Late Fusion Multivariate Stock Market Prediction

Markus Armbruster
CS231N
armbrusm — X638434 SUID# — 06735111
markusarmbruster@gmx.de

Abstract

Stock market prediction has been a field of research for decades. Advances in AI have helped to build better estimates for future price movement predictions. This paper investigates a few deep learning classification methods to predict rewarding BUY signals which can then be used in applied stock market trading strategies.

The methods used in this paper are grounded on convolutional neural networks and are applied on two distinct representations of the same stock market data. Two basic CNN models serve as baseline to further evaluate against a more advanced Fusion CNN model. The results of the Fusion model are promising and outperforming the basic network architectures as well as the paper that was used as performance reference.

1. Introduction

Stock markets and its prediction of future price movement has been a subject of matter in much research. The Efficient Market Hypothesis claims that stock prices cannot be predicted nor modelled by today's algorithms. But advances in AI can help in approximating the market movements better and better.

In general, stock market analysis is divided into two different core approaches.

A) *market fundamentals analysis* which considers macro market data and other broader long-term indicators to predict market movements
B) *technical analysis* which solely focuses on price and volume movements of a specific market instrument during a short-term window to predict market movements.

In this project I focused on technical analysis but in contrast to many common approaches by using recurrent neural networks (RNNs) or time-series analysis, I applied computer vision to predict BUY signals.

The goal of this project is to prove or disprove the learnability and therefore the possibility for predicting market movements with deep learning models that are specifically grounded on computer vision approaches like convolutional neural networks (CNNs). To achieve this, I evaluated plain vanilla CNNs, used transfer learning on existing models like Resnet and built a more advanced fusion model.

The dataset for the models needed to be generated from scratch as there was no valid dataset publicly available. Preprocessing was very intensive to build a valid dataset that translates the raw data into the needed labeled representations. Therefore, I collected a vast amount of historical raw data for the market instrument *BTC/USD Perpetual Futures* which is used in this paper. It has been extracted on a one-minute timeframe, labeled with BUY signals and preprocessed to candle stick chart images and market indicator images.

To train and evaluate the results, I split the data as usual into train, validate and test sets and compared the accuracy and precision of each network architecture. The fusion model turned out to perform better than the baseline architectures and achieved an acceptable precision score of 0.42.

2. Related Work

Although there are many publications on deep learning image classification, there is only a limited number of papers which focus on applying computer vision on stock market data. Most papers and literature apply time series analysis like statistical analysis, signal processing, pattern recognition, machine learning and other deep learning techniques [1].

Statistical and mathematical analysis in time series data can be achieved through determining mathematical objectives like min/max, moving average, variance, standard deviation, autocorrelation, cross-correlation etc. These methods are core in determining so called *market indicators* which serve as key inputs for this project as well.

In addition, literature often refers to regression analysis, autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), Bayesian analysis and Kalman filters. [2]

Machine learning algorithms that are used are clustering algorithms (e.g. KNNs), hidden Markov models, support vector machines (SVMs) and deep neural networks (DNNs) / Multi-layer Perceptrons (MLPs). Cavalcante et al. [3] for

example surveyed forecasting models based on DNNs and ensemble methods. Guresen et al. [4] evaluated neural network models and in particular multi-layer perceptron models to predict NASDAQ stock index. Ballings et al. [5] compared the performances of different models (e.g. random forest, adaboost and kernel factory) with classifier models to estimate movements of stocks in the market.

As mentioned, more advanced financial time series forecasting research mostly focused on RNNs and LSTMs in recent years. However, models that integrate technical analysis data with deep neural networks is not very common in literature. A few that stand out for this approach are Mehtab et al. [6] who compared more traditional machine learning methods with a CNN on NIFTY market instruments and Sezer et al. [7] who proposed a deep neural network that uses financial technical analysis indicators like MACD, RSI and others to predict Dow30 stock prices turning points.

