Noname manuscript No.
(will be inserted by the editor)

Transferring Trading Strategy Knowledge to Deep Learning Models

Avraam Tsantekidis · Anastasios Tefas

the date of receipt and acceptance should be inserted later

Abstract Trading Strategies are constantly being employed in the financial markets in order to increase consistency, reduce human errors of judgement and boost the probability of taking profitable market positions. In this work we attempt to transfer the knowledge of several different types of trading strategies to Deep Learning models. The trading strategies are applied on price data of Foreign Exchange (FOREX) trading pairs and are actual strategies used in production trading environments. Along with our approach to transfer the strategy knowledge, we introduce a preprocessing method of the original price candles making it suitable for use with Neural Networks. Our results suggest that the Deep Models that are tested perform better than simpler models and they can accurately learn a variety trading strategy.

Keywords Trading Strategy · LSTM · RNN · Deep Learning

1 Introduction

Financial Exchanges are considered to be all the licensed hubs where financial institutes, investors and other entities submit their demands to buy or sell financial assets and speculate on their future values, among other financial activities. The most important function of financial markets is to efficiently allocate capital to businesses so they can expand their activities. Investors that place their investment in successful companies can have those investments increase in value as the stock price of said companies rise. The price of an asset is determined by the price investors are willing to pay for it at the time. Since this way of pricing can be volatile, the price of assets usually fluctuate through time.

Another activity happening in the Financial Markets is the speculation of the true price of assets. Seasoned investors can, at their own discretion, take advantage of price fluctuations to profit. Financial firms such as hedge funds employ discretionary traders who use quantitative and qualitative analysis of price movements to decide where to invest capital. Humans undertaking such tasks, have the disadvantage that an individual's sentiment may greatly affect their judgement and thus the efficacy and consistency of their trading behaviour.

Even though humans are not always suitable for the task, the quantitative methods they use have met great success in the past with firms having considerable profits from utilizing them [10, 15]. Due to the success of quantitative methods such as technical analysis and since they are relatively formulaic many of them were implemented as automated computer programs. The trading algorithms that emerged from this process were more consistent, a great deal faster than humans and their judgement was always the same, independent of any factors other than the quantitative measurements of the traded asset.

Another method of making investment decisions that has gained a lot of attention is utilizing machine learning. The existing literature on the subject is mainly focused on the discovery of preexisting patterns in time-series which can be taken advantage of for profit. One method for utilizing machine learning methods in trading is the direct regression of the price time-series of an asset. Approaches such as [18],

Avraam Tsantekidis, Anastasios Tefas
Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
E-mail: {avraamt,tefas}@csd.auth.gr

attempt to directly regress the future price or cashflows of a company that can then be used to derive trading decisions. Other types of regression has also been explored in the form of prediction of known patterns that arise from the price movements [17, 5, 1, 32].

In [19], an ensemble of machine learning methods, including Neural Networks, is used to predict exchange rate fluctuations. The task is framed as a classification problem for each of the different machine learning models, trained to predict the direction of multiple Foreign Exchange (FOREX) currencies. The prediction of each model is augmented by the correlations across currency pairs. Finally the signals are aggregated to form the final prediction that are used to modify a FOREX portfolio position.

The aformentioned work [19] and other similar research [28, 26] attempt to predict the price direction in a supervised setting by training their models to predict a handcrafted set of best-case targets that can be profitable if predicted correctly. This is often not the case since the labels are constructed from observing the future behaviour of the price time-series, which might not be correlated with the available information provided as input to the models trying to learn them.

One approach to avoid creating handcrafted labels is the use of Reinforcement Learning. Many of the latest successes of Deep Learning applications such as [13, 22] utilize Deep Reinforcement Learning (DRL), where agents can perform decision making in simulated environments such as video and board games. DRL techniques have managed to train agents to the point that their performance can surpass human agents participating in the same environments.

In [2], DRL is applied to financial trading, with an agent that is trained to automatically perform trading decisions on several types of financial products. This approach directly encodes the Profit and Loss (PNL) to a reward function that is used to train the agent. Thus the agent develops its own trading strategy through training in order to achieve profit, without any straightforward method to directly steer the decisions of the agents towards a specific behaviour.

In this work, we attempt to transfer the knowledge of real trading strategies used in production trading to a Deep Learning model. Our method enables training Deep Learning models to replicate trading signals emitted without utilizing any prior knowledge about the source of the signals or the details of their calculation. This would allow replicating trading signals from sources that are not always available such as successful human traders who can't work through the night or may retire. Another aspect of trading signals is that they are crisp decisions of buying or selling an asset. On the other hand if a Deep Learning model such as a Neural Network learns to replicate the trading signals of an unknown source it can provide a probability distribution across the decision choices. This allows the trading signals from the Neural Network model to be included in fuzzy decisions processes. To the best of our knowledge this is the first work that attempts to transfer the knowledge of real algorithmic trading strategies to Neural Network models.

The outline of the paper is as follows. In Section 2, technical analysis is explained and examples of technical indicators are given with ways to apply them for generating trading signals. In Section 3, we present the methods used to generate our training data and the models that were compared. In Section 4, the experimental setup is described and the results are discussed. Finally, in Section 5, conclusions are drawn and future work is suggested.

2 Trading Signals

The activity taking place on the financial markets consists of many participants that utilize information to take actions on the markets. This information may consist on qualitative information, such as the sentiment of the news articles surrounding a company, or rely on quantitative attributes of an asset's price movements [11]. Utilizing data produced by the trading activity taking place in the financial markets is based on the hypothesis that such data contains quantifiable events and patterns [7, 30], that can be taken advantage of when making investment decisions.

The agents who try to determine such patterns and take executive decisions when investing their assets have a wide range of tools at their disposal. Sometimes the detection of aforementioned patterns is done by personal intuition from people who have spent many years examining the markets, who claim that experience alone is enough, while others utilize technical analysis of asset price charts to decide their positions.

