have a huge boost in model performance because patterns are more easily learnable. Clean data is just as important because it helps the model generalize better and learn patterns that are time-invariant.

Additionally, although neural networks can learn what features are important and are able to combine features themselves feature engineering can save a lot of training time and improve accuracy. In this section, I detail the data preparation steps and techniques that were carried out before training.

6.1.1 Feature Engineering

6.1.1.1 Time features

Examining the closing price of BTC/USD, there is some difference in the level of the days of the week. I calculated the daily percentage changes in the prices. The mean and median of the respective daily changes both showed less volatility on weekends (Figure 6.1). Because of this, I added the day of the week as a feature in the input vectors. There are some patterns in monthly aggregations as well but considering the number of years in the data (12) these may not be statistically significant. Since neural networks are known to be rather robust against potentially not predictive features I added a month-of-the-year feature as well. Because the number of time steps is relatively small, measures were taken to reduce the dimensionality of the input vector.

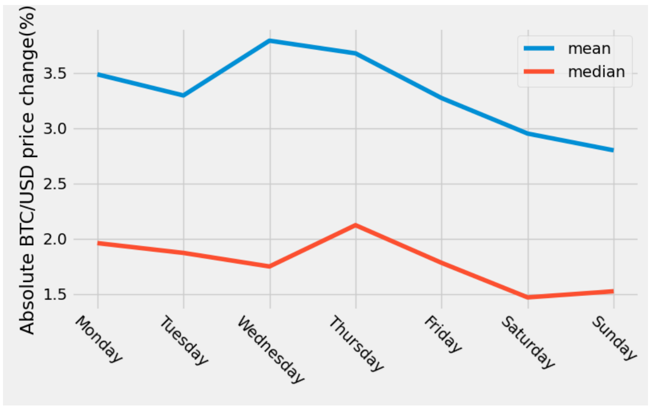

Figure 6.1: Absolute mean and median percentage change aggregated by day

6.1.1.2 Financial features

High, Low, Open, and Close features (ID 2-5 in Table 5.1) correlate incredibly strongly for obvious reasons. Taking the High and Low features a daily movement feature was created from the (signed) difference of the two. After this, the two source features were removed.

The Open feature (with ID 4) was removed completely because the previous day's Close feature is the same. This is due to the fact that unlike regular stocks on stock exchanges, crypto-currencies are traded all day.

6.1.1.3 Technical features

Feature with ID 15 - block_time - was dropped because it indicates the average time it takes to create blocks. Since the BTC blockchain is regulated to have approximately 10 minutes of block time this feature only indicates the deviations from this which is due to chance. If there is an anomaly in this feature that cannot be accredited to chance (e.g. large group of miners stopping mining suddenly and finding a good hash is more difficult), that anomaly would also show up in the hashrate.

The block_size (ID 16) correlates with the number of transactions it stores. The amount of transactions in a block depends on how fast a good hash was found so this is also a descriptive feature with no real predictive power for the future and was discarded. Small block sizes may mean fewer transactions (feature with ID 10) or the fortune of the miners to have found a good hash quickly.

Difficulty (ID 14) means how hard it is to mine a block (i.e. find a good hash). This directly correlates with hashrate so it was also removed from the dataset.

The current height of the blockchain does not carry any unique information about the momentum and state of the chain that is not encoded in transaction_count and hashrate. Due to this, ID 12 was removed from the dataset.

6.1.1.4 Social features

Valid social media data was only available from 2017-05-26. These features (IDs 17-22 from Table 5.1) had very high Person-correlations with each other which indicates strong linear connections between these features. Therefore, I applied Principal Component Analysis (PCA, F.R.S. [25]) and kept 93.60% of the variance of the original data with half of the original dimensions. After PCA, the three axes of the projected dataset contain 63.60%, 19.10%, and 10.90% of the original variance leaving only a little amount of information on the remaining 3 axes individually.

6.1.1.5 Target Variable

Due to the challenges, it is a common practice of the scientific literature to focus on classification regarding financial predictions [60][41][11][45][76]. Not only does classification provide an easier context for evaluation among used methods but it is a lot more meaningful information for the trader indicating whether to sell or buy the commodities.

According to a review of the field by Henrique et al. [32], 42% of the listed papers (57 total) use direction as a target which has a commanding relative majority over the other listed options (Return, Return and Risk, Volatility, Exact values, Maximum-Minimum, Volume, Mean of values). Additionally, because of the noise even on a day-to-day level, it may be more profitable to predict tendencies rather than the next value in the series.

This is why I decided to create a target for the next d days using linear regression and use that as a basis for the classification task.

Windows with w lengths were created from the multivariate series. Each window has a class label which is created as follows.

$$\mathrm{s} = \begin{cases} \frac{\boldsymbol{v}-\boldsymbol{v}[0]}{\mathrm{std}(\boldsymbol{v})}, & \text{if } \mathrm{std}(\boldsymbol{v}) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

where $\boldsymbol{v}$ is a vector containing the today's and the next d days' closing BTC/USD prices. $\boldsymbol{v}[0]$ is the first element (today's price) of the vector and the std() function is used to calculate the standard deviation of the vector.

The idea and intuition behind this equation are to be able to tell in what direction the exchange rate will continue regardless of scale, as seen in Figure 6.2. Generally, linear regression requires two parameters in the line equation $y = m * x + b$ where m is the slope of the line and b is the intercept. However, shifting the values to [v][0] ensures that only the slope of the line has to be predicted.