In this project I followed the research of Sezer et al. [7] with their novel technique by using a CNN network with a 2D representation of the technical analysis data and built on top of it a more advanced model architecture. The original paper achieved an overall accuracy score of 0.58 and a 0.22 precision score for BUY signals. These scores are used as baseline for the results in this project.

3. Dataset

As there was no publicly available dataset, I needed to create a new dataset. Since this was a very intensive preprocessing task and it's important to understand the data structure, I'll explain this in more detail.

Raw data was extracted from one of the major cryptocurrency exchanges like *Binance* [8] and then preprocessed into two distinct representations which are labeled correspondingly.

A) *Candle-Stick Charts Images:* candle-stick charts are one of the main visualizations in modern market data analysis and are very common among financial analysts. These charts beyond more visualize ups (green) and downs (red)

B) *2D-Indicators Images:* Market Indicators are secondary representations of market data. There's a vast number of possible indicators that are widely used like RSI, SMA, MACD [9]. These indicators are stacked into a 2D representation for different timeframes per indicator.

3.1. Raw data

Market raw data has been extracted from one of the major cryptocurrency exchanges for a time frame of Apr 2nd 2024 02:06am GMT to Apr 27th 2024 08:02am GMT. The data needed to be converted from JSON format into a flattened tabular structure.

TIME	OPEN	CLOSE	HIGH	LOW	VOL
1712023560	69338	69335.4	69340	69299.9	202.345
1712023620	69335.4	69311.7	69345.8	69311.7	132.542
1712023680	69311.7	69264	69323.2	69254.5	179.386
1712023740	69264	69250	69300.1	69245.9	195.638
1712023800	69250	69271.4	69299.9	69249.9	125.999
...	...	...	...	...	...

Table 1: BTC/USD Futures raw data.

I. *Time:* indicates the unix timestamp for the given prices.

II. *Open, Close, High, Low:* specific prices per timestep in USD-Tether. Close price will be mainly used in further processing.

III. *Vol:* the overall volume that has been traded during this timestep. A high volume indicates a lot of market activity.

3.2. Labeling

The labeling process of the raw data is based on a few parameters:

1) *R-Value:* the R-Value is calculated based on the close-price and is defined by the risk and reward ratio for a trade. Reward means how much financial gain a trade must achieve. Risk stands for the maximum drawback a trade can have before it must be closed negatively. The values for both ratios are defined as follows.

$$Reward\ ratio = \frac{Target\ Price}{Current\ Price} > 1.01$$

$$Risk\ ratio\ = -\frac{Reward\ ratio}{3}$$

2) *Lookahead window:* to set the correct label for timestep t we must look into the future and check whether we first find a close price that is higher than the reward ratio or if we first find a price at time t that is lower than the risk ratio. The Labels will then be set accordingly binary:

$$f(closes) = \begin{cases} 0, & close_t < Risk\ ratio\ or\ t\ > 180 \\ 1, & close_t \geq Reward\ ratio \end{cases}$$

The labeled dataset consists of 36k labels. There's a potential imbalance of negative vs positive labels which was challenging for training as can been seen later in the paper.

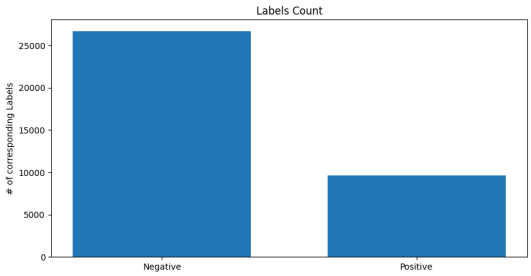

Figure 1: Number of labels per category.

3.3. 2D Indicators Images

I used different well known and widely used market indicators that are computed based on a time window and corresponding OHLC (Open, High, Low, Close) prices.

A) *SMA:* A way to reduce noise in data, and like momentum in an optimizer for gradient descent, I used a few SMAs to smoothen the close price over some time steps

B) *RSI:* The relative strength index is a momentum indicator that measures the speed and magnitude of a stock's recent price change to evaluate overvalued or undervalued conditions of that stock.[10]

$$RSI_t = 100 - \left[\frac{100}{1 + \frac{Avg\ Gain_{t-1} \times Steps + Gain_t}{Avg\ Gain_{t-1} \times Steps + Loss_t}}\right]$$

C) *Trendline:* To find a basic upward or downward trend, I used the slope over past timesteps.

