

Financial Time Series Prediction using Deep Learning

Ariel Navon*, Yosi Keller†

Abstract

In this work we present a data-driven end-to-end Deep Learning approach for time series prediction, applied to financial time series. A Deep Learning scheme is derived to predict the temporal trends of stocks and ETFs in NYSE or NASDAQ. Our approach is based on a neural network (NN) that is applied to raw financial data inputs, and is trained to predict the temporal trends of stocks and ETFs. In order to handle commission-based trading, we derive an investment strategy that utilizes the probabilistic outputs of the NN, and optimizes the average return. The proposed scheme is shown to provide statistically significant accurate predictions of financial market trends, and the investment strategy is shown to be profitable under this challenging setup. The performance compares favorably with contemporary benchmarks along two-years of back-testing.

1 Introduction

Time series analysis is of major importance in a gamut of research topics, and many engineering issues. This relates to analyzing time series data for estimating meaningful statistics and pattern characteristics of sequential data. In particular, the forecasting of future values based on previously observed measurements. For instance, given the samples of a discrete signal $\mathbf{x}(n)$, $n \in \mathbb{Z}$, the forecasting task aims to estimate $x(n+T)$, where $T > 0$. Alternatively, we aim to estimate other parameters of $\mathbf{x}(n')$, $n' > n$, such as the standard deviation of $\mathbf{x}(n')$ denoted as $\sigma(\mathbf{x}(n'))$, or future time series trends,

$$\hat{y}(n|\mathbf{x}, T) = \text{sign}(\hat{x}(n+T) - x(n)). \quad 1.1$$

A loss function, also denoted as a cost function,

$$\mathbb{L}(n) = \mathbb{L}(y(n), \hat{y}(n))$$

is defined to quantify the accuracy of the resulting prediction $\hat{y}(n)$, with respect to the ground-truth $y(n)$.

*Faculty of Engineering, Bar Ilan University, Israel. navon2@gmail.com.

†Faculty of Engineering, Bar Ilan University, Israel. yosi.keller@gmail.com.

Numerous works studied time series data, by applying statistical approaches. The Kalman Filter [1] and Auto-Regressive (AR) Models are seminal statistical approaches, where the ARMA [2] and ARIMA [3] (Auto-Regressive Integrated Moving Average) models are further generalizations of the AR, derived by combining the AR models with the moving-average (MA) models. Other common AR schemes is the ARCH (Autoregressive Conditional Heteroskedasticity), GARCH [4] (Generalized ARCH) and the ARIMA algorithms, and are often applied to financial series forecasting.

Financial time series analysis deals with the extraction of underlying features to analyze and predict the temporal dynamics of financial assets. Due to the inherent uncertainty and non-analytic structure of financial markets [5], the task proved to be challenging, where classical linear statistical methods such as the ARIMA model, and statistical machine learning (ML) models have been widely applied [6, 7].

Deep-Learning (DL) [8] approaches that relate to computational algorithms using artificial neural networks with a large number of layers, allows to directly analyze raw data measurements without having to encode the measurements in task specific representations. In this work we propose to harness DL networks to efficiently learn complex non-linear models directly from the raw data (an "end-to-end" approach), for the prediction the financial market trends. We focus on direct prediction of Standard & Poor's 500 (denoted as: "S&P500") index and assets trends based on raw-form data of equities rates, and utilize probability estimates given by the trained neural network.

In particular, we propose the following contributions:

First, we present an end-to-end learning scheme based on the raw rates data of assets, in contrast to previous financial forecasting schemes where the analysis utilized multiple engineered features [9].

Second, we propose a trading strategy that utilizes the probabilistic predictions of the neural network model, to determine the entry and exit points of a trade.

Last, the proposed trading system is shown to be profitable while outperforming baseline schemes, as demonstrated in a challenging realistic commissions charged trading environment.

The rest of this work is organized as follows. Prior works in time-series forecasting and applying machine learning to financial data are surveyed at Chapter 2. The proposed Deep-Learning model end-to-end approach for financial trends forecasting is introduced in Chapter 3, and the probabilistic trading strategy is proposed in Chapter 4. These schemes are experimentally verified in Chapter 5, by applying them to several stocks assets from the S&P 500 index, using realistic transaction costs.

2 Background

In this chapter we review previous works in time series forecasting in general, and financial forecasting in particular. Section 2.1 surveys machine-learning schemes for financial forecasting, while Deep-Learning schemes for financial data are discussed in Section 2.2.

2.1 Machine Learning Approaches For Financial Data Forecasting

Machine learning algorithms are often applied to time series forecasting in general, and financial time series prediction. Kanas [10] showed that the non-linearity of the models being used for time series forecasting is of major importance. Thus, one of the most commonly applied schemes is the K-Nearest neighbors (kNN). The kNN algorithm assumes a similarity between time series sequences that occurred in the past, to future sequences, and as such, the nearest-neighbors are used to yield the forecasts of the kNN model. Ban et al. [11] applied kNN to multi-asset data, utilizing a set of stocks sharing similar dynamics. Thus, achieving less bias and improved resiliency to temporal fluctuations, than those of a single stock.

Hidden Markov Models (HMM) are also commonly applied to financial time series forecasting. A HMM encodes a finite-state machine with a fixed number of non-observable states. These hidden variables are assumed to be related by a Markov process allowing HMMs to be applied to temporal pattern recognition tasks. Hassan [12] applied HMMs to forecasting stock market, by training an HMM model on a specific stock, and matching past temporal patterns to current stocks patterns. The prediction is derived by extrapolating current prices based on past events. Decision Trees [13] and SVM [14, 15] were also applied to time series forecasting.

2.2 Financial Time Series Analysis Using Deep Learning

Deep Learning (DL) techniques were successfully applied to a gamut of problems such as computer vision [16, 17], automatic speech recognition [18, 19], natural language processing [20, 21, 22], handwriting recognition [23], and bio-informatics [24], to name a few, outperforming contemporary state-of-the-art schemes. Yet, the use of DL in financial time-series data is not widespread. Nevertheless, some DL schemes were applied to financial data, utilizing text-based classification, portfolio optimization, volatility prediction and price-based classification.

Ronnqvist [25] proposed a DL approach for estimating financial risk based on the news articles. Publicly available textual data was used to quantify the level of banking-related reports, and a classifier was trained to classify a given sentence as distress or tranquility. For that two NNs were applied, the

first aims to reduce dimensionality by learning a semantic representation, while the other NN is trained to classify the learned representation of each sentence. Fehrer and Feuerriegel [26] trained a text-based classifier to predict German stock returns based on news headlines, and reported 56% accuracy on a three-class prediction of the following trade day, without developing a trading strategy. Ding [27] studied a similar topic by using structured information extracted from headlines to predict daily S&P 500 movements.