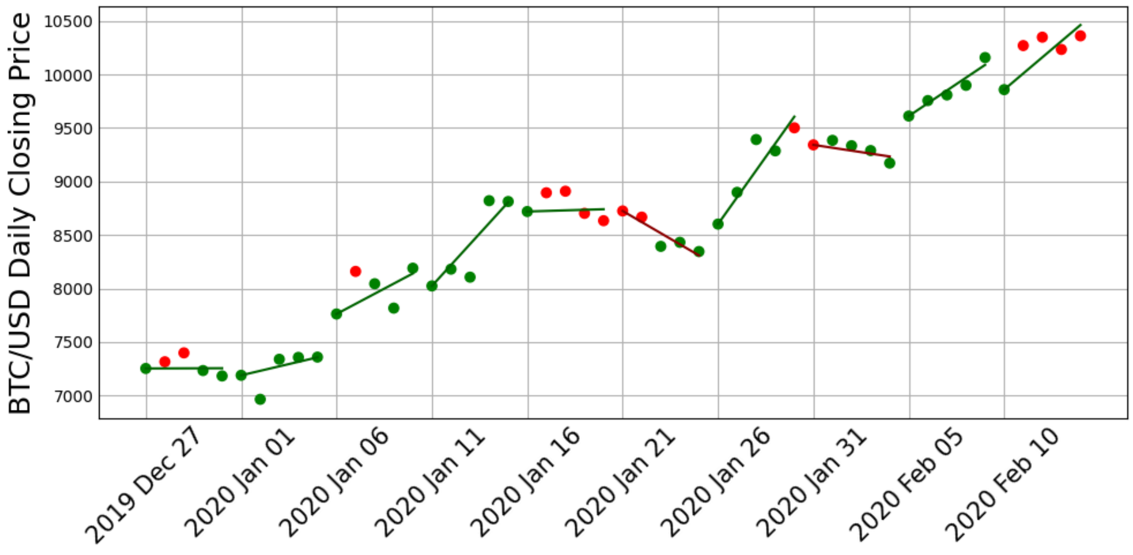

Figure 6.2: Visualized slopes. Green-Red colors indicate Decrease-Increase (Equation 6.2) classes respectively. (Lengths are proportionate to the magnitudes of the slope.)

These slope values then can be used to create classes. I created two targets ***c***lasses from these values according to Equation 6.2.

$$\boldsymbol{c} = \begin{cases} decrease & \text{if s} \leq 0 \\ increase & \text{if } 0 < \text{s} \end{cases} \tag{6.2}$$

Further dividing the target space with more discretizing borders (i.e. now 0.0) is also possible. Changing the borders or the number of classes might mean opportunities for this method to improve and they should be further investigated. For now, however, these lie outside the scope of this work.

As an input, the model is also provided the back-shifted regression values as input features. This means that the previous d day's slope value is also an input feature for predictions.

6.1.2 Transformations and cleaning

6.1.2.1 Smoothing

Series with low signal-to-noise ratio paired with high volatility usually have a lot of outliers. Scientific literature deals with this problem with smoothing and/or some kind of signal decomposition. I decided to apply smoothing for the target feature, Close. The type of smoothing is exponentially weighted moving mean with a relatively small *halflife* of h (this value was experimented with). This greatly helps reduce the huge jumps in the values.

6.1.2.2 Missing values

Almost 7 years of social media data were missing before 2017-05-26. The missing values in these years were imputed with zero.

The volumeFrom(ID 6) feature had zero values from 2011-06-20 Monday until 2011-06-25 Saturday. Since this occurred only during this six-day interval it was treated as missing data. Values were imputed based on the average of last and next week's values of the respective day. More precisely in Equation 6.4:

$$t_i = (t_{i+7} + t_{i-7})/2 \tag{6.3}$$

where t_i is ith value of the VolumeFrom feature between 2011-06-20 and 2011-06-25.

6.1.2.3 Logarithms

Values and indicators relating to Bitcoin have changed in scale, especially since its initial boom in 2018. Examples of this phenomenon may include the number of new addresses, transaction count, or traded volume. When variables span across multiple ranges of magnitude taking the respective logarithm is an effective practice to limit the range of values. This transformation also helps tail-heavy distributions to have a more normal distribution shape. This is beneficial because models with tail-heavy distributions do not perform as well as features that have bell-shaped density functions.

For the reasons above, I took the logarithms of the following features (followed by the respective IDs from Table 5.1):
new_addresses(8), active_addresses(9), transaction_count(10), VolumeFrom(6), VolumeTo(7), reddit_active_users(18), reddit_comments_per_day(22). Since many features contained zero values (but not negatives), the values were shifted by one resulting in the transformation:

$$\boldsymbol{f} = log(\boldsymbol{f} + 1) \tag{6.4}$$

where $\boldsymbol{f}$ is the vector containing the values of the feature. Addition and logarithm taking is applied by element.

6.2 Training Data Generation

For time series forecasting, a hyperparameter can describe the number of previous time steps that are considered for predictions. This is often referred to as window_size. Overlapping windows of size w were created while shifting the window by (i.e. window stride = 1). Non-overlapping windows would have resulted in a drastic reduction of the training (and test) set so setting the stride of the window selection to as low as possible was a priority. With this, each input for the model is a two-dimensional matrix of shape $w\text{X}f$ which describes the previous w time steps with f features. Given a sample size of s, a window size of w, and forecasting d days ahead (Section 6.1.1.5), the number of final training samples N equals $s - (w - 1) - d$. This is because w samples are discarded because we cannot generate windows from the first $w - 1$ samples and beyond the last d samples we do not have data points to calculate the target. In my experiments, I used $w = 10$ and $d = 5$ during all of the trainings.

6.3 Filtering Targets

The goal is to enable to model to learn temporal and spatial patterns in the input space to predict the regression. Since there are a lot of external impacts it is not logical to expect the model to predict for example a big headline that greatly affects prices. For this reason, targets are filtered so the model is less expected to predict big jumps in the data which are unforeseeable.

Since the price is incredibly volatile (sometimes tens of percents with seemingly no reason) alone it is not the best indicator of external events. I investigated different methods to filter for such events. When news hit the internet and Bitcoin prices are greatly affected Reddit activity is expected to grow. I found that the number of reddit_comments per day is also a solid indicator of such events. For external events, the closing price can show a big dip or a big leap in prices but Reddit activity can only increase. I created an indicator value for each day according to Equation 6.5.

$$\text{ind\_value} = \max(0,\text{pct\_change}(\text{reddit\_comments})) * \text{abs}(\text{pct\_change}(\text{Close})) \tag{6.5}$$

where pct_change is the percentage change of the feature from yesterday, reddit_comments if the reddit_comments_per_day feature from Table 5.1. This linear combination[1] of the two proved really effective in detecting reactions for external events.

Days in the top p percent based on this indicator were invalidated, not expecting the model to be able to predict these changes. Since the target variable is based on linear regression for the next d days, regressions in the previous $d-1$ days would also be affected by these anomalies. In Figure 6.3 visualization of this filtering can be seen. The black points are marked at unpredictable jumps. Red groups of $d - 1$ points were marked as affected before each black point. Inputs whose target regression would start at a red point are taken out of the training set because anomalies (i.e. black points) affect their slope and therefore their labels. This way, the model does not try to learn patterns that were marked unpredictable by this method.