D) *Standard Deviation:* As prices oscillate up and down, they tend to stay within upper and lower bounds over time. This can specifically be seen by the concept of Bollinger Bands.[9]

All these indicators are calculated with TA-Lib [11] based on the close prices of the given raw data and a corresponding set of past time windows. The used windows are of the following range:

$$T = [5, 10, 15, 21, 60, 80, 120, 200]$$

Overall, this resulted in a vector of 64 indicators per timestep. These indicators are then reshaped into a 2D representation of shape [H, W], where H = W = 8:

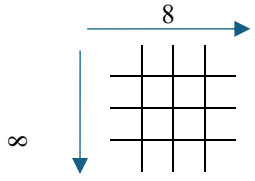

Figure 2: Example of Indicators 2D Image.

3.4. Charts Images

Charts are constructed as candle-stick charts with a secondary axis for volume.

The baseline for generating these charts is again the raw OHLC data. The time window is 60 minutes backwards in time and rolling over the whole dataset and thus generating $N = T - 60 + 1$ individual images. The new chart dataset has a shape of [N, 3, H, W].

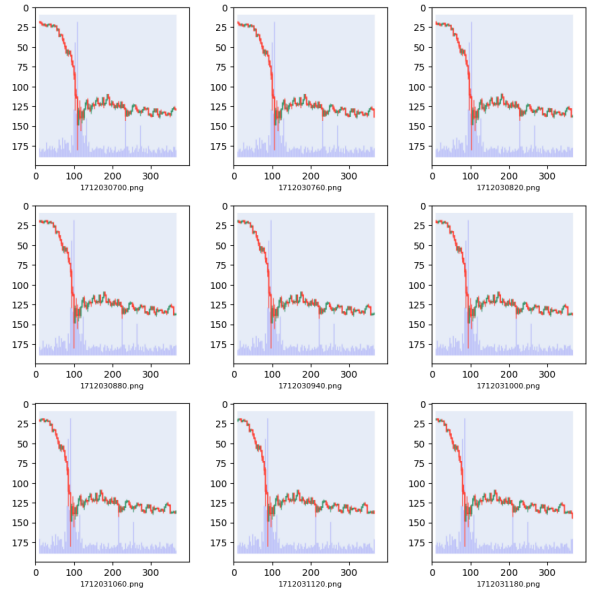

Figure 3: Generated Candle-Stick charts with volume bars.

4. Method

I used three different methods and compared them with each other to evaluate the best approach. The first method applied a CNN on generated candle stick chart images. The second method used a CNN on the 2D indicator images. Both of those methods are used as baseline for the third approach, which is a combination of the first and second method and defined as late fusion method.

4.1. Charts Images Baseline

I used a pretrained Resnet18 [12] on ImageNet and a shallow few layers plain vanilla CNN. As the dataset used in this case is very different from ImageNet, the Resnet unsurprisingly didn't perform very well.

As accuracy and precision for the Resnet was like the vanilla CNN, I continued just with the vanilla CNN as this was also faster to train. The vanilla CNN architecture was defined as follows:

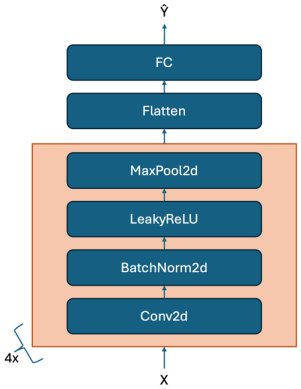

Figure 4 Chart Vanilla CNN Architecture.

Experiments with deeper CNN networks, like more Conv, BN, Activation, Pool blocks, had no significant impact on improving performance.