Portfolio optimization is the optimal dynamic selection of investment assets, and was studied by Heaton [28], by trying to predict a portfolio that will outperform the biotechnology index IBB. For that, an autoencoder was trained, using the weekly return data of the IBB stocks 2012-2016. All stocks in the index were autoencoded, and the stocks found to be most similar to their autoencoded representation were chosen.

Xiong [29] applied Long-Short-Term Memory (LSTM) neural networks to model the S&P500 volatility, using the Google stock domestic trends as an indicator of the market volatility. Thus, reflecting multi-parameters macro-economic status, as well as the public mood, and outperforming benchmarks such as linear Ridge/Lasso and GARCH, with respect to the mean average absolute error percentage. A deep neural network for financial market trends prediction was proposed by Dixon [9] that trained a prediction model using multiple financial instruments (including 43 commodity and forex assets), aiming to classify the future trend as either positive, flat or negative. The dataset consisted of aggregated feature training sets of all symbols, each encoded by 9895 engineered features and price differences. The neural net consists of five fully connected layers and the model is shown to predict the instrument's trend, while ignoring transaction costs.

3 Deep Learning Prediction of Stock Price Trends

In this work we aim to derive a DL based prediction model to forecast the future trends of financial assets, based on the raw data. The proposed scheme is a dynamic probabilistic estimation of the asset price trend, that can be applied to active trading, by entering either a long or short position, while determining the exit point given an open position is introduced in Chapter 4.

The term *long trade* refers to the operation of buying an instrument for the sake of later selling it in a higher price. Thus, it is used when a positive trend is expected, and its potential loss is limited to the cost of the trade. A *short trade* is opened when a price decline is expected, by first borrowing a financial instrument, and later closed by buying back the instrument originally borrowed. The losses from short positions are unbounded, as future prices are unbounded. The terms *buy* and *long* are used

often interchangeably, as well as the terms *sell* and *short*.

Let $\mathbf{x}(n)$, $n \in \mathbb{Z}$ be a discrete financial time-series signal, such as the historic closing prices of an asset, and $y(n|\mathbf{x}, T) \in \{-1, 1\}$ is the temporal trend such that

$$y(n) = \begin{cases} 1, & \text{if } x(n+T) > x(n) \text{ positive trend} \\ -1, & \text{if } x(n+T) \leq x(n) \text{ negative trend} \end{cases} \quad 3.1$$

where $T > 0$ is the prediction interval.

Let $\hat{x}(n)$ and $\hat{y}(n)$ be the estimated signal and trend, respectively. We aim to minimize the softmax loss function $\mathbb{L}(n) = \mathbb{L}(y(n), \hat{y}(n))$ trained using the price logs data $\mathbf{x}(n)$

$$\mathbb{P}(y = j|\mathbf{x}(n)) = \frac{e^{Z_j}}{\sum_{k=1}^K e^{Z_k}} \quad 3.2$$

for $j \in 1, \dots, K$, where \mathbf{Z}_k are the inputs to the softmax layer of the neural network. For the Up/Down two-classes classification problem, $K = 2$, and the logistic loss is given by

$$\mathbb{L} = -\log\left(\frac{e^{Z_i}}{\sum_{k=1}^2 e^{Z_k}}\right) \quad 3.3$$

where i is the ground truth label $y(n)$. The proposed scheme utilizes both the *hard* and *soft* estimates of $y(n)$, and we denote the soft estimate $p(y|\mathbf{x})$.

3.1 Deep Learning-based Prediction of Price Trends

The proposed scheme, depicted in Fig. 1, consists of two phases. The first, detailed in Section 3.2, aims to predict the signal's trend $y(n)$ and corresponding probability $p(y|\mathbf{x})$, while the second, discussed in Chapter 4 applies the predicted trend to derive an investment strategy, operating in a commissions-charged trading environment, where in each time step one can either buy/hold/sell the asset. We start by preprocessing the price history input data $\mathbf{x}(n)$ into a M -length features vector, consisting of sequential normalized price values, and the temporal gain is given by

$$\mathbf{g}^n = \hat{y}(n) \cdot (x(n+T) - x(n)). \quad 3.4$$

3.2 Deep Learning Model for Price Trends Prediction

In order to predict the future stock price trend $\hat{y}(n)$ we apply a classification neural network trained using the raw prices data of the previous M minutes

$$\hat{y}(n) = \hat{y}(n|x^{n-M+1}, \dots, x^n).$$

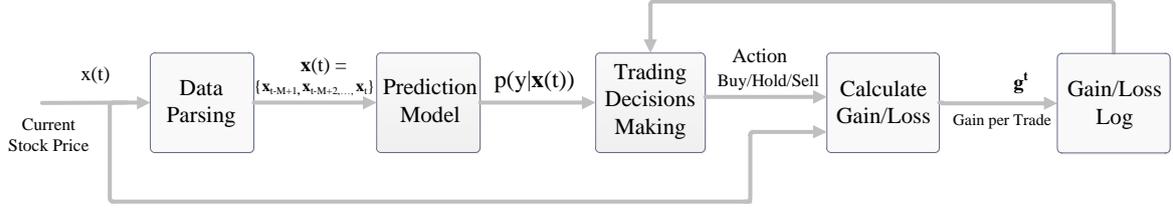


Figure 1: Schematic Illustration of our Deep-Learning based financial trends prediction system

For that we applied the Neural Net depicted in Fig. 2, and also experimented in utilizing convolutional layers, though it did not yield significant accuracy improvement, due to the low dimensional features space.

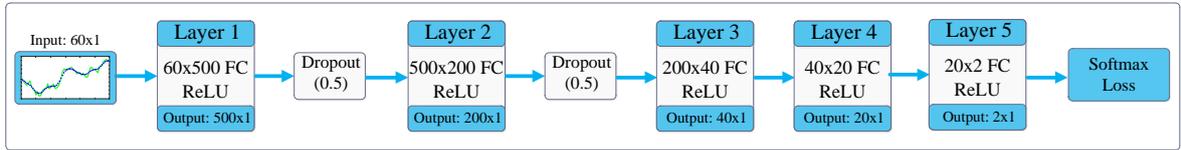


Figure 2: The neural network applied to predict the price trends.