[1]weights could be fine-tuned in the future

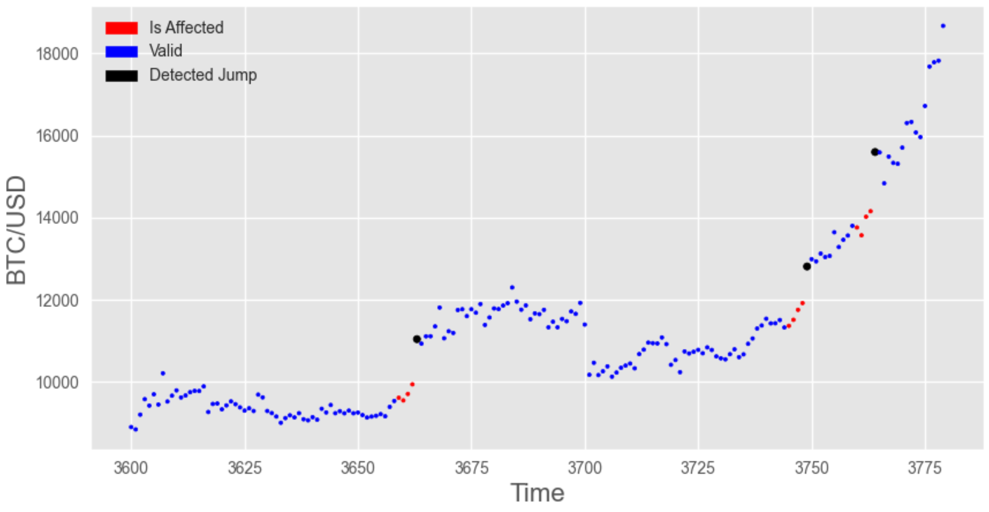

Figure 6.3: Closing price of BTC/USD. Windows whose target would start at a red point are discarded

6.4 Training Methods and Regularization

The training data is very limited and overfitting is very common in scenarios like this. Regularization methods help to mitigate the chance/degree of overfitting.

I found that batch normalization [42] and layer normalization [9] boosted performance in several models. Both are suspected to help mitigate the effects of outliers in noisy data. Penalties such as l1 and l2 regularization techniques were also used. l1 and l2 (or Lasso and Ridge Regression respectively) are added terms to the loss functions and they tend to have an impact on how the weights are tuned during training[2].

$$J(\boldsymbol{\theta})_{ridge} = L + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2 \tag{6.6}$$

$$J(\boldsymbol{\theta})_{lasso} = L + \alpha \sum_{i=1}^{n} |\theta_i| \tag{6.7}$$

Equation 6.6 and Equation 6.7 illustrate the details of Ridge and Lasso Regressions respectively. J($\boldsymbol{\theta}$) indicates the resulting cost function with θ symbolizing the weights and L being the initial loss during training. Ridge regression punishes large weights while Lasso regression usually eliminates the least important features' weights.

Dropouts [33] were also installed in all models to help the model to generalize better.

To combat class imbalance, mentioned in Section 2.3.3, the time series was undersampled. Although this reduces the training (and test) data, this helped stabilize training. Since positive labels had a majority over negative labels (about 1500 vs. 2000 in the training

[2]But it does not affect the bias terms directly

set), I decided to discard m positive samples from the training set so that the classes had the same amount of labels. The chosen samples were not random, the first m positive samples were dropped because if patterns change through time then these samples are the least relevant.

The used optimizer for all trainings was the Adam [47] optimizer.

6.5 Baseline Models

6.5.1 Naive Predictor

This baseline calculates the regression and the label of the past w days and uses that as a prediction for the next d days.

6.5.2 ARIMA

Since ARIMA cannot be trained on any targets, the forecasting horizon of the model was set to d ($d = 5$ in the experiments). From the predicted values of Close, the slope and its class (Section 6.1.1.5) were calculated and the evaluation was based on it. The model was an ARIMA(3,1,3)[3] model and was trained on the whole preprocessed training split. The forecasting algorithm was a rolling method - after predicting the values for the next d days the model was always refitted to move it forward through time for the next prediction.

6.5.3 Neural Networks

The following tables in this subsection summarize the used neural network baselines. A less detailed, visualized version is also available in Appendix A.
Used expressions and names in the tables are in sync with the Tensorflow API[4]. Any parameter which is not included is set to the default value in Tensorflow except for the initialization of the weights with LeakyReLU activation function. He or Kaiming [30] initialization was used in this case.

[3]The used implementation was from the sktime library [3]
[4]Version: 2.10.0

Table 6.1: MLP baseline detailed architecture. Visualization in Figure A.1

Layer	Parameters	Activation	Regularizer
Dense	units=6	LeakyReLU(0.3)	l1 and l2
Flatten	-	-	-
Dense	units=30	LeakyReLU(0.3)	l1 and l2
BatchNormalization	-	-	-
Dense	units=30	LeakyReLU(0.3)	l1 and l2
BatchNormalization	-	-	-
Dropout	rate=0.2	-	-
Dense	units=3	Softmax	-

Table 6.2: LSTM baseline detailed architecture. Visualization in Figure A.1

Layer	Parameters	Activation	Regularizer
LSTM	units=20, return_sequences=True	Tanh	-
Layer Normalization	-	-	-
LSTM	units=20, return_sequences=True	Tanh	-
LayerNormalization	-	-	-
LSTM	units=20, return_sequences=False	Tanh	-
Flatten	-	-	-
Dense	units=20	Relu	l1 and l2
Dense	units=10	Relu	l1 and l2
BatchNormalization	-	-	-
Dropout	rate=0.3	-	-
Dense	units=3	Softmax	-

6.6 Transformer-based models

6.6.1 Embeddings and Positional Encoding

The original paper targets NLP problems with this architecture. The inputs on that field are usually token vectors where each element is a word ID from a given vocabulary. Since these are discrete values, the continuous-valued time series described in this work cannot use the same implementation. Additionally, the intrinsic dimension of the data is suspected to be a lot less than the fifteen features of the used multivariate time series because of heavy correlations.
To boost training efficiency, I implemented an autoencoder[6] with a single hidden layer to create a pretrained embedding. After experiments, results seen in Figure 6.4, the latent dimension was chosen to be eight.

To encode the positions of the respective vectors I used the same technique as in [75]. This mechanism adds different constant vectors to the latent representations of the inputs

[5]When using 2D convolutions with multivariate time series a larger kernel_size may be preferable to enable pattern recognition across wider ranges of spatial and temporal dimensions. This modification however would probably imply adding more layers with smaller kernel_sizes. I wanted to keep the baseline architectures relatively simple and less complex. Different kinds of pooling layers are not included either for the same reason.