All participants in the trading activity have their own methods to parse and digest the information of the financial markets and decide on their investment strategies. The resulting decision of a trading

strategy, whether it comes from purely human-made decision or by strict algorithmic processing, can be considered as one or multiple trading signals to buy or sell a specific asset. More formally, a trading signal y_t is an indication given by a trading strategy g at time t to either buy or sell a specific asset.

2.1 Technical Analysis Methods

In this work, quantitative methods are used as the source of learned trading signals, but the presented methods can be applied on any source of signals, as long as the information that was taken into consideration to produce them is contained within the provided input to the model.

Technical Analysis (TA) has been used for many decades as tool for generating trading signals with varying levels of success. Many authors criticise the use of TA as it violates the strong version of the efficient market hypothesis [12], while others advise it should be taken into consideration in conjunction with external market knowledge [24].

The popularity of TA, has given rise to cases of "herding" [4] where the majority of investors seeking short-term profit rely on the TA signals [24], causing significant volatility on traded assets [3, 21, 20]. Although TA might not seem a valid method of taking decisions by people strictly applying fundamental analysis, the fact that so many people use it to inform their trading decisions ends up impacting the markets. The resulting price movements may end up confirming the original TA predictions as a "self-fulfilling prophecy".

Trading strategies derived from TA, usually consist of a rule based system applied on technical indicators. Technical indicators are the various filters applied on market data, that are then used to carry out some form of technical analysis. The two most common technical indicators are:

2.1.1 Simple Moving Average (SMA)

One of the most elementary technical indicators is the Simple Moving Average [14]. It is calculated by averaging the price of the past price samples. The price samples used are usually the close prices of the Open-High-Low-Close (OHLC) tick subsampling technique, which is used to produce the OHLC candles. The number of past price samples averaged is a parameter that can be changed.

$$\begin{aligned} \mathrm{SMA}_n(t) &= \frac{\sum_{i=t}^{t-n+1} p_i}{n} \\ &= \frac{(p_t + p_{t-1} + \cdots + p_{t-n+1})}{n} \end{aligned} \tag{1}$$

where p_t is the subsampled close price at time t and n is the window size of the calculated SMA.

2.1.2 Relative Strength Index (RSI)

The Relative Strength Index is typically referred to as an oscillator for determining if assets are "over-bought" or "over-sold" [14]. In essence it measures the relative upwards to downwards movements of an asset's price in the value range of 0 to 100.

$$U(t) = \begin{cases} p_t - p_{t-1}, & \text{if } p_t > p_{t-1} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$D(t) = \begin{cases} p_{t-1} - p_t, & \text{if } p_t < p_{t-1} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $U(t)$ and $D(t)$ are the conditional return measures for each time-step t. To calculate the RSI, the RS measure is first calculated,

$$\mathrm{RS}_n(t) = \frac{\sum_{i=t}^{t-n+1} U(i)}{\sum_{i=t}^{t-n+1} D(i)} \tag{4}$$

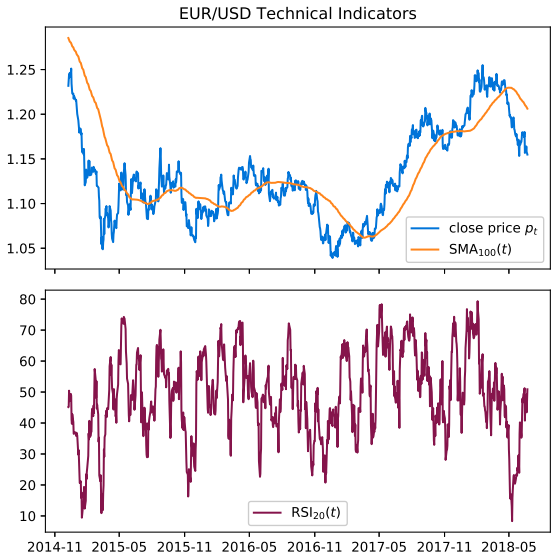


Fig. 1: Example of technical indicators SMA_{100} and RSI_{20} along the close price of the EUR/USD FOREX pair.

which follows the windowed average logic similar to the SMA calculation. Finally the RS measure is bounded to a 0-100 value resulting in the Relative Strength Index (RSI).

$$\text{RSI}_n(t) = 100 - \frac{100}{1 + \text{RS}_n(t)} \tag{5}$$

2.2 Trading Techniques

A rough categorisation of the trading "styles" that certain strategies exhibit are as follows:

1. **Trend Following** which attempts, by using a momentum measure, to take action when a trend is detected and follow it until it concludes, thus gaining a profit while the trend lasts.
2. **Mean Reversion** which attempts to trade with the short-term logic that some asset prices do not diverge far from their mean, thus trying to achieve multiple small gains by from the price movements around that mean.

Different assets call for different styles of strategies, for instance stocks usually exhibit trend patterns which means that trend following strategies may be more suitable, while Foreign Exchange currencies tend to have a behaviour better suited for mean reversion strategies. Although there are many other types of trading using technical analysis, (namely breakout trading, retracement, etc.) their definition and explanation are not related to the context of this work and the above types are only presented as a simple example to help the reader understand how trading signals can be generated from such strategies.

The simplest form of trading strategy using indicators such as the SMA and RSI is a rule based approach of setting specific "trigger" ranges for each of their values. When an indicator moves into such a "trigger" range a specific trading signal is emitted. This can be either automatically fed into a trading engine to execute the emitted signal or to simply alert a human trader about it and suggest an action.

For example, a very common signal for a trend following strategy is the 200-day SMA, where the signal to buy is when the price surpasses the last SMA value and the signal to sell is when the price falls below the last SMA value [14]. The range parameter of the SMA that is chosen to be 200 is not based on strongly supported research but only because it's a commonly used example.

On the opposite side of the spectrum, a mean reverting method using RSI is to sell an asset when the RSI surpasses 70 which is supposed to mean that the asset is "overbought" and to buy an asset when the RSI dips below 30 which means the asset is "oversold". Again the specific ranges for buying and

selling are commonly used suggestion and, to the best of our knowledge, are not strongly supported by evidence.