3.3 Preprocessing

Let $\mathbf{x}(n)$ be the input to the net consisting of closing price data of S&P500 assets in one-minute resolution. For each data point $\mathbf{x}(n) \in \mathbb{R}^{60}$, we used the raw closing prices of the previous $M = 60$ minutes. The dataset was preprocessed by parsing the prices-data, where each sample $S(n)$ encodes a particular minute during the trading hours of a trading day. Each such sample is composed of its preceding 60 data points $\mathbf{x}(n)$ and a label $y(n)$ representing the trend. Our approach relates to intra-day trading, and was trained on data belonging to the same trading day. In order to avoid irregular trading periods, we omitted trading days that belong to earning publication periods, as well as omitting days with partial trading hours. In order to avoid overfitting we used Dropout layers [30] after the first and second layers, as well as early-stopping of the training, based on the validation set. The data set was divided to temporally non-overlapping, training, validation and test sets.

3.4 Labeling the Financial Dataset

The duration of the prediction interval T , relates to the correlation between the features $\mathbf{x}(n)$ (past intra-day price values) and the future values $y(n + T)$, that decreases as the prediction interval T value

increases. In contrast, the financial price variations are more significant for larger prediction intervals. This trade-off was resolved by studying multiple datasets using $1 \leq T \leq 30$ minutes, where a different model was trained for each dataset, and T was chosen by cross-validation, as detailed in Table 1.

4 Probabilistic Trading Strategy

In order to apply the proposed DL prediction model to financial data, we derive a buy-hold-sell probabilistic trading strategy, aiming to maximize the overall cumulative return

$$\mathbb{G} = \sum_{n=1}^{n=N_{Testn}} \mathbf{g}^n \quad 4.1$$

along the back-testing period ($1 \leq n \leq N_{Testn}$), where the temporal gain \mathbf{g}^n is given in Eq. 3.4. The main challenge in achieving trading profitability, is the *transactions costs* T_C , charged per each transaction, as the average likelihood ratio

$$R = \frac{E\{p(\hat{y} = y|\mathbf{x})\}}{E\{p(\hat{y} \neq y|\mathbf{x})\}} \quad 4.2$$

between the probabilities of correct and incorrect predictions is typically just a few percents. Moreover, the average intra-day volatility is relatively small

$$T_C \sim P(n + T_{intraday}) - P(n), \quad 4.3$$

where $T_{intraday}$ indicates a typical intra-day trading period. We generalize the definition of \mathbf{g}^n to \mathbf{g}_{TC}^n taking into account the transaction costs

$$\mathbf{g}_{TC}^n = \hat{y}(n) \cdot (x(n + T) - x(n)) - T_C, \quad 4.4$$

such that we refer to a unified transaction cost for buy&sell transactions involved in a single trade.

We aim to identify the subset of the profitable transactions, denoted by $\alpha_{p(y|\mathbf{x})} \in \{0, 1\}$, such that $\alpha_{p(y|\mathbf{x})} = 1$ implies a profitable transaction, and $\alpha_{p(y|\mathbf{x})} = 0$, relates to a non-profitable one. The cumulative gain is thus given by

$$\mathbb{G} = \sum_{n=1}^{n=N_{Testn}} (\mathbf{g}_{TC}^n \alpha_{p(y|\mathbf{x})}^n). \quad 4.5$$

The proposed Probabilistic Trading Strategy utilizes the soft-information $p(y|\mathbf{x})$ of the neural network output, used to estimate $\alpha_{p(y|\mathbf{x})}$. Section 4.1 introduces the trading strategy with respect to *opening* a trade (when to “buy”), while Section 4.2 discusses the closing of a trade (when to “sell”). Using long and short trades, the position could be opened based on an either positive or negative expected price trends.

4.1 Trade Opening Using Soft-Information

The use of the soft-information $p(y|\mathbf{x})$ provided by the DL model allows to select a subset of the trades with higher prediction accuracy, improving the gain \mathbb{G} while taking the commissions into account. For that we consider the classification margin, such that

$$\alpha^n = \begin{cases} 1, & \text{if } \|p_2(y|\mathbf{x}) - p_1(y|\mathbf{x})\| \geq T_H \\ 0, & \text{if } \|p_2(y|\mathbf{x}) - p_1(y|\mathbf{x})\| < T_H \end{cases}, \quad 4.6$$

where T_H is determined by cross-validation.

Due to the nonstationarity of the financial process, applying Eq. 4.6 using a fixed threshold T_H , might prove inaccurate. Hence, we propose to compute an adaptive threshold T_H by cross validation, where we apply T_H to estimate the median gain over the D previous data points. The value of D was also estimated using cross-validation and the validation set. We also considered a greedy approach of choosing a threshold T_H that maximizes the gain of the D prior points, but it was shown to be unstable and less accurate.

The use of the adaptive threshold T_H allows to chose a subset of profitable trading points. However, higher (and more realistic) trading commissions (above 0.05% of the transaction volume) require further screening of the active trading points α^n . For that we propose to avoid opening a new trades between $n \in [n, n + N_{SafeLen}]$ when the number of losses (negative gain trades) exceeds a predefined threshold denoted as $N_{SafeTrigger}$

$$\sum_{n-N_{SafeWin}+1}^n (\mathbf{g}_{TC}^n < 0) \geq N_{SafeTrigger} \quad 4.7$$

The last screening mechanism, targeted to experimentally avoid missclassified patterns, is noted as "safety-switch scheme". The resulting trade opening scheme is depicted in Fig. 3.

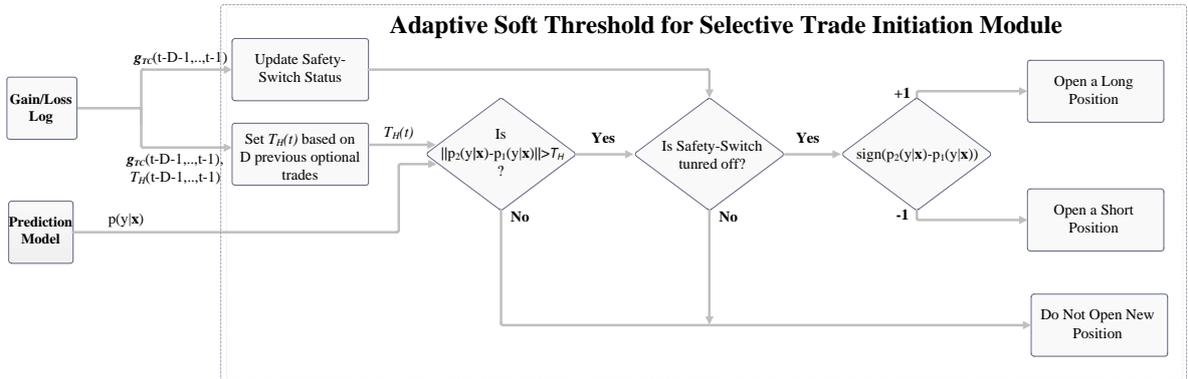


Figure 3: The proposed soft-threshold based trade opening approach.