[6]A model which is taught to represent its input in lower dimensions

Table 6.3: CNN baseline detailed architecture[5]. Visualization in Figure A.2

Layer	Parameters	Activation	Regularizer
Conv2D	filters=5,kernel_size=(6,6)	LeakyReLU(0.2)	l1 and l2
BatchNormalization	-	-	-
Conv2D	filters=3,kernel_size=(1,1)	LeakyReLU(0.2)	l1 and l2
BatchNormalization	-	-	-
Conv1D	filters=5, kernel_size=6	LeakyReLU(0.2)	l1 and l2
Conv1D	filters=3,kernel_size=3	LeakyReLU(0.2)	l1 and l2
Flatten	-	-	-
Dropout	rate=0.3	-	-
Dense	units=20	LeakyReLU(0.2)	l1 and l2
BatchNormalization	-	-	-
Dense	units=10	LeakyReLU(0.2)	l1 and l2
BatchNormalization	-	-	-
Dense	units=3	Softmax	l1 and l2

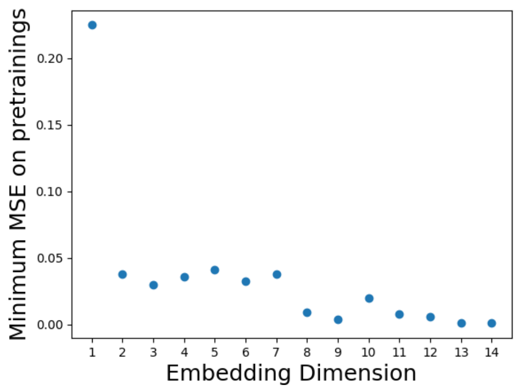

Figure 6.4: Latent dimensions of the embedding layer vs. recoverability of the input vectors

that the model can learn. These values are detailed in Equation 6.8 which is directly from Vaswani et al. [75].

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{6.8}$$

where pos is the position of the vector in the input, d_{model} is the dimension of the latent vectors and i is the ith element of the respective vector. To summarize, this technique gives values of different frequencies of sinusoidal functions of each vector element.

6.6.2 Tested Models

The tested two models are based on encoder blocks from Figure 3.1. The first model is built on a single **Encoder Block**. The detailed architecture can be seen in Table 6.4.

I also tested a slightly bigger model, called **Encoder Stack**. Although this is a bigger model, this may give more insights into how efficient and competitive a single Transformer encoder block is in this task. Architecture is detailed in Table 6.5.

Table 6.4: Transformer Encoder Block Model. Visualization in Figure A.3

Layer	**Parameters**	**Activation**	**Regularizer**
Layer Normalization	-	-	-
embedding	units=8	LeakyReLU(0.3)	l1_l2
Positional Encoding	-	-	-
Encoder Block	intermediate_dim=20, num_heads=3,dropout=0.1	LeakyReLU(0.3)	l1_l2
Flatten	-	-	-
Dense	units=3	SoftMax	-

Table 6.5: Transformer Encoder Stack Model. Visualization in Figure A.3

Layer	**Parameters**	**Activation**	**Regularizer**
Layer Normalization	-	-	-
embedding	units=8	LeakyReLU(0.3)	l1_l2
Positional Encoding	-	-	-
Encoder Block	intermediate_dim=20, num_heads=2,dropout=1	LeakyReLU(0.3)	l1_l2
Encoder Block	intermediate_dim=20, num_heads=2,dropout=0.1	LeakyReLU(0.3)	l1_l2
Encoder Block	intermediate_dim=20, num_heads=2,dropout=0.1	LeakyReLU(0.3)	l1_l2
Flatten	-	-	-
Dense	units=10	-	-
Dense	units=3	SoftMax	-

Chapter 7

Evaluation And Results

This chapter details metrics about the dataset that the proposed methods were evaluated on. After into the dataset, later sections show what models were experimented with and how the models were set up for evaluation. Finally, Section 7.3 contains the results of the trainings and statistics about the effects of proposed methods.

7.1 Data source used for evaluation

The proposed methods were applied to the data that can be collected using the CryptoCompare API [1]. The website provides data on technical indicators of the Bitcoin blockchain, some social media metrics related to Bitcoin, and statistics on daily trading. These features describe the state of the blockchain from many aspects. Table 7.1 shows some statistics about the data collected from the API.

Table 7.1: Statistics on some features

	Close	**VolumeFrom**	**transaction count**	**active users**[1]	**comments / day**[2]
Min	0.04951	0.0	260	0	0
Max	67549.14	572349.32	490644	39469	34838.71
Earliest Value	2010-07-17	2010-07-17	2010-07-17	2017-01-30	2017-01-30
std(% Change)	0.0776	1.3861	0.2947	0.3207	0.5255

The provided time series is daily sampled, meaning features describe daily data. The first point in the series is 2010-07-17 and the last one to 2022-07-22. The data before 2021 is used for training and the data in 2021-2022 is reserved for testing. 20% of the training data is used for validation. Detailed information about the splits can be found in Table 7.2.

[1]active_reddit_users in Table 5.1
[2]reddit_comments_per_day in Table 5.1

Table 7.2: Splitting the data points to train/test splits

	Number of Data Points	Percent
Train	3820	87
Test	567	13
Total	4388	100

7.2 Configurations

Since (test) data is limited and trainings were sometimes unstable, an accidental good result for the test set could deceive the evaluation of the models. This is why I took into account multiple runs to evaluate the models. I ran every configuration for twenty-five trainings and used the results to make evaluations. Configurations differ in smoothing, detailed in Section 6.1.2.1, target filtering (Section 6.3) values alongside model types. In this chapter, the notation for the amount of exponential smoothing is h and p for the used percentage from target filtering. All combinations of these parameters would mean an enormous search space to explore with all models through multiple runs so I experimented with fixed, sampled values from the ranges. For this reason, the whole search space - for neural networks - is $H \times P \times M$ where $H = \{0, 1, 3\}$, $P = \{0, 1, 3, 5, 8, 13\}$ and $M = \{\text{LSTM, CNN, MLP, Encoder block, Stacked Encoder}\}$ denoting the possible half-life values, filtering percentage values and model types respectively.

Smoothing leads to numerically better performance on the test set because trends and changes will get smoothed resulting in easier predictions. Applying target filtering with the value of p is not feasible at inference time because that would require us to look into the future. For this reason, all configurations were evaluated on an unfiltered, unsmoothed test set.

7.3 Results

I frequently use medians and MoM or Median of Medians since it is robust against outliers in terms of training results and it gives a great indication of the average performance of a given setup.

Figure 7.1 details the aggregated performance of the tested configurations. The median accuracy for a given configuration was calculated across all runs. Then these medians were aggregated for a given cell across several configurations (e.g. all configurations where $h = 0$ and $p = 3$). This gives a median of medians accuracy for a given place in the grid.