2.3 Strategy Signal Generation

In this work we attempt to transfer the price parsing capabilities of complex trading strategies that generate trading signals, to a machine learning model. Consider an input price time-series $\mathbf{x} = (x_1, x_2, \vec{\dots}, x_n)$ of length n that describes the price of a financial asset as it is traded in the markets. Let g be an unknown signal generator that given a price time-series $\mathbf{x}$ produces a signal y_n.

$$g(\mathbf{x}) = y_n \in \{-1, 0, 1\} \tag{6}$$

The output label {-1, 0, 1} represents the 3 possible outputs of the generator which stand for different signals. When the generator emits -1 it refers to shorting or selling an asset, while 1 refers to buying an asset. When 0 is emitted, then no trade action is advised.

The unknown signal generator might be an algorithmic trading strategy, or a human deciding what assets to trade. The signal y_n is the trading signal to be executed after the n-th step provided in the time-series $\mathbf{x}$ of size n. The signal generator can produce signals for the whole time-series $\mathbf{x}$ by applying it on a chronologically expanding subset of the time-series starting from the earliest time-step.

$$y_i = g((x_1, x_2, \vec{\dots}, x_i)),\ \forall i \in \{2, 3, \dots, n\} \tag{7}$$

Thus we can annotate all the signals produced by generator g as the vector $\mathbf{y} = (y_2, y_3, \vec{\dots}, y_n)$. For posterity we include an extra signal y_1 to $\mathbf{y}$ so that the vectors $\mathbf{y}$ and $\mathbf{x}$ have aligned indices in a chronologically consistent manner.

To approximate the signals produced by the generator, a probabilistic machine learning model f is trained so that:

$$f(\mathbf{x}) = P(g(\mathbf{x})|\mathbf{x}) \tag{8}$$

Other than strictly algorithmically defined trading strategies, the potential uses of this transfer of knowledge may be applied to other sources of trading signals. One such source might be human traders that base their trading actions on their own instinct in addition to chartist techniques.

In this work, the DL techniques of Long-Short Term Memory Recurrent Neural Networks (LSTM) are applied to learn a generator of trading signal. The trading signals employed were sourced from actual trading strategies applied in the currency exchange markets by SpeedLab A.G, which is the affiliated financial management company in this work.

3 Strategy Knowledge Transfer

In this Section a proposed set of stationary time-series features are described. These features greatly improve the performance of Deep Learning models as it will be shown in Section 4. The proposed LSTM model architecture is presented along with the deep learning techniques that are applied. A Convolutional Neural Network (CNN) model architecture is also presented and its performance results are compared to the LSTM model.

3.1 Features

First the data used to monitor the markets by chartists and technical analysis experts are defined. The main method to parse all the trading activity into presentable price movements is the Open-High-Low-Close subsampling method [16]. This method defines time-windows that sample 4 trade prices for all the trades they encompass,

1. Open Price $p_o(t)$: the price of the first trade in the window.
2. High Price $p_h(t)$: the highest price a trade was executed in the window
3. Low Price $p_l(t)$: the lowest price a trade was executed in the window

4. Close Price $p_c(t)$: the price of the last trade in the window

Each window captures trades that have timestamps in the range $(t - w + 1, t]$, where t is a moment in time and w is the size of each window. Variables $p_o(t), p_h(t), p_l(t)$ and $p_c(t)$ denote the subsampled OHLC price of the window that ends at time t.

Machine learning models such as Artificial Neural Networks benefit when their inputs have similar distributions to their activation function and when the input statistics are stationary. In the case of raw price data neither is true so a a normalization scheme is required to remedy any potential problems. The proposed features consist of the following deltas:

1. $p_c(t) - p_c(t-1)$
2. $p_h(t) - p_h(t-1)$
3. $p_l(t) - p_l(t-1)$
4. $\log(p_h(t)) - \log(p_c(t))$
5. $\log(p_c(t)) - \log(p_l(t))$
6. $\log(p_c(t)) - \log(p_c(t-1))$

The delta $p_c(t) - p_c(t-1)$ is usually referred to as the return. The reason for choosing this set of features is their stationary nature. Also many technical indicators utilize these kind of features internally such the deltas between prices. For this work we will not examine the merit of each of these features individually or whether additional features will improve performance, but focus on the ability of Deep Learning models to learn the process a signal generator uses for producing its emitted signals.

The features introduced are normalized using standard normalization (standardization), which means to shift the mean and standard deviation of the data to 0 and 1 respectively. In this case we only apply the standard deviation shifting since our features are price deltas and shifting their mean would distort the implied nature of zero valued price deltas. Also to avoid any problematic outliers before normalizing we clip the values outside the 0.1 and 99.9 percentiles of each features' probability distribution.

3.2 LSTM Model

A Recurrent Neural Network (RNN) is a type of Artificial Neural Network that exploits the concept of time in the data by employing an internal memory module to keep track of events as they happen and attempt to associate relevant parts of past inputs to future inputs. Because RNNs suffered of the problems of vanishing gradients a popular solution was developed called the Long-Short Term Memory RNNs that implemented a gating mechanism to protect the internal memory representations of the RNN [8] from the decay due to unrelated inputs and gradients. The protected hidden activation is the "cell state" which is regulated by said gates in the following manner:

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf} \cdot \mathbf{x} + \mathbf{W}_{hf} \cdot \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{9}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi} \cdot \mathbf{x} + \mathbf{W}_{hi} \cdot \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{10}$$
$$\mathbf{c}'_t = tanh(\mathbf{W}_{hc} \cdot \mathbf{h}_{t-1} + \mathbf{W}_{xc} \cdot \mathbf{x}_t + \mathbf{b}_c) \tag{11}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}'_t \tag{12}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{oc} \cdot \mathbf{c}_t + \mathbf{W}_{oh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{13}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \sigma(\mathbf{c}_t) \tag{14}$$

where $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ are the activations of the input, forget and output gates at time-step t. These gates control how much of the input and the previous state will be considered and how much of the cell state will be included in the hidden activation of the network. The notation $\odot$ denotes the element-wise multiplication between vectors. The protected cell activation at time-step t is denoted by $\mathbf{c}_t$, whereas $\mathbf{h}_t$ is the activation that will be given to other components of the model. The matrices $\mathbf{W}_{xf}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{xi}$, $\mathbf{W}_{hi}$, $\mathbf{W}_{hc}$, $\mathbf{W}_{xc}$, $\mathbf{W}_{oc}$, $\mathbf{W}_{oh}$ are used to denote the weights connecting each of the activations with the current time-step inputs and the previous time-step activations.