The safety-switch scheme significantly reduces the number of trade openings, down to $\sim 1\%$ of the overall number of the data points, as depicted in Fig. 5. Thus, choosing the more reliable predictions of the net. We attribute that to the low certainty of the trend prediction DL scheme that is able correctly classify only 53% of the time slots, and to the high T_C value comparing to the temporal gain value g^n of a single trade.

4.2 Setting the Trade-Length

We study the deal closure (“sell”) event at time $n_c > n_o$, where n_o and n_c are the deal opening and closure times, respectively. Given the prediction interval T , mentioned in Section 3.4, we considered three options for choosing the end point of each trade. First, closing the trade after T minutes regardless of the predictions of $p(y|\mathbf{x})$, such that

$$n_c = n_o + T \quad 4.8$$

Second, closing the trade at n_c such that the hard decision sequence changes

$$\hat{y}(n_c) \neq \hat{y}(n_o). \quad 4.9$$

Last, we tested waiting T minutes after the hard decision sequence change as in Eq. 4.9. We found Eq. 4.9 to be the most accurate, and it allows adaptive trading durations of varying lengths.

5 Experimental results

The applicability of the proposed DL scheme was studied by splitting the dataset to temporally non-overlapping training, validation and test sets. The test set consists of the data from June 23rd 2014 to June 22nd 2016. The validation set is based on the data during the previous calendar year, from June 23rd 2013 to June 22nd 2014, and all previous data was used as a training set. Thus, our model does not account for changes in the market dynamics, and implicitly assumes that a strategy learnt using the data up to 2013, can be applied to trades in 2016.

We used the closing price data for trading, and chose assets with high trading volumes, that are sufficiently liquid, such that the orders are always executed on time (“Spread” and “Slippage” effects are ignored).

In each trade we invested an equal sum, and both the gain results and the transaction costs are measured in percentage of this sum, where we apply a combined commission rate for both buy and sell actions. We report the cumulative gain over the two-years test period, that is the sum of the gain of all active trades during that period of time.

5.1 Experimental Setup

The proposed schemes were experimentally verified by using the market data of the Standard & Poor’s 500 (“S&P 500”) assets given in one-minute resolution, purchased from the QuantQuote market data supplier [31]. For each asset analyzed, we utilized all available history data of full-trading days, based on regular trading hours (9:30AM-4:00PM), and ignoring off-hours trading data. We also omitted days with partial trading hours, as well as earning-publication periods that are usually up to 2-3% of the data points, and are given in the data set.

As we study intra-day trading, the data was divided to different trading days, where all features were derived from the current trading day, with no overnight trades allowed. Each trading-day data is parsed into ~ 300 different data samples, and we used the raw closing price data in the preceding $M = 60$ minutes, as features. Each input sequence was filtered by five taps long, moving uniform averaging, and then normalized by reducing the mean, and dividing by its standard deviation.

The training labels were set as in Eq. 3.1, and multiple models were trained for varying T values. For each asset we chose T using cross-validation, where common values range between 20 – 30 minutes, as depicted in Table 1.

We applied over-sampling to balance the training and validation sets, such that there will be an equal number of positive and negative training samples for the trend prediction. The overall size of the data sets after parsing and balancing depends upon the dates of available data for each asset, the model parameters, and trading strategy. For the SPY ETF (detailed in Section 5.3) with $T = 1$ there are $\sim 1.2M$ samples before balancing, divided to 967K training, 81K validation, and 164K test samples. The test set was not balanced.

The prediction model is based on the neural network depicted in Fig. 2 that was trained using the MatConvNet package [32] with mini-batches of size 100, where the learning rate was adaptively reduced, by a factor of 5, when observing a flat error reduction of the validation set during training. The validation set was also used for early-stoppage. The reliability soft-information provided by the model is used by the proposed probabilistic trading strategy for detecting high confidence trades.

5.2 Implementation details

The network architecture for our base NN model is shown in Fig. 2. It has 5 fully-connected layers with 500, 200, 40, 20 and 2 ReLU activation units. Dropout follows the first and second layers. The input is a 60 1 low-pass-filtered adjacent historical raw prices values. We apply Stochastic-Gradient-Decent to train the models with 100 batch size and 0.5 dropout rate. Initial learning rate is 0.001 and decreased

by a factor of 5 as validation errors stop decreasing, down to learning rate of $1e-7$. All initial weights are sampled from Gaussian distribution with 0.01 standard deviation. We implemented the system in the MatConvNet framework. The average inference time per input single price vector is 0.2ms on single Nvidia GeForce GTX 780 GPU for the overall system framework.

5.3 Active Trading of SPY ETF

We exemplify our schemes by applying the proposed active-trading to the SPY ETF, that is the SPDR exchange-traded fund (ETF) - a Standard & Poor’s Depository Receipts, designed to track the S&P 500 stock market index, allowing investors to buy/sell the entire S&P 500 index using a single instrument. This is one of the ETFs with the largest trading volumes and liquidity. The learning period D and the position holding time T were set to $D = 5$ and $T = 28$, respectively using cross-validation, as shown at Table 1.

During the testing period consisting of the last two years of our data (23.6.2014-22.6.2016), the SPY price increased by 10.63%. Figure 4 depicts the cumulative trading gain over the test period for different models vs. the prediction range T . We trained a different DL prediction models for each value of T , and report the results for different transactions costs. It follows that the DL model chosen by setting D and T according to the validation set (Table 1) yields a cumulative gain of 61.1% over the testing period for commission of 0.1%.

	D=1	D=5	D=10	D=50
T=1	1.5	-0.8	-0.5	0
T=4	0.5	0.1	-0.8	-3.3
T=8	1.8	0	-0.6	-2.5
T=12	3.8	2.7	2.3	-1.2
T=16	0	2.3	2.1	0.3
T=20	-3.7	6.8	3.1	-7.8
T=24	5.7	20.2	20.1	-8.9
T=28	2.9	21.2	18.9	-3.7
T=32	5.1	16.9	16.5	-8.2

Table 1: Cross validation results for the trade labeling length T and adaptive threshold window length D . We report the cumulative gain of the proposed active-trading strategy over a one-year validation period, and a commission of 0.08%.