Median accuracies are also visualized per model in Figure 7.2. This figure gives an overview of performances and the number of trainable model parameters. More precisely, the box-plot shows the distribution of median accuracies (of twenty-five runs per configuration) for configurations with the given model. To supplement this, Table 7.3 summarizes statistical prediction strategies for the test set.

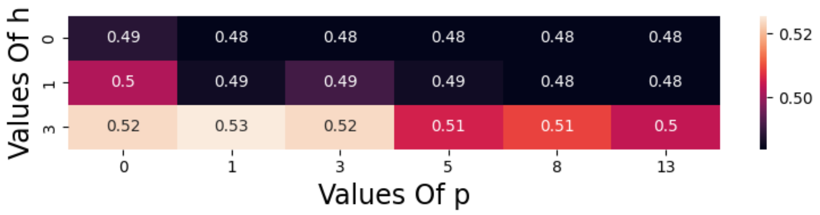

Figure 7.1: MoM accuracy results aggregated by p and h values.

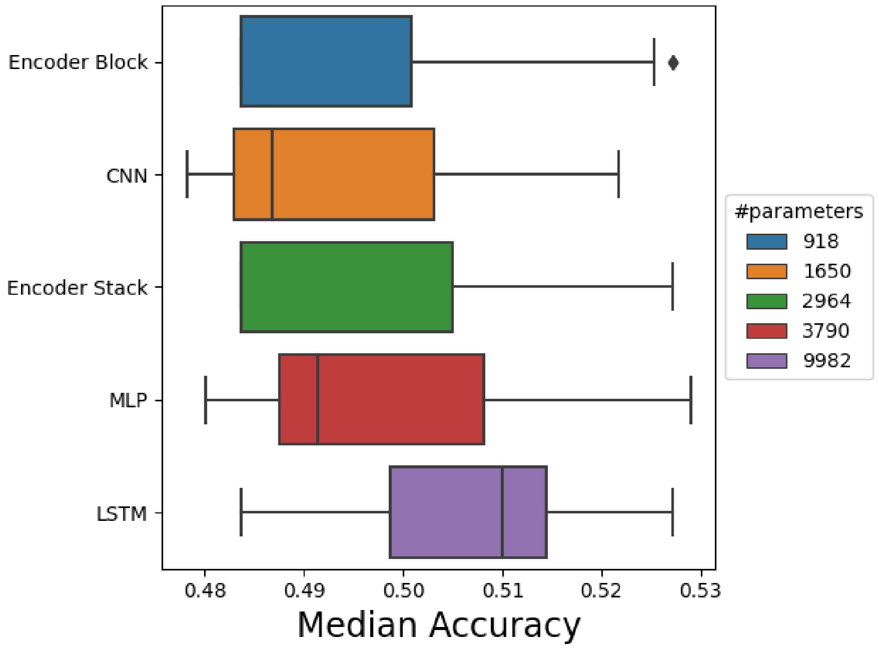

Figure 7.2: Distribution of median configurations accuracies over model types. Colors and order are determined by the number of trainable parameters.

Table 7.3: Statistical Strategies

Model	Accuracy
All Negative Predictions	**0.5276**
All Positive Predictions	0.4724
Naive Prediction[3]	0.5191
ARIMA	0.5018

Table 7.4: MoM AUC of predictors

Model (#parameters)	MoM AUC
CNN (1650)	0.4913
Encoder Block (918)	0.4772
Encoder Stack (2964)	0.5120
LSTM (9982)	**0.5207**
MLP (3790)	0.5143

Plotting False Positive Rate and True Positive Rate gives the Receiver Operating Characteristic (ROC) curve. The area under this curve (it is between 0 and 1) measures the perfectness of a classifier - Area Under the ROC Curve (AUC) equals one for a perfect classifier and a half for random predictions. Table 7.4 details the MoM Area Under the ROC Curve. It shows the median AUC of all runs for a configuration aggregated by a median for a given model.

[3]Introduced in Section 6.5.1

Figure 7.3 shows the activation scores of the self-attention mechanism on the test set across different heads. These figures are from the best-performing Encoder Block model with $p = h = 0$ configuration. The mean attention scores were taken across the test set for each self-attention head (three) accounting for the three pictures. The axes have ten ticks because of the used w window size of ten which is why this figure shows the self-attention scores between different time steps. Because of the window generation algorithm, the ninth (indexed) time step in the window is the latest in absolute time.

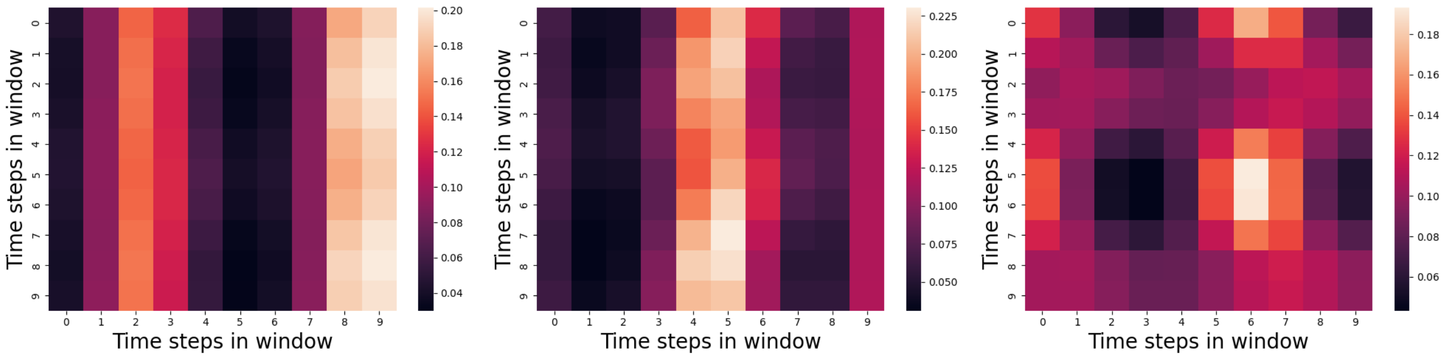

Figure 7.3: Visualizing average activations of three self-attention heads in the Encoder Block model on time series data.

Chapter 8

Conclusions

Figure 7.2 show that the transformer-based models have the lowest accuracy and are outperformed in this metric by every neural network and statistical baseline except for all positive predictions. Additionally, this figure suggests that stacking encoders do not give additional power in this setup for the classifiers. Conducting the two-sample Kolmogorov-Smirnov [35] test to compare the prediction accuracies (of twenty-five runs) gives p-values greater than 0.05 for all configurations. From this, we can assume with high confidence that the classifiers' powers (on this task with this dataset) are the same. Medians of the Encoder Stack and Encoder Block predictions do not show on the boxplots. The reason for this is that the lowest accuracies are the medians as well for these classifiers. In the majority of the trainings they learned to predict all positive classes which indicates they failed to grasp any useful patterns in the training set.