The LSTM model layer structure is sequential and consists of two LSTM layers and two Fully Connected (FC) layers. The full model specification is described in Table 1. Batch Norm layers refer to the Batch Normalization layer that were introduced in [9] helping reduce internal covariate shift during training, by ensuring that the batch statistics of each layer stay the same throughout training. Dropout

Table 1: LSTM model specification

Layer Type	Layer Specification	Activation
Input	600×6	-
LSTM	6×128	Tanh
Batch Norm	128	-
Dropout	40%	-
LSTM	128×128	Tanh
Batch Norm	128	-
Dropout	40%	-
Fully Connected	32	PReLu
Fully Connected	3	Softmax

Table 2: CNN model specification. The Convolutional layers specification describe the filter size $\times$ number of filters $\rightarrow$ causal time-series output

Layer Type	Layer Specification	Activation
Input	600×6	-
Convolutional	$16 \times 32 \rightarrow 584$	PReLU
Batch Norm	32×584	-
Convolutional	$16 \times 16 \rightarrow 568$	PReLU
Batch Norm	16×568	-
Dropout	40%	-
Convolutional	$16 \times 16 \rightarrow 552$	PReLU
Batch Norm	16×552	-
Convolutional	$16 \times 8 \rightarrow 536$	PReLU
Batch Norm	8×536	-
Dropout	40%	-
Fully Connected	32	PReLu
Fully Connected	3	Softmax

is referring to the technique of randomly deactivating a percentage of activations, reduces neuron co-adapation, thus slowing down overfitting [23]. Finally PReLu activation refers to the modified version of Rectified Linear Unit that is introduced in [6].

Since our model is an LSTM and should receive time-series as input we use windows of size 300 time-steps with the 6 features described in Section 3.1 of each time-step. According to previous work with LSTM [27] we avoid training for the first 100 time-steps since the recurrent network benefits from first having a "burn-in" sequence before being forced to produce predictions. Consequently the model is trained for the last 200 time-steps of each window.

The last 2 FC layers described in Table 1 are applied in a time distributed manner so that they separately parse the output that the final LSTM layer emits at every time-step. To train the LSTM network back-propagation through time [31] is employed to efficiently calculate the gradient value of each parameter. The gradient values are supplied as inputs to the RMSProp algorithm [25] which decide the training step of each parameter in the model that is being trained.

3.3 CNN Model

To compare our proposed approach with other Deep Learning methods, we also employ a Convolutional Neural Network with causal padding to predict the same labels y as our LSTM model. Causal padding is a proposed change to the padding applied to CNNs by [29] to align chronologically the inputs to the outputs of a CNN model. Batch normalization and Dropout is also applied in the CNN model and we use a total of 4 layers. The exact specification of the CNN model is described in Table 2.

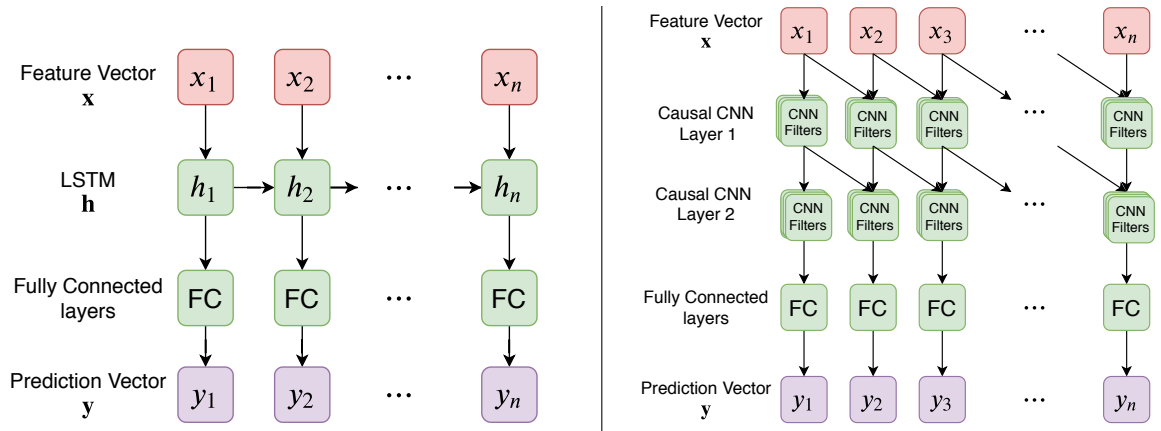
Since we apply causal convolutions the activations are chronologically aligned with the input and output. This allows for applying the final 2 FC layers of the CNN model similarly to the LSTM model's FC layers in a time distributed manner as described in Section 4.3. Multiple CNN variants were tested by varying the parameters of filter sizes and number of filters.

The training algorithm optimizer that was used, similarly to the LSTM, was RMSprop with learning rate 1×10^{-3} and a scheduled decay is put in place that reduces the learning rate by 50% every 100 epochs.

3.4 Time-Series Processing

The two proposed models are capable of processing time-series in a slightly different way from each other. For each model, a basic description is given to better present the details of their operation on time-series.