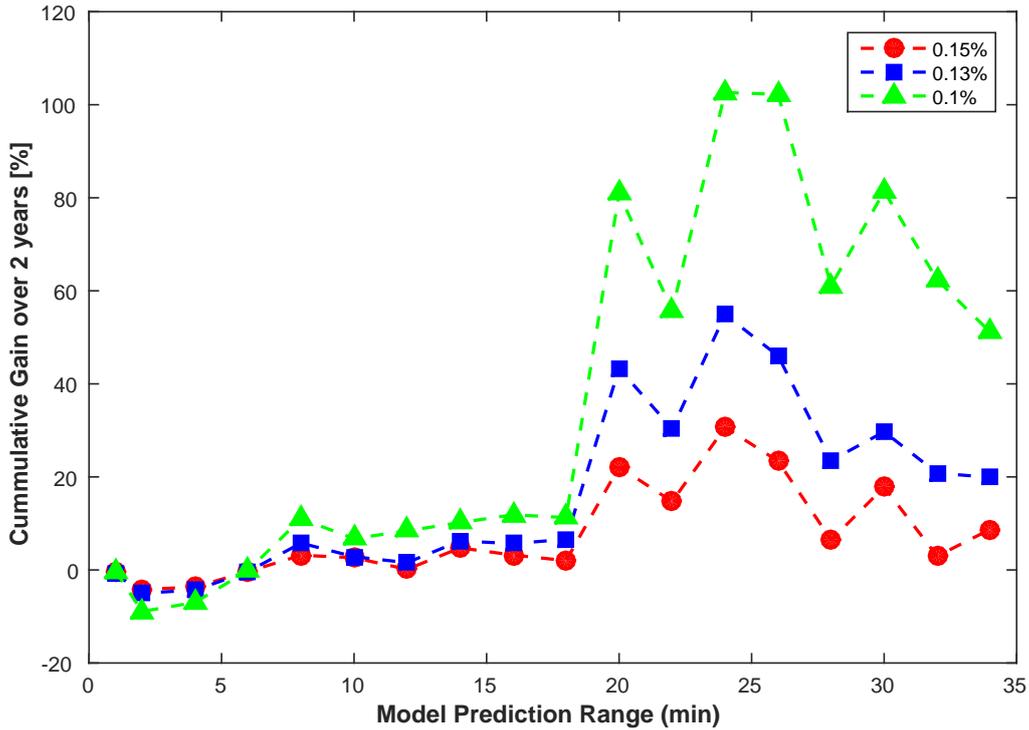


Figure 4: Cumulative trading gain with SPY ETF over two years for different model prediction ranges. We report the results for different commission rates.

We show the effect of the different phases of the proposed trading scheme in Fig. 5, that depicts the histogram of the gain of trades, when applying different components of the proposed scheme. For that we compared the following schemes:

- Fixed trading length of $T = 1$. No adaptive-soft-threshold and safety-switch mechanisms for selective trade initialization.
- Varying trade length, by closing the trade when the model forecasting sequence changes direction, without an adaptive-soft-threshold and a safety-switch.
- Varying trade-length and adaptive-soft-thresholds, without a safety-switch.
- Utilizing all of the proposed components as in Section 4.

It follows that the distribution of the proposed scheme outperforms the other schemes, where a significant part of the distribution is concentrated at the zero-gain bin, corresponding to the underlying assumption of the proposed approach, as only trades probable to achieve a positive gain are chosen.

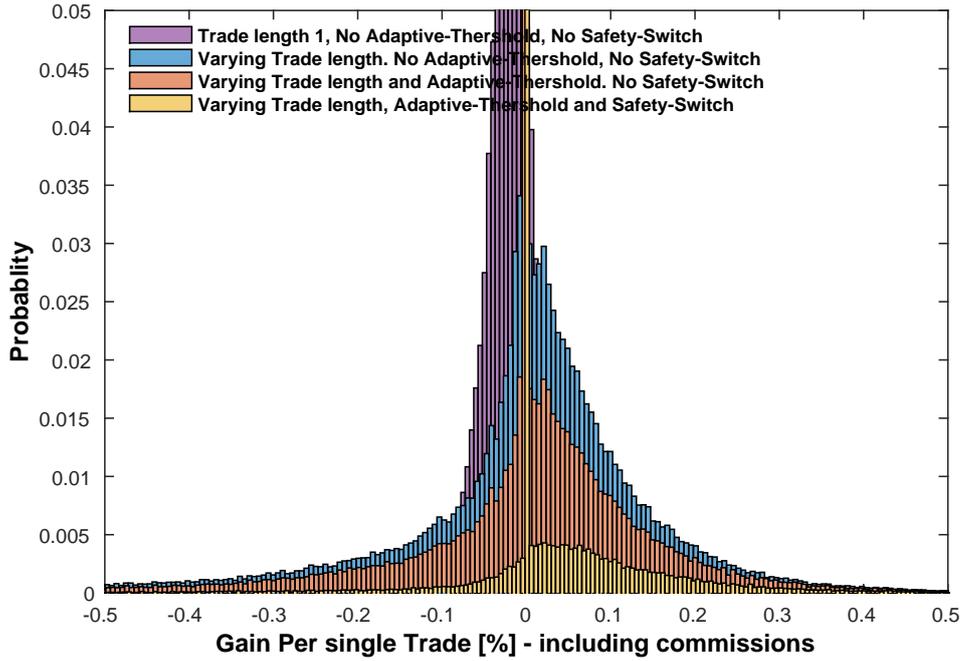


Figure 5: The probability of returns-per-trade over the test period. We report the results for different configurations of the proposed scheme.

5.4 Comparison to Benchmarks

In order to evaluate the performance of our Deep-Learning based active trading approach, we compared against the Support-Vectors-Machine (SVM) scheme, that is considered a state-of-the-art classification scheme. The results over the test period are presented in Fig. 6, where in each time point we report the cumulative gain, starting with the first test day. Both active-trading strategies (DL-based and SVM-based) presented in Fig. 6, include transactions costs of 0.06% (buy&sell), where no commissions were applied for the buy-and-hold strategy.

The Kernel SVM with a RBF kernels was implement by applying PCA to the parsed data base, while preserving 95% of the data variance, thus reducing the dimensionality from 60 to 7. Higher values of the preserved variance lead to inferior results. The classification probability estimate of the SVM was used by the proposed Probabilistic Trading Strategy. These results were compared to the baseline asset price change, corresponding to the buy-and-hold strategy.

Figure 7 shows that the proposed DL scheme outperforms the SVM-based model, and the asset-baseline benchmark. However, as the transaction costs increase - the profitability of commission-based models such as ours is reduced, leading to inferior performance compared to the buy-and-hold passive strategy.