Furthermore, Figure 7.3 show that in the left attention head the last time steps get very high attention scores. This may mean the network has learned some kind of autoregressive moving average which might explain the poor performance of both ARIMA and the encoders.

Figure 7.2 and Table 7.4 both show that LSTMs proved to be the best predictors both in terms of accuracy and AUC. LSTMs had decisively better results in both metrics. Worth noting that LSTMs had the most number of parameters and the trend from Figure 7.2 suggests that some models may have had too few parameters because the higher the parameter count the better the models tend to perform. Figure 7.2 also shows that all models had high accuracies and evaluating them on multiple runs gave more accurate insights.

Figure 7.1 indicate that smoothing Close prices boost performance on the unsmoothed test set. It also shows that target filtering had little to no effect at small values but a negative performance impact at larger values. This performance drop may be due to the heavy sample losses because high p values discard more and more samples from the dataset.

Chapter 9

Summary

In this work, I experimented with Transformer-based classification for Bitcoin time series data. The tested models failed to outperform popular baselines and it was also observed that stacking multiple encoder blocks does not yield additional prediction power. Moderately filtering targets (with small values of p) with Reddit trends and price jumps to avoid learning random patterns did not decisively worsen or boost performance. However, heavy target filtering showed detrimental effects on performance, presumably due to too much sample loss. An exponential smoothing of the closing prices did improve predictions visibly for BTC price TSC.

There are several points where this method could be improved in the future. Preprocessing techniques and target choices have a huge impact on model performances. Techniques in this field have a great variety and combinations could provide hidden opportunities for improvement.

Transfer learning may provide a solution for data deficiency. There are several other cryptocurrencies and bigger blockchains such as Ethereum or Cardano getting more and more attention. Since the mechanisms and behaviors are really similar it may be possible to learn patterns or embeddings on one blockchain that would help develop prediction models for another. On a similar note, extending the features with data from other blockchains may be beneficial too. Bitcoin trends are believed to have an impact on other smaller blockchains' prices and predicting those could be helped by Bitcoin data.

In this work, I concentrated on classification but a Transformer could be trained in the future to predict the exact values of the series. Comparing that method to, amongst others, ARIMA's rolling forecasting could be another valuable insight into Transformers' performance.

Acknowledgements

I would like to thank my advisor, Dr. Bálint Gyires-Tóth for continuous feedbacks and ideas for this project. He helped me create my first work of this magnitude and constantly pointed out areas where improvements could be made.

Bibliography

[1] Cryptocurrency API, Historical Real-Time Market Data | CryptoCompare — min-api.cryptocompare.com. https://min-api.cryptocompare.com/documentation. [Accessed 15-Oct-2022].

[2] r/Bitcoin — reddit.com. https://www.reddit.com/r/Bitcoin/. [Accessed 15-Oct-2022].

[3] Welcome to sktime x2014; sktime documentation — sktime.org. https://www.sktime.org/en/stable/. [Accessed 24-Oct-2022].

[4] RE Abdel-Aal and Ahmed Z Al-Garni. Forecasting monthly electric energy consumption in eastern saudi arabia using univariate time-series analysis. *Energy*, 22(11):1059–1069, 1997.

[5] Ayodele Ariyo Adebiyi, Aderemi Oluyinka Adewumi, and Charles Korede Ayo. Comparison of arima and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, 2014, 2014.

[6] Saúl Alonso-Monsalve, Andrés L Suárez-Cetrulo, Alejandro Cervantes, and David Quintana. Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Systems with Applications*, 149:113250, 2020.

[7] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*, pages 106–112. IEEE, 2014.

[8] Parul Arora, Himanshu Kumar, and Bijaya Ketan Panigrahi. Prediction and analysis of covid-19 positive cases using deep learning models: A descriptive case study of india. *Chaos, Solitons & Fractals*, 139:110017, 2020.

[9] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[11] Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. Evaluating multiple classifiers for stock price direction prediction. *Expert systems with Applications*, 42(20):7046–7056, 2015.

[12] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, page 26–33, USA, 2001. Association for Computational Linguistics. DOI: 10.3115/1073012.1073017. URL https://doi.org/10.3115/1073012.1073017.

[13] Chris Burniske and Adam White. Bitcoin: Ringing the bell for a new asset class. *Ark Invest (January 2017) https://research. ark-invest. com/hubfs/1_Download_Files_ARK-Invest/White_Papers/Bitcoin-Ringing-The-Bell-For-A-New-Asset-Class. pdf*, 2017.

[14] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE international conference on big data (big data)*, pages 2823–2824. IEEE, 2015.

[15] Sheng Chen and Hongxiang He. Stock prediction using convolutional neural network. In *IOP Conference series: materials science and engineering*, volume 435, page 012026. IOP Publishing, 2018.

[16] Wei Chen, Huilin Xu, Lifen Jia, and Ying Gao. Machine learning model for bitcoin exchange rate prediction using economic and technology determinants. *International Journal of Forecasting*, 37(1): 28–43, 2021.

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] D. Dickey and Wayne Fuller. Distribution of the estimators for autoregressive time series with a unit root. *JASA. Journal of the American Statistical Association*, 74, 06 1979. DOI: `10.2307/2286348`.

[20] Johann Du Preez and Stephen F Witt. Univariate versus multivariate time series forecasting: an application to international tourism demand. *International Journal of Forecasting*, 19(3):435–451, 2003.

[21] Anne Haubo Dyhrberg. Bitcoin, gold and the dollar – a garch volatility analysis. *Finance Research Letters*, 16:85–92, 2016. ISSN 1544-6123. DOI: `https://doi.org/10.1016/j.frl.2015.10.008`. URL `https://www.sciencedirect.com/science/article/pii/S1544612315001038`.

[22] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

[23] Saeede Anbaee Farimani, Majid Vafaei Jahan, Amin Milani Fard, and Seyed Reza Kamel Tabbakh. Investigating the informativeness of technical indicators and news sentiment in financial market price prediction. *Knowledge-Based Systems*, 247:108742, 2022.

[24] Matthew Fox. Dogecoin pops 8% after Elon Musk says he'll eat a happy meal on TV if McDonald's accepts the meme coin as payment — markets.businessinsider.com. `https://markets.businessinsider.com/news/currencies/dogecoin-price-elon-musk-tweet-mcdonalds-should-accept-meme-coin-2022-1`, 2022. [Accessed 15-Oct-2022].