Initially the features described in Section 3.1 are extracted for each of the price time-series, resulting in the feature vector $\mathbf{x} = (x_1, x_2, \vec{\dots}, x_n)$. For each feature, several time-series "chunks" or windows are extracted to be used as samples during the stochastic optimization. Each extracted sample has a total of 300 steps with 6 feature values on each step. The 6 feature values are the ones described in Section 3.1. Similarly to the features, we apply the algorithmic strategies' logic to produce a label for each time-step resulting in label vector $\mathbf{y} = (y_2, y_3, \vec{\dots}, y_n)$.



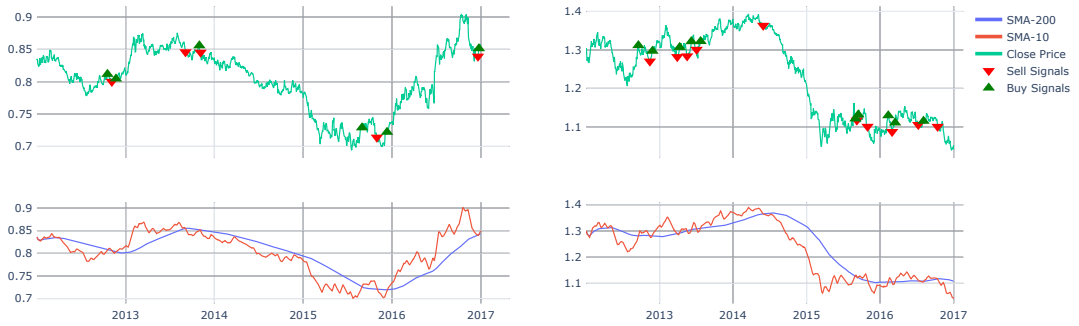
(a) LSTM (b) Causal CNN

Fig. 2: Time-series processing by the proposed models. Each model is given the feature time-series $\mathbf{x}$ consisting of the 6 features described in Section 3.1 and processes it to produce the prediction of the strategy's label at each time-step. The Causal CNN is simplified with a shorter receptive field and less layers, to avoid the clutter of illustrating numerous connections.

The LSTM model processes each time-step in a sample window sequentially. For every step the LSTM takes, the FC layers receive the values of the LSTM hidden state and produce a prediction, as show in Figure 2a. The prediction produced corresponds to the strategy label of that step.

The Causal CNN operates on multiple inputs at once combining past with present information into each of the convolutional filters. As the input is propagated through the convolutional layers the field the layers receive information from further in the past as is show in Figure 2b. In our experiments we use four convolutional layers as shown in Table 2 and the causal convolution layers as shown in Table 2 have a wide receptive field of 16 total inputs being observable.

Both the LSTM and CNN models process the time-series information and extract some useful representation from them. This representation is then given to a set of FC layers which produce a prediction using a softmax output layer with three output neurones representing the three possible classes. The available classes are the same as those the algorithmic strategies compute which are shown in Section 2.3. Thus the prediction of each of the proposed models can be directly compared to the labels the algorithmic strategies assigned to each step and also be used to train said models using Stochastic Optimization.



(a) EURGBP (b) EURUSD

Fig. 3: A Simple Moving Average crossover signals example. The top figures shows the labels produced by the strategy overlaid on top of the close price of each 24 hour interval, while the bottom figures are the two SMA values. Whenever the two SMA values on the bottom cross a buy or sell label is generated. Moving averages used are a 200-step moving average and a 10-step moving average.

4 Experiments

In this section a description of the replicated strategies is given along with a data augmentation technique. Next the training parameters are presented and the resulting performance of the compared models is discussed.

4.1 Introductory Example

In order better explain our approach we present an example scenario of learning a widely accessible open-source trading strategy. The strategy chosen follows the instructions we first present in Section 2.2, namely the trend following strategy that triggers on a simple SMA crossing rule.

Given two SMA of the closing price time-series, one with a longer length window (e.g. 200 days) and one SMA with a shorter window (e.g 10 days), we generate a buy signal whenever the shorter SMA value crosses upwards the longer SMA value at time-step t. Conversely a sell signal is generated when the shorter SMA value crosses the longer SMA downwards.

$$\text{Trend Signal}(t) = \begin{cases} -1, & \text{if } \text{SMA}_{10}(t) \leq \text{SMA}_{200}(t) \text{ and } \text{SMA}_{10}(t-1) > \text{SMA}_{200}(t-1) \\ 1, & \text{if } \text{SMA}_{10}(t) \geq \text{SMA}_{200}(t) \text{ and } \text{SMA}_{10}(t-1) < \text{SMA}_{200}(t-1) \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

,where $\text{SMA}_{10}(t)$ is the SMA of the close price with window size 10 at time-step t and $\text{SMA}_{200}(t)$ is the the SMA with window size 200.

The resulting decisions of the strategy are illustrated in Figure 3 for currency pair EUR/USD and EUR/GBP. This example strategy presented here is not making consistently profitable decisionstime-step. The actual strategies examined in the next section for which the exact algorithm is not disclosed, have a more complex set of signals and combine them to improve the deficiencies of this example strategy.

With this simple signal generator we can create labels for all the available price data which consist of 28 foreign exchange currency pairs. Each time-step in the price time-series receives a label from this sample trend strategy. Since the possible choices for each time-step are three, we can represent that using three softmax output neurons on the neural network model of our choice.

Having generated all the labels for this strategy, we can then generate all the features described in Section 3.1. Using the model described 1 with a single LSTM layer instead of two, the example crossover strategy with parameters of 20 and 5 steps of moving averages is learned. Shorter averaging windows are used to ensure enough signals are generated for the model to be properly trained. In the next section we present a method to augment the data, so that enough samples, to train the deep models, are generated. The performance of the trained LSTM on the described example strategy is shown in Table 3

Table 3: Performance of single layer LSTM trained on example SMA crossover strategy with step sizes 20 and 5.