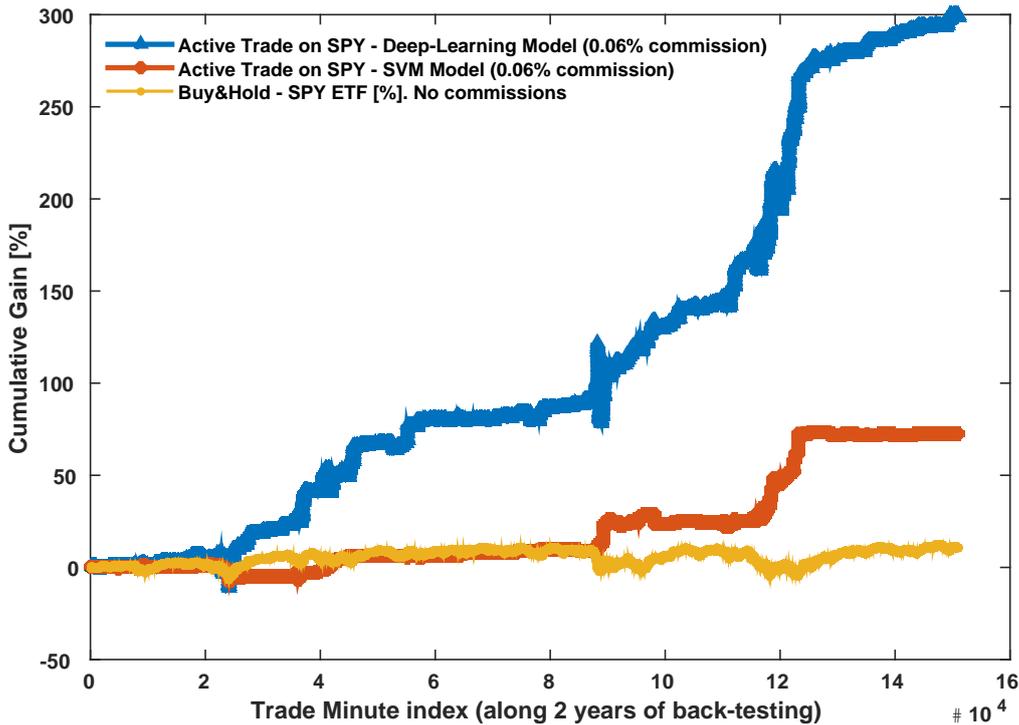


Figure 6: Cumulative tradings gain. We report The performance of the proposed Deep-Learning based scheme, a SVM prediction model utilizing the proposed probabilistic trading strategy, and a buy-and-hold strategy.

5.5 Active Stocks Trading

In this section we apply the proposed scheme to nine stocks from the S&P 500 index: INTC (Intel Corporation), AAPL (Apple Inc.), GOOGL (Google), BAC (Bank of America Corporation), AMZN (Amazon.com Inc.), KO (The Coca-Cola Company), T (AT&T Inc.), JNJ (Johnson & Johnson), BA (The Boeing Company). We applied the same scheme as in the previous section, where a model was trained for each stock separately. The test and validation periods are the same as those of the SPY ETF.

The cumulative gain over the two-years test period versus different transaction commissions for the different instruments (including SPY) is shown in Fig. 8, where it follows that the cumulative gains differs for different instruments, and while positive results were achieved, they are strongly depended on the transactions cost.

Further analysis of is given in Fig. 9, where we study the daily returns, assuming unified buy and sell commissions of 0.1%. Figure 9a reports the mean and standard deviation (volatility) of the daily gain results of the different instruments, while Figure 9b depicts the median, the 25% and 75% percentiles

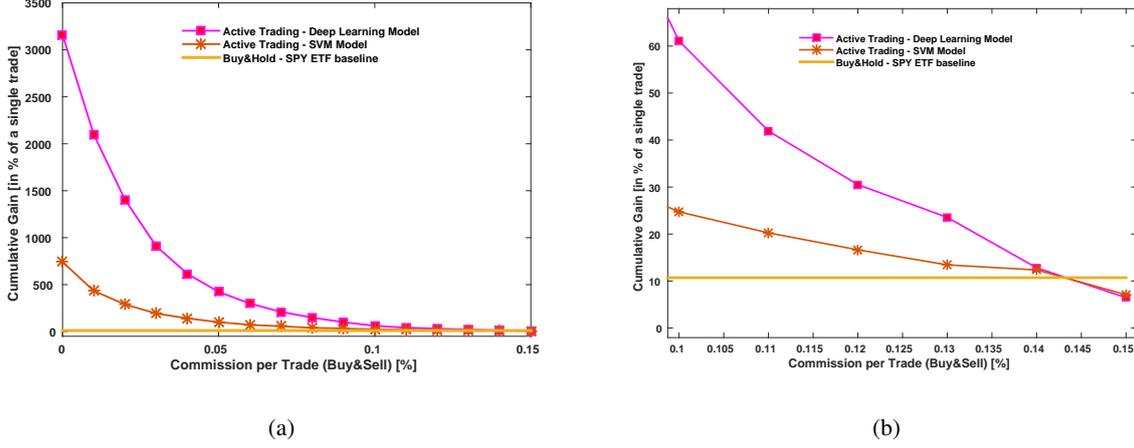


Figure 7: Overall cumulative gain over two years test period as a function of the transaction cost. We report the results of the proposed probabilistic active trading strategy for both the Deep-Learning model and the SVM model., as well as the buy-and-hold investing strategy are also presented. (a) A overall range of commissions. (b) The high commissions range.

and the region containing 99.3% of the samples.

The volatility of a financial asset is an important measure of its risk [33]. Table 2 and Fig. 10 report the cumulative two-years returns of the DL approach for the ten instruments, and the corresponding volatility. The volatility is computed as the annualized historical volatility, given by

$$\sigma_{Ann} = \sqrt{N_{minute}} \cdot STD(\mathbf{g}_{TC}) \quad 5.1$$

where N_{minute} is the number of trading minutes per year and \mathbf{g}_{TC} is the minute-level returns vector.

We also report the baseline instrument price. It follows that the performance of the proposed active trading scheme is correlated with the assets volatility, as exemplified in Fig. 10.

Figure 11 shows the annual Sharpe Ratio [34] versus the commissions for different assets using the proposed approach. As the Sharpe Ratio [35] quantifies the risk-adjusted return, it follows that it is strongly correlated with the commissions for all assets. The long term Sharpe ratio of the S&P 500 index that is commonly taken as 0.406, can be used as a reference. Thus, for commissions of 0.07% and lower, all assets achieve Sharpe ratios above 0.5, and for commission of 0.1%, three assets (T, JNJ and KO) underperform. It should be noted that these three instruments have the lowest volatility among all tested assets.