[25] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 1*, 2:559–572.

[26] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[27] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[28] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[29] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2):8–12, 2009.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. URL `https://arxiv.org/abs/1502.01852`.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, and Herbert Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, 124:226–251, 2019.

[33] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[35] John L Hodges. The significance probability of the smirnov two-sample test. *Arkiv för Matematik*, 3 (5):469–486, 1958.

[36] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[37] https://www.facebook.com/bbcnews. Coronavirus: How the pandemic has changed the world economy — bbc.com. `https://www.bbc.com/news/business-51706225`, . [Accessed 23-Oct-2022].

[38] https://www.facebook.com/bbcnews. Bitcoin becomes official currency in Central African Republic — bbc.com. https://www.bbc.com/news/world-africa-61248809, . [Accessed 23-Oct-2022].

[39] https://www.facebook.com/bbcnews. China declares all crypto-currency transactions illegal — bbc.com. https://www.bbc.com/news/technology-58678907, . [Accessed 23-Oct-2022].

[40] https://www.theguardian.com/profile/rob davies. GameStop: how Reddit amateurs took aim at Wall Street's short-sellers — theguardian.com. https://www.theguardian.com/business/2021/jan/28/gamestop-how-reddits-amateurs-tripped-wall-streets-short-sellers, 2021. [Accessed 15-Oct-2022].

[41] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & operations research*, 32(10):2513–2522, 2005.

[42] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[43] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[44] Suhwan Ji, Jongmin Kim, and Hyeonseung Im. A comparative study of bitcoin price prediction using deep learning. *Mathematics*, 7(10):898, 2019.

[45] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319, 2011.

[46] Ebru Seyma Karakoyun and AO Cibikdiken. Comparison of arima time series model and lstm deep learning algorithm for bitcoin price forecasting. In *The 13th multidisciplinary academic conference in Prague*, volume 2018, pages 171–180, 2018.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[49] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[50] Hao Li, Yanyan Shen, and Yanmin Zhu. Stock price prediction using attention-based multi-input lstm. In Jun Zhu and Ichiro Takeuchi, editors, *Proceedings of The 10th Asian Conference on Machine Learning*, volume 95 of *Proceedings of Machine Learning Research*, pages 454–469. PMLR, 14–16 Nov 2018. URL https://proceedings.mlr.press/v95/li18c.html.

[51] Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*, 2021.

[52] Ioannis E Livieris, Niki Kiriakidou, Stavros Stavroyiannis, and Panagiotis Pintelas. An advanced cnn-lstm model for cryptocurrency forecasting. *Electronics*, 10(3):287, 2021.

[53] Eduardo José da S Luz, William Robson Schwartz, Guillermo Cámara-Chávez, and David Menotti. Ecg-based heartbeat classification for arrhythmia detection: A survey. *Computer methods and programs in biomedicine*, 127:144–164, 2016.

[54] Sean McNally, Jason Roche, and Simon Caton. Predicting the price of bitcoin using machine learning. In *2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP)*, pages 339–343. IEEE, 2018.

[55] Prapanna Mondal, Labani Shit, and Saptarsi Goswami. Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13, 2014.

[56] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[57] David MQ Nelson, Adriano CM Pereira, and Renato A De Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. Ieee, 2017.

[58] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[59] Marco Ortu, Nicola Uras, Claudio Conversano, Silvia Bartolucci, and Giuseppe Destefanis. On technical trading and social media indicators for cryptocurrency price classification through deep learning. *Expert Systems with Applications*, 198:116804, 2022.

[60] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert systems with applications*, 42(1):259–268, 2015.

[61] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[62] Xiangyun Qing and Yugang Niu. Hourly day-ahead solar irradiance prediction using weather forecasts by lstm. *Energy*, 148:461–468, 2018.

[63] Jiayu Qiu, Bin Wang, and Changjun Zhou. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, 15(1):e0227222, 2020.

[64] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[65] Murtaza Roondiwala, Harshal Patel, and Shraddha Varma. Predicting stock prices using lstm. *International Journal of Science and Research (IJSR)*, 6(4):1754–1756, 2017.

[66] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[67] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[68] Muhammad Saad, Jinchun Choi, DaeHun Nyang, Joongheon Kim, and Aziz Mohaisen. Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions. *IEEE Systems Journal*, 14(1):321–332, 2019.

[69] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[70] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[71] Giulia Serafini, Ping Yi, Qingquan Zhang, Marco Brambilla, Jiayue Wang, Yiwei Hu, and Beibei Li. Sentiment-driven price prediction of the bitcoin based on statistical and deep learning approaches. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[72] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[73] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[74] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th conference on business informatics (CBI)*, volume 1, pages 7–12. IEEE, 2017.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information*

Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[76] Bin Weng, Mohamed A Ahmed, and Fadel M Megahed. Stock market one-day ahead movement prediction using disparate data sources. *Expert Systems with Applications*, 79:153–163, 2017.

[77] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[78] Chih-Hung Wu, Chih-Chiang Lu, Yu-Feng Ma, and Ruei-Shan Lu. A new forecasting framework for bitcoin price with lstm. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 168–175. IEEE, 2018.

[79] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[80] Jing Zhang, Shicheng Cui, Yan Xu, Qianmu Li, and Tao Li. A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60–69, 2018.

[81] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.