Train				Test			
Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
0.998	0.992	0.99	0.991	0.994	0.972	0.974	0.973

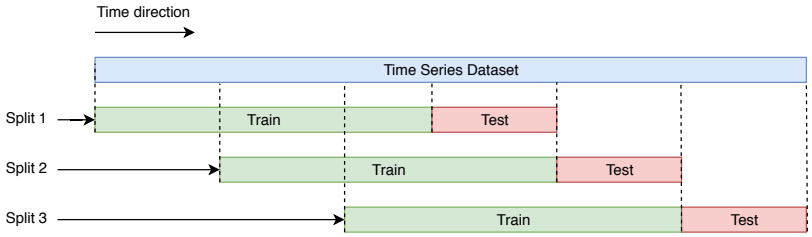
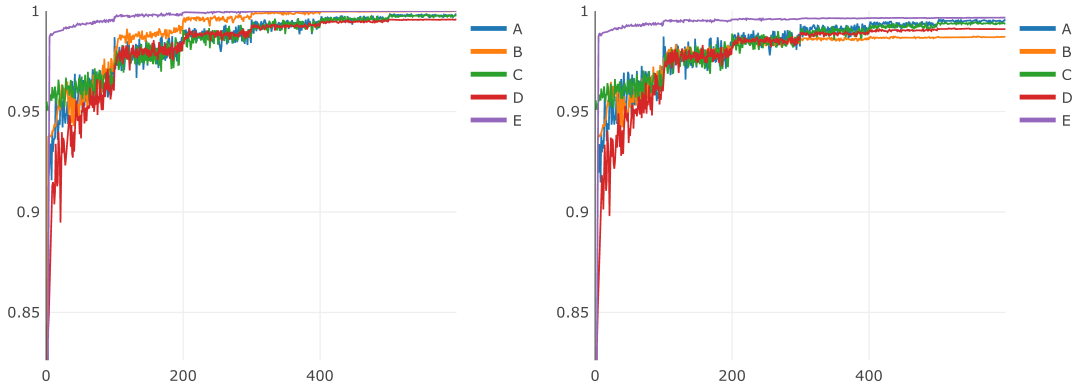


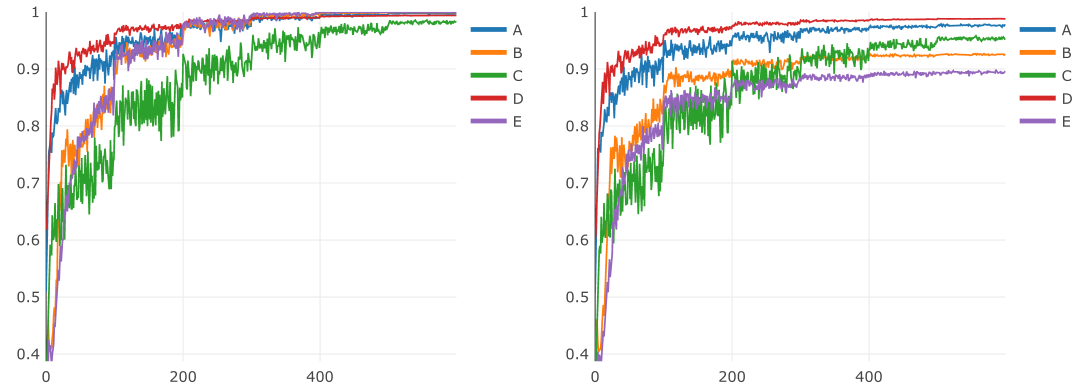
Fig. 4: Walk Forward cross validation.



(a) Train accuracy

(b) Test accuracy

Fig. 5: Mean accuracy metrics across validation splits during training. Each line corresponds to each strategy as described in Section 4.2. (Best viewed in colour)



(a) Train F1

(b) Test F1

Fig. 6: Mean F1 metrics across validation splits during training. Each line corresponds to each strategy as described in Section 4.2. (Best viewed in colour)

Table 4: Mean performance metrics for each strategy of the model after training. Input features with the label Proposed denote that the proposed features of Section 3.1 were used while the label OHLC denote that raw OHLC values were used.

Strategy	Input Features	Model	Train				Test			
			Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
A	Proposed	LSTM	**0.998**	**0.997**	**0.998**	**0.997**	**0.995**	**0.977**	**0.976**	**0.976**
		CNN	0.994	0.988	0.990	0.989	0.988	0.943	0.932	0.937
	OHLC	LSTM	0.989	0.967	0.965	0.966	0.989	0.962	0.955	0.958
		CNN	0.958	0.885	0.747	0.754	0.952	0.828	0.736	0.743
B	Proposed	LSTM	**1.000**	**0.999**	**0.999**	**0.999**	**0.987**	**0.924**	**0.928**	**0.926**
		CNN	0.977	0.901	0.827	0.859	0.968	0.844	0.767	0.799
	OHLC	LSTM	0.959	0.812	0.644	0.700	0.957	0.810	0.629	0.684
		CNN	0.941	0.350	0.335	0.326	0.941	0.346	0.334	0.325
C	Proposed	LSTM	**0.998**	**0.982**	**0.984**	**0.983**	**0.994**	**0.953**	**0.952**	**0.952**
		CNN	0.996	0.963	0.983	0.973	0.990	0.917	0.940	0.928
	OHLC	LSTM	0.985	0.902	0.860	0.878	0.983	0.885	0.846	0.864
		CNN	0.971	0.808	0.726	0.760	0.967	0.769	0.675	0.706
D	Proposed	LSTM	**0.996**	**0.993**	**0.995**	**0.994**	**0.991**	**0.988**	**0.988**	**0.988**
		CNN	0.987	0.984	0.983	0.983	0.981	0.976	0.973	0.974
	OHLC	LSTM	0.910	0.881	0.834	0.847	0.902	0.866	0.834	0.842
		CNN	0.856	0.830	0.767	0.781	0.852	0.847	0.747	0.764
E	Proposed	LSTM	**1.000**	**1.000**	**1.000**	**1.000**	**0.997**	**0.902**	**0.891**	**0.896**
		CNN	0.999	0.968	0.952	0.959	0.989	0.656	0.534	0.579
	OHLC	LSTM	0.994	0.855	0.726	0.776	0.992	0.789	0.664	0.713
		CNN	0.989	0.544	0.334	0.334	0.989	0.334	0.333	0.332

4.2 Source Strategies

The signal generators that we use to train the LSTM models originate from 5 individual strategies that have been used as part of a production trading system at SpeedLab AG. Due to the competitive nature of financial trading, we are unable to disclose the full definition of the strategies, but a brief description of each strategy is provided.