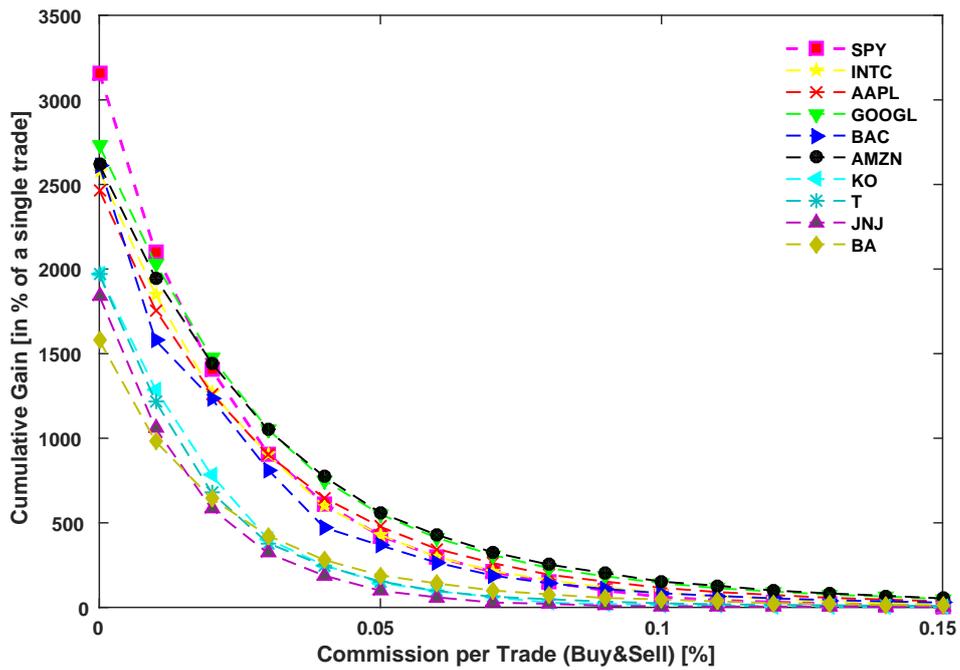


Figure 8: Cumulative trading gain over two years of back-testing as a function of the transaction cost (unified cost for buy&sell). Each graph represents a different instrument.

	Active-Trade	Baseline	Ann. Volatility
SPY	61.1	10.6	14.0
INTC	87.7	14.1	24.6
AAPL	115.4	24.3	23.5
GOOGL	144.0	35.1	25.7
BAC	83.6	-11.1	26.4
AMZN	152.2	86.1	34.2
KO	16.3	18.5	15.7
T	24.2	22.4	15.6
JNJ	7.4	9.0	15.3
BA	47.4	4.4	22.2

Table 2: Cumulative gain of the proposed active-trading strategy during two-year test period (0.1% commission), versus baseline instrument movement and the annualized historical volatility.

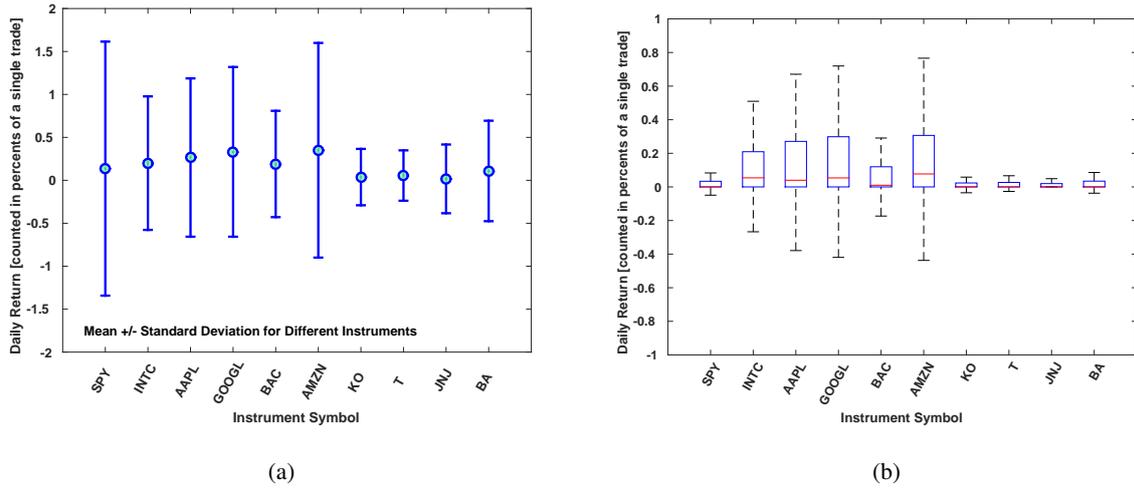


Figure 9: Daily return of the proposed Deep-Learning active trading strategy on different instruments, and a commission of 0.1%. (a) Mean standard deviation of the daily returns of the different instruments. (b) the median, and 25% and 75% percentiles.

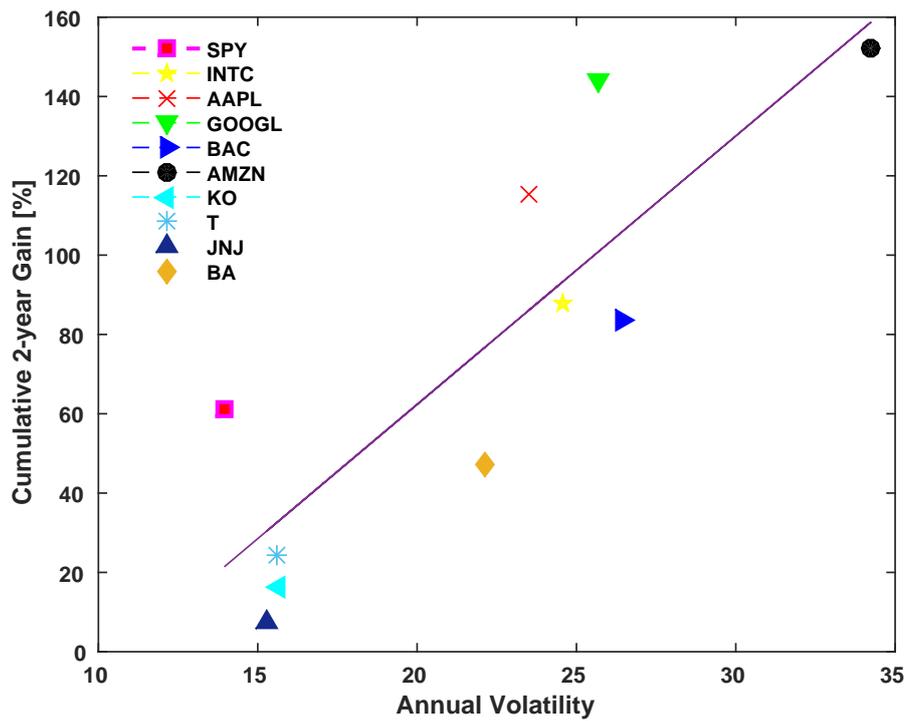


Figure 10: Linear regression relation between the cumulative gain achieved by an active-trading with different instruments as a function of the instruments volatility.