Appendix A
Model Visualizations

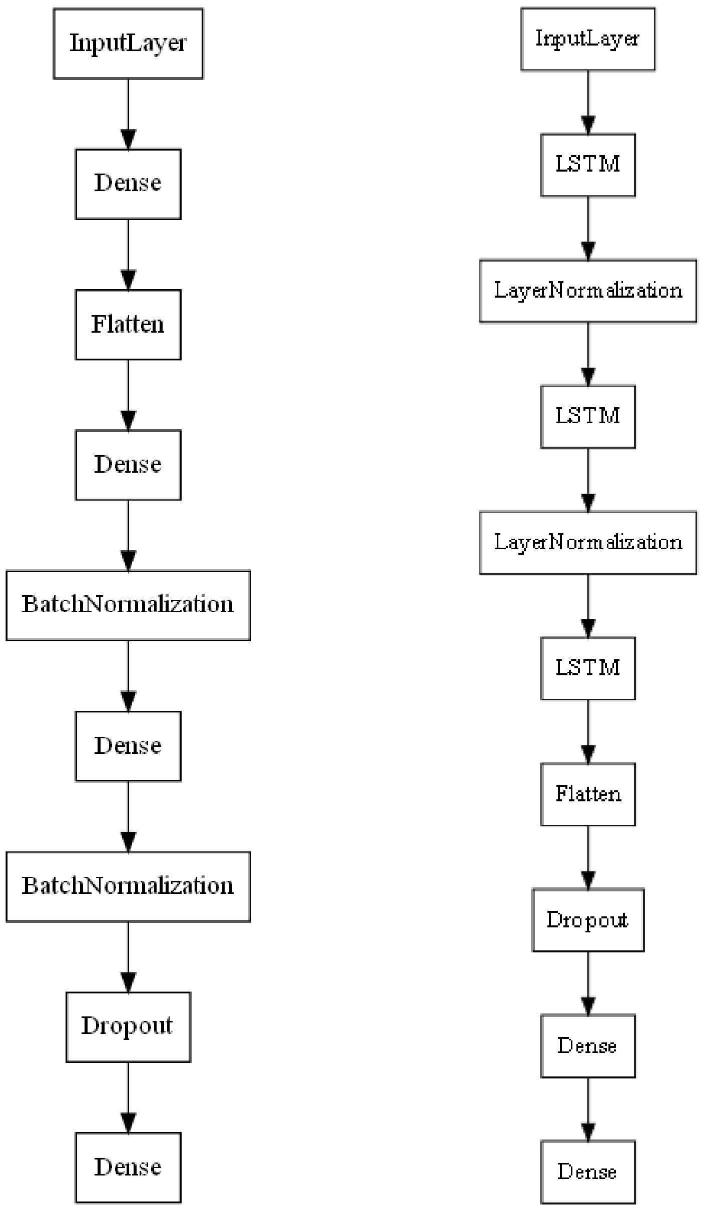

Figure A.1: MLP(left) and LSTM(right) baseline visualization. Details in Section 6.5.3

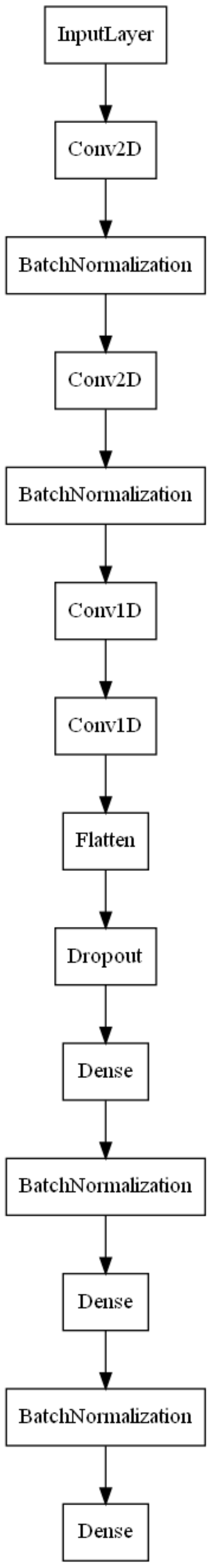

Figure A.2: CNN baseline visualization. Details in Section 6.5.3

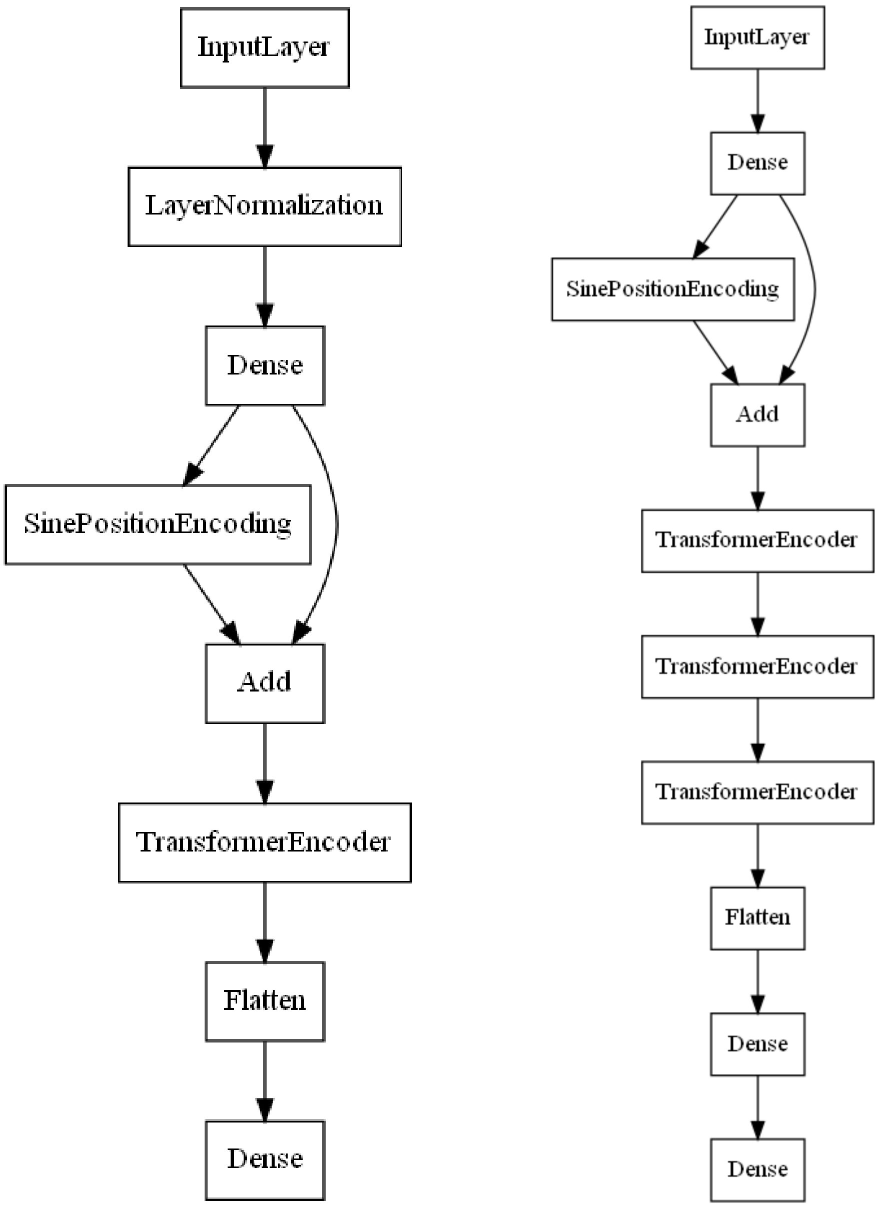

Figure A.3: Encoder Block(left) and Encoder Stack(right) baseline visualization. Details in Section 6.6.2