1. **Strategy A**: Trend following strategy that can be adjusted for long or short term trends and provides slow but deliberate signals.
2. **Strategy B**: Trend following strategy with trigger conditions of a breakout strategy.
3. **Strategy C**: Trend following strategy similar to strategy A but taking into consideration different factors.
4. **Strategy D**: Breakout strategy that monitors abnormal events to emit signals.
5. **Strategy E**: Mean Reversion strategy monitoring for lack of trend and emit signals during lateral market movements.

By training on a wide range of trading styles we ascertain that the learning results presented in this Section can be applied to a variety of signal generators.

When training Deep Learning models, an important issue is the need of adequate amounts of data. Unfortunately in our case, the parameterized strategies generate too few signals to have a reasonable training sample for our DL model for each FOREX pair data. To resolve this issue, the training data of each trading pair is augmented with an extra set of input features and labels. These supplemental features and labels are extracted from the price time-series of different FOREX pairs that were rescaled to fit the range of values of the originally trained FOREX pair. This means that if strategy A is being trained on FOREX pair EUR/USD, that has a range of price values $[1.02, 1.6]$, we rescale all other FOREX pairs' price time-series within that range and apply the signal generator we initially used for the EUR/USD pair. Then the resulting features and labels are added to the training set of the EUR/USD pair. The same process is applied to all the tested strategy-currency pairs that are evaluated.

The dataset consists of several FOREX trading pairs including AUD/CAD, AUD/CHF, EUR/USD, EUR/AUD, GBP/USD, GBP/JPY and USD/CAD. The price data $\mathbf{x}$ is supplied to the generator strategies that produce the signals $\mathbf{y}$ that will act as the labels our model will learn. Because of the differences in market behaviour between currency pairs, not all strategies are suitable for all trading pairs. In this work each strategy is used to generate trading signals for at least 3 FOREX pairs that would be considered suitable for their behaviour.

In total the data used consists of 28 FOREX trading pairs. For each pair the data available is from 2010 to 2018, with minute subsampling resolution totalling a bit over 80 million 1-minute OHLC candles. For each 1-minute candle, 6 features were extracted as shown in Section 3.1 and labels were generated by each strategy.

4.3 Training Specification

All the models are trained with RMSProp which is initialized with learning rate 1×10^{-3} and a scheduled decay is put in place that reduces the learning rate by 50% every 100 epochs. In Figures 5 and 6 there is a clear indication of the benefits of the learning rate decay that was employed every 100 epochs. Each training step uses a batch size of 128 samples, large enough so that the batch normalization layers can collect an accurate and consistent approximation of each layer's activation statistics. Each experiment is trained for 600 epochs.

To conduct cross validation we use the walk forward method, which consists of slicing the training and test sets in a time consistent manner. To achieve this we consider the training and test set selection as a sliding window on the chronologically sorted time-series. The training part of the selection is always before the test part as shown in Figure 4 to avoid any advantage from using information from the future.

A separate training experiment is conducted for each of the strategies and each of the selected FOREX pairs totalling $|\text{strategies}| \times |\text{FOREX pairs}| = 5 \times 3 = 15$ experiments for each model and type of input features. Each experiment consists of 3 walk forward splits as shown in Figure 4. We measure train and test performance metrics, such as accuracy, precision, recall and F1, at the end of each training epoch.

The proposed LSTM model is able to learn and emit the correct signals in almost every situation. The cases where the correct signal is not emitted are usually caused by samples lying extremely close to the decision boundaries of the model. Such cases are very few as shown in the performance metrics in Figures 5 and 6.

4.4 Performance Comparisons

As a comparison to the LSTM model architecture, a CNN model is also trained on the same data. The CNN model's performance metrics are included in Table 4. The CNN models presented manage to perform very similarly to the LSTM on the training data, but does not maintain the same performance on the test data. The results clearly indicate that across all strategies and input features, the LSTM model is able to generalise better that the CNN model.

An observation that was made is that the accuracy metric on most of the strategies is nearly perfect, even in the test dataset, whereas the rest of the metrics are not as close to perfect. This can be attributed to the fact that the labels produced by some of the strategies are highly unbalanced, since some of them are very selective when entering the market and emit too few buy or sell signals. The resulting bias might not affect accuracy since it does not consider the different population of each class, but it does affect the rest of our metrics which are averaged in an unweighted manner across classes. Still our proposed models manage to perform well, overcoming this peculiarity in our data.

This leads to the conclusion that the proposed LSTM model is better suited for the task of copying generated signals on such time-series. The CNN results presented in Table 4 are the best performing ones of the CNN architectures that were tested. The major problem of larger CNNs was that overtraining occurred even though both Batch Normalization and Dropout was used.

To validate the proposed features mentioned in Section 3.1, we compare the aforementioned CNN and LSTM models, using as input the raw Open-High-Low-Close (OHLC) values. The OHLC are normalized using standardization moving their mean and standard deviation to 0 and 1 respectively. The rest of the model and training parameters remain unchanged. The resulting performances are shown in Table 4. The proposed features perform better across all models and strategies, confirming that using this kind of features yields better performance compared to the raw OHLC values.

There also seems to be a discrepancy among the learning of each strategy by the proposed models. More specifically the metrics for learning Strategies B and E perform considerably worse that the rest. This can also be attributed to the fact that they are two of the strategies that are more conservative in entering the market and thus emitting less signals that can be used to train the proposed models.