4.2. 2D Indicator Images Baseline

The architecture for this CNN model was derived from the work of O. B. Sezer and A. M. Ozbayoglu [7]:

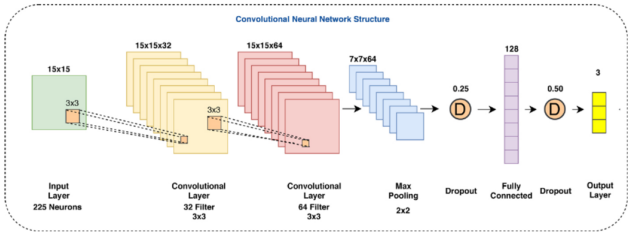

Figure 5: Indicator CNN Architecture.[7]

In addition, I changed to LeakyRelu as activation function to account for negative values, which come from one of the indicators like the trend indicator or due to normalization in the preprocessing.

To help the network to learn more easily, I also reordered the input data. I used a correlation matrix to realign columns to be closer to each other the more they correlated with each other. Especially for CNNs with their retrieval of spatial information, this helps to derive a better latent feature space.

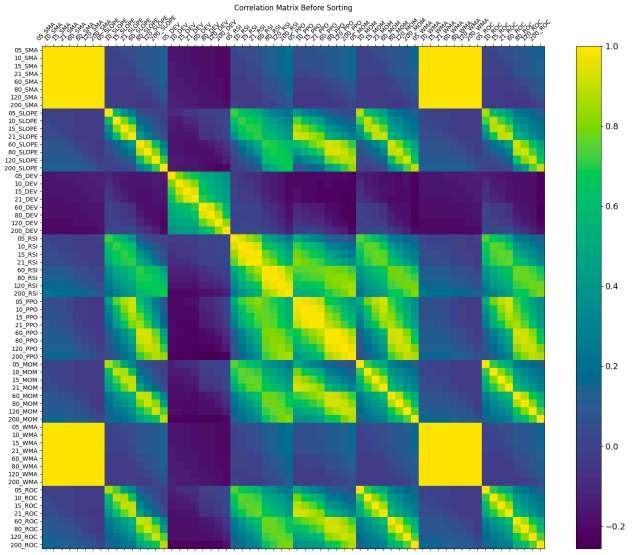

Figure 6: Indicators before Correlation

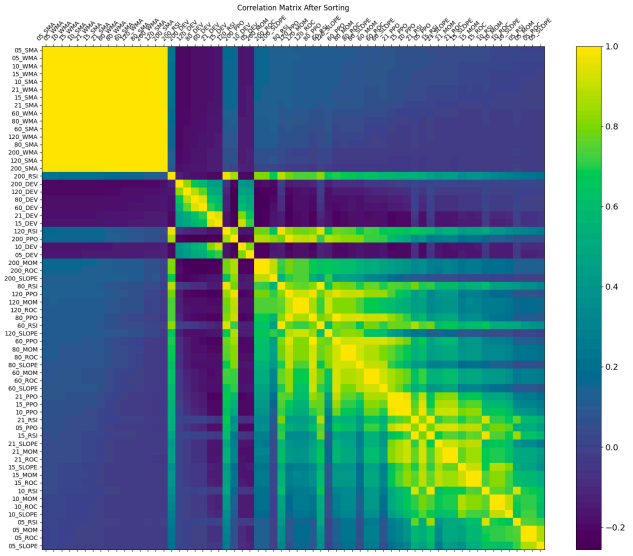

Figure 7: Indicators after Correlation and Reorder

4.3. Late Fusion

The method for the late fusion approach is aligned to the late fusion method for video classification [13]. The basic idea behind this architecture is to use the latent feature space of both baseline methods (indicator images and chart images) and combine them to achieve an overall better performance. The architecture is designed as follows:

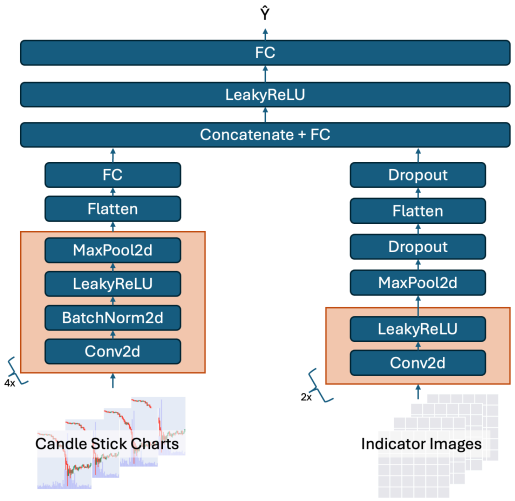

Figure 8: Late Fusion CNN Architecture

By concatenating the individual results and then using an additional fully connected layer, the model should be able to extract even richer feature information that hasn't solely been seen in just either one of the baseline models.

5. Experiments

Only focusing on accuracy as performance metric is not sufficient nor the right measure for this specific task. As the problem is defined by finding the right BUY signals, focusing on precision [14] is a better fit. Specifically, because True Positives vs. False Positives is more important than any True vs. False Negatives.

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Due to the size of the dataset and limited computational capacity, the dataset of 36,298 images was reduced to 3,629 (0.1) for train examples, 725 (0.02) for validation examples and 725 (0.02) for test examples.

I used a batch size of 32, Adam as optimizer and a maximum of 30 epochs.

To find the right regularization and learning rate, I used a random hyperparameter search. A learning rate of 1e-2 and weight decay of 1e-2 turned out to deliver the best results.

I used binary cross entry loss *with* logits [15] as loss function instead of basic binary cross entropy loss without logits as I wanted to use the same baseline network architectures without any modifications also for the fusion model. A sigmoid activation at the end of the charts and indicators network would have required removing the activation function for the fusion model.

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top$$
$$l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right]$$

First, I used the charts' images baseline model and trained it with the generated candle stick charts. Validation loss, precision and accuracy developed as follows:

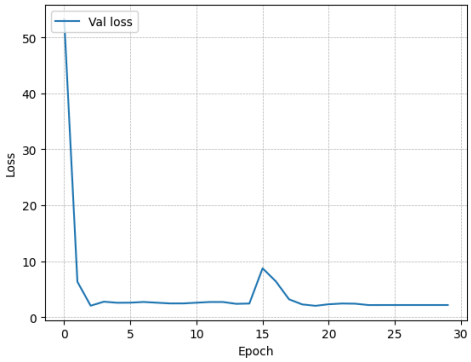

Figure 9: Candle Chart Validation Loss

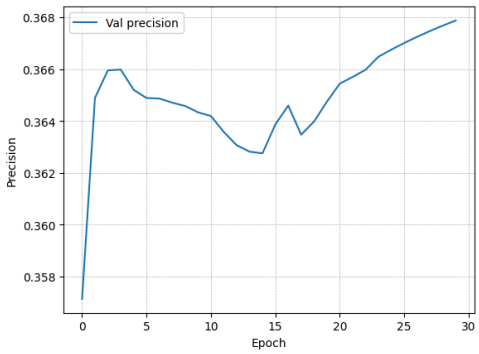

Figure 10: Candle Chart Validation Precision

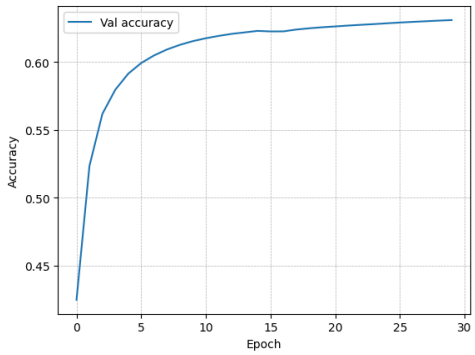

Figure 11: Candle Chart Validation Accuracy

Second, I used the indicators' images baseline model and trained it with the 2D market indicators data. Validation loss, precision and accuracy developed as follows:

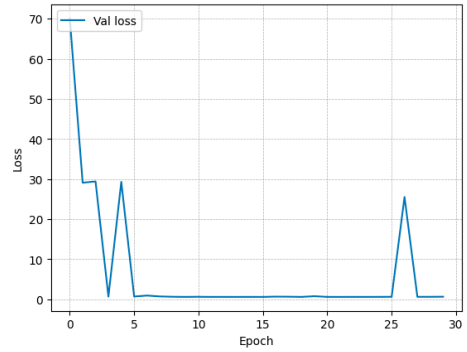

Figure 12: Indicator Validation Loss

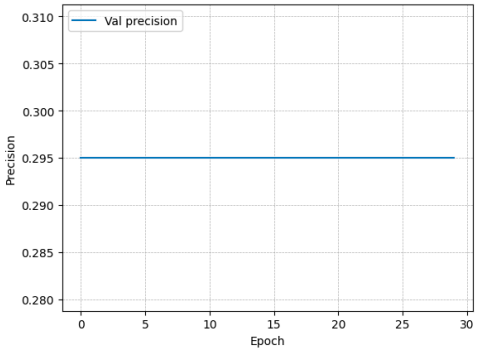

Figure 13: Indicator Validation Precision

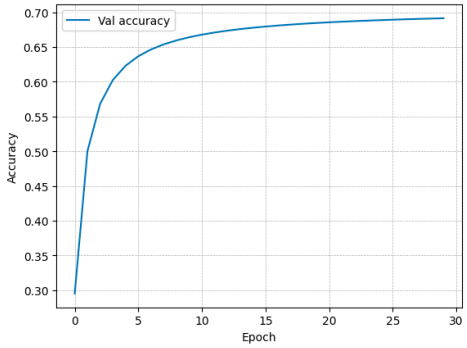

Figure 14: Indicator Validation Accuracy

Hyperparameter search was specifically hard for this 2D indicators network. Whereas loss and accuracy improved per epoch, precision stayed flat at 0.29.

Lastly, I trained on the fusion model and used the charts' images network in addition to the 2D indicators network. Validation loss, precision and accuracy developed as follows:

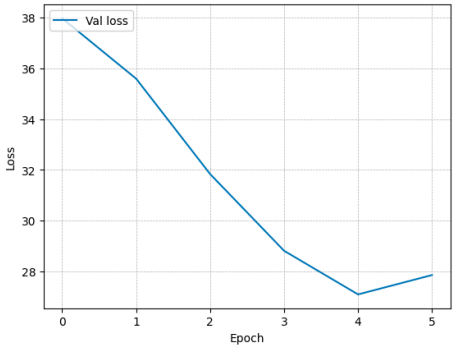

Figure 15: Late Fusion Validation Loss

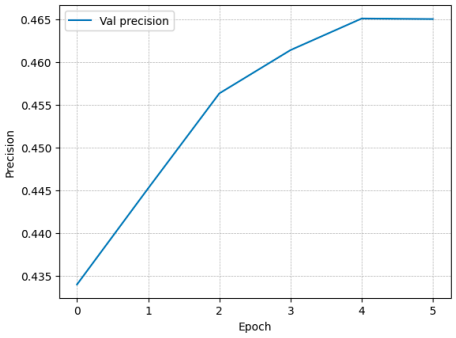

Figure 16: Late Fusion Validation Precision

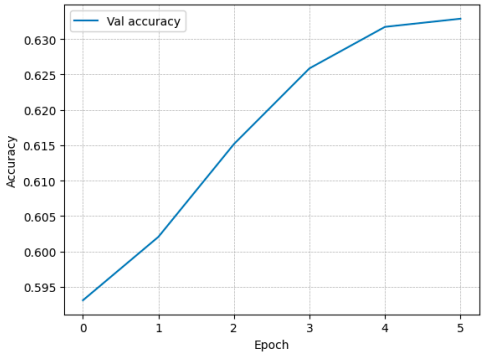

Figure 17: Late Fusion Validation Accuracy

Specifically, this model started to overfit very fast, so I applied early stopping as it converged already after 6 epochs.