6 Conclusions and future work

In this work we presented a Deep Learning approach for financial time series forecasting. For that, we propose an end-to-end Deep-Learning forecasting model based on raw financial data inputs, contrary to

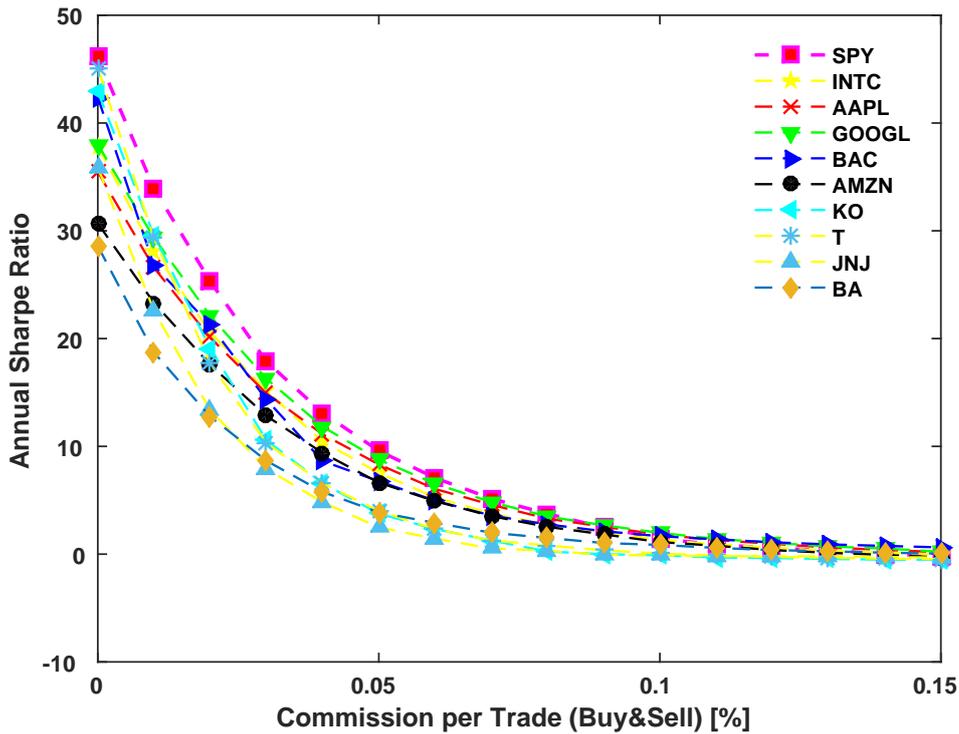


Figure 11: Annual Sharpe ratio as function of trade commissions for different instruments.

common statistical approaches to financial time series analysis which are based on engineered features. Our approach is shown to produce statistically significant forecasts, and we also derives a probabilistic active trading scheme achieving profitability in a realistic commission charged trading environment. This trading strategy utilizes the soft outputs of the DL to indicate the prediction validity, and is shown to outperform both active-trading based on other machine-learning algorithms, and the baseline buy-and-hold investing strategy.

Future work should include unifying the predicting model and the trading strategy. The model could be dynamically updated based on the whole historical data available at each point of time. These procedures are expected to produce further improvement in the forecasting accuracy. Additional improvement could be achieved by additional information from other sources, such as streaming media reports. In particular, following Markowitz’s portfolio theory, optimizing a portfolio of assets (ETFs, stocks, etc.) should yield improved results, in terms of lower volatility, and higher profitability.

References

- [1] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.
- [2] James Durbin. Efficient estimation of parameters in moving-average models. *Biometrika*, 46(3/4):306–316, 1959.
- [3] George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.
- [4] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.
- [5] Ruey S Tsay. *Analysis of financial time series*, volume 543. John Wiley & Sons, 2005.
- [6] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010.
- [7] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. In *Business Intelligence*, pages 62–77. Springer, 2013.
- [8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [9] Matthew Francis Dixon, Diego Klabjan, and Jin Hoon Bang. Classification-based financial markets prediction using deep neural networks. 2016.
- [10] Angelos Kanas and Andreas Yannopoulos. Comparing linear and nonlinear forecasts for stock returns. *International Review of Economics & Finance*, 10(4):383–398, 2001.
- [11] Tao Ban, Ruibin Zhang, Shaoning Pang, Abdolhossein Sarrafzadeh, and Daisuke Inoue. Referential knn regression for financial time series forecasting. In *International Conference on Neural Information Processing*, pages 601–608. Springer, 2013.
- [12] Md Rafiul Hassan and Baikunth Nath. Stock market forecasting using hidden markov model: a new approach. In *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pages 192–196. IEEE, 2005.

- [13] Robert K Lai, Chin-Yuan Fan, Wei-Hsiu Huang, and Pei-Chann Chang. Evolving and clustering fuzzy decision tree for financial time series data forecasting. *Expert Systems with Applications*, 36(2):3761–3773, 2009.
- [14] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.
- [15] Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307–319, 2003.
- [16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [17] 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.
- [18] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.
- [19] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772, 2014.
- [20] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- [21] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of CONLL*, 2016.
- [22] Ehud Ben-Reuven and Jacob Goldberger. A semisupervised approach for language identification based on ladder networks. *arXiv preprint arXiv:1604.00317*, 2016.
- [23] Alex Graves. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer, 2012.
- [24] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in Bioinformatics*, page bbw068, 2016.

- [25] Samuel Rönnqvist and Peter Sarlin. Bank distress in the news: Describing events through deep learning. *arXiv preprint arXiv:1603.05670*, 2016.
- [26] Ralph Fehrer and Stefan Feuerriegel. Improving decision analytics with deep learning: The case of financial disclosures. *arXiv preprint arXiv:1508.01993*, 2015.
- [27] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *IJCAI*, pages 2327–2333, 2015.
- [28] JB Heaton, NG Polson, and JH Witte. Deep portfolio theory. *arXiv preprint arXiv:1605.07230*, 2016.
- [29] Ruoxuan Xiong, Eric P Nichols, and Yuan Shen. Deep learning stock volatility with google domestic trends. *arXiv preprint arXiv:1512.04916*, 2015.
- [30] 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.
- [31] *QuantQuote - Expert Market Data Solutions*. <https://quantquote.com/historical-stock-data>.
- [32] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.
- [33] Jean Folger. [Investopedia.com/university](http://investopedia.com/university). Mar 2017.
- [34] Andrew W Lo. The statistics of sharpe ratios. *Financial analysts journal*, pages 36–52, 2002.
- [35] William F Sharpe. The sharpe ratio. *The journal of portfolio management*, 21(1):49–58, 1994.