These results confirm our hypothesis that an LSTM can become a near-identical signal emitter that copy some generator strategy. It is also shown that the LSTM models performs better than the CNN models for cloning a signal generator in a time-series setting. Finally the proposed features are shown to perform better for all models than the raw OHLC values for this task.

In many trading firms another source of information that is used is the published news articles, which many traders take into consideration before trading. A future direction of this work would be to transfer the trading signals emitted by a strategy that takes news into consideration, by also giving the same news as input to the machine learning model.

5 Conclusion

In this work the knowledge from production strategies that emit trading signals is successfully transfers to an LSTM model. Although in this instance the employed strategies can be algorithmically derived from the OHLC price time-series, the presented model can be applied in cases where that is not true, such as strategies directly generated by human traders. This can be useful for companies to ensure that a successful trader's behaviour can be simulated by such model to continue the same trading activity if they becomes unavailable. The proposed model is compared to another Deep Learning model, namely a Convolutional Neural Network (CNN) which it surpasses at all performance metrics. The proposed features' contribution to performance is tested by replacing them with the raw (but normalized) OHLC values of the price time-series, which also reveals that the proposed features improve performance significantly.

Acknowledgements

We kindly thank Speedlab AG for providing their expertise on the matter of FOREX trading and the comprehensive dataset of FOREX currency pairs. This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code: T2EDK-02094). Avraam Tsantekidis was solely funded by a scholarship from the State Scholarship Foundation (IKY) according to the "Strengthening Human Research Resources through the Pursuit of Doctoral Research" act, with resources from the "Human Resources Development, Education and Lifelong Learning 2014-2020" program.

References

1. Chen CH, Lu CY, Lin CB (2019) An intelligence approach for group stock portfolio optimization with a trading mechanism. Knowledge and Information Systems pp 1–30
2. Deng Y, Bao F, Kong Y, Ren Z, Dai Q (2017) Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems 28(3):653–664
3. Frankel JA, Froot KA (1990) Chartists, fundamentalists, and trading in the foreign exchange market. The American Economic Review 80(2):181–185
4. Froot KA, Scharfstein DS, Stein JC (1992) Herd on the street: Informational inefficiencies in a market with short-term speculation. The Journal of Finance 47(4):1461–1484
5. Goumatianos N, Christou IT, Lindgren P, Prasad R (2017) An algorithmic framework for frequent intraday pattern recognition and exploitation in forex market. Knowledge and Information Systems 53(3):767–804
6. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034
7. Hirshleifer D, Subrahmanyam A, Titman S (1994) Security analysis and trading patterns when some investors receive information before others. The Journal of Finance 49(5):1665–1698
8. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780

9. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167
10. Jegadeesh N, Titman S (1993) Returns to buying winners and selling losers: Implications for stock market efficiency. The Journal of finance 48(1):65–91
11. Krishnamoorthy S (2018) Sentiment analysis of financial news articles using performance indicators. Knowledge and Information Systems 56(2):373–394
12. Malkiel BG, Fama EF (1970) Efficient capital markets: A review of theory and empirical work. The journal of Finance 25(2):383–417
13. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529
14. Murphy JJ (1999) Technical analysis of the financial markets: A comprehensive guide to trading methods and applications. Penguin
15. Neely C, Weller P, Dittmar R (1997) Is technical analysis in the foreign exchange market profitable? a genetic programming approach. Journal of financial and Quantitative Analysis 32(4):405–426
16. Nison S (2001) Japanese candlestick charting techniques: a contemporary guide to the ancient investment techniques of the Far East. Penguin
17. Ozorhan MO, Toroslu IH, Sehitoglu OT (2018) Short-term trend prediction in financial time series data. Knowledge and Information Systems pp 1–33
18. Pai PF, Lin CS (2005) A hybrid arima and support vector machines model in stock price forecasting. Omega 33(6):497–505
19. Petropoulos A, Chatzis SP, Siakoulis V, Vlachogiannakis N (2017) A stacked generalization system for automated forex portfolio trading. Expert Systems with Applications 90:290–302
20. Shiller RJ (1987) Investor behavior in the october 1987 stock market crash: Survey evidence
21. Shiller RJ, Fischer S, Friedman BM (1984) Stock prices and social dynamics. Brookings papers on economic activity 1984(2):457–510
22. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, et al. (2016) Mastering the game of go with deep neural networks and tree search. nature 529(7587):484
23. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1):1929–1958
24. Taylor MP, Allen H (1992) The use of technical analysis in the foreign exchange market. Journal of international Money and Finance 11(3):304–314
25. Tieleman T, Hinton G (2012) Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning 4(2):26–31
26. Tsantekidis A, Passalis N, Tefas A, Kanniainen J, Gabbouj M, Iosifidis A (2017) Forecasting stock prices from the limit order book using convolutional neural networks. In: Business Informatics (CBI), 2017 IEEE 19th Conference on, IEEE, vol 1, pp 7–12
27. Tsantekidis A, Passalis N, Tefas A, Kanniainen J, Gabbouj M, Iosifidis A (2017) Using deep learning to detect price change indications in financial markets. In: Signal Processing Conference (EUSIPCO), 2017 25th European, IEEE, pp 2511–2515
28. Tsantekidis A, Passalis N, Tefas A, Kanniainen J, Gabbouj M, Iosifidis A (2018) Using deep learning for price prediction by exploiting stationary limit order book features. arXiv preprint arXiv:181009965
29. Van Den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior AW, Kavukcuoglu K (2016) Wavenet: A generative model for raw audio. In: SSW, p 125
30. Wang J, Zhou S, Guan J (2012) Detecting potential collusive cliques in futures markets based on trading behaviors from real data. Neurocomputing 92:44–53
31. Werbos PJ (1990) Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10):1550–1560
32. Zhang X, Li Y, Wang S, Fang B, Philip SY (2019) Enhancing stock market prediction with extended coupled hidden markov model over multi-sourced data. Knowledge and Information Systems 61(2):1071–1090