6. Results and Evaluation

I used three different methods and compared them with each other to evaluate the best approach. On the test set, the following values have been the results:

	Precision	*Accuracy*
Chart CNN	**0.39**	0.63
Indicator CNN	**0.34**	0.66
Late Fusion CNN	**0.42**	0.60

Table 2: Comparison of Test Results

Mainly focusing on precision, the chart's CNN with a precision score of 0.39 did better than the Indicator CNN with a precisian score of 0.34. 0.42 is a good precision score for the late fusion model. Even though overall these precision values seem low there are a few characteristics why I still find them remarkable.

A) *Imbalance in dataset:* This normal imbalance in a dataset like this, leads in general to an optimization of true/false negatives over true/false positives. This can be specifically observed if I overfit to the train set and don't use any regularization like weight decay
B) *Labeling:* The labels are defined with a Risk-to-Reward Ration of 1-to-3 which means that statistically every precision beyond 0.33 will lead to more returns than losses over time if applied in a live trading environment.

Even though I was able to outperform the results for the precision score of 0.22 from the original paper by O. B. Sezer and A. M. Ozbayoglu [7] this needs to be put into perspective that I used a different market instrument like BTC/USD Perpetuals, and I also applied it in a different time window.

7. Conclusion and Future Work

The approach of combining two latent spaces and two distinct representations of market data seems promising to add confluence to a successful trading strategy.

This approach should now be back-tested with more historical data and a platform like backtrader [16] to apply the model on other market instruments and / or to calculate the true monetary reward if the model would be applied to a live trading strategy.

In addition, the performance of the fusion model could be compared to other more recent architectures like Transformers [17]. Vision Transformers like ViT [18] made good advances to applying transformers also to images. Therefore, these architectures could be applied on either one of the charts or the indicators input data or again combined like the fusion model described in this paper.

References

[1] I. Goodfellow, J. Bengio, A. Courville. Deep Learning. Das umfassende Handbuch, 2018
[2] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung. Time series analysis: forecasting and control, 2015
[3] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, A. L. Oliveira. Computational intelligence and financial markets: A survey and future directions, Expert Systems with Applications 55, 2016
[4] E. Guresen, G. Kayakutlu, T. U. Daim, Using artificial neural network models in stock market index prediction, Expert Systems with Applications 38, 2011
[5] M. Ballings, D. Van den Poel, N. Hespeels, R. Gryp. Evaluating multiple classifiers for stock price direction prediction, Expert Systems with Applications 42, 2015
[6] S. Mehtab, J. Sen. Stock Price Prediction Using Convolutional Neural Networks on a Multivariate Timeseries, 2020. https://arxiv.org/abs/2001.09769
[7] O. B. Sezer, A. M. Ozbayoglu. Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach, 2018. https://www.researchgate.net/publication/324802031_Algorithmic_Financial_Trading_with_Deep_Convolutional_Neural_Networks_Time_Series_to_Image_Conversion_Approach
[8] Binance Cryptocurrency, 2024. https://www.binance.com
[9] J. J. Murphy. Technische Analyse der Finanzmärkte, 2006
[10] M. J. Pring. Technical Analysis Explained, 2014
[11] TA-Lib - Technical Analysis Library, 2024 https://github.com/TA-Lib/ta-lib-python
[12] K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition, 2020. https://arxiv.org/abs/1512.03385
[13] Fei-Fei Li, Ehsan Adeli. Lecture 10: Video Classification: Late Fusion, April 2024
[14] M. Grandini, E. Bagli and G. Visani. Metrics for multi-class classification: an overview, 2020. https://arxiv.org/abs/2008.05756
[15] Pytorch. Binary Cross Entropy Loss With Logits, 2024. https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html
[16] D. Rodriguez. Backtrader Python Library Documentation, 2024. https://www.backtrader.com/docu/
[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017. https://arxiv.org/abs/1706.03762
[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2020. https://arxiv.org/abs/2010.11929