# Deep Learning approaches for Financial Forecasting

Daniele Domenichelli and Fiorenzo Parascandolo
University of Bologna

**Abstract**

The predictability of the Stock Market Trend is an open problem which, in various periods of the last one hundred and fifty years, has attracted the attention of numerous researchers, scientists and brokers. The debate was shelved in the early of seventies and then re-emerged simultaneously with the growing affirmation of Deep Learning at the beginning of 2000. The last proposals are, in fact, based on this technology and the most successful applications suggest turning financial data into images, in order to exploit the ability of Convolutional Neural Network. This allows to extract the characteristics describing the past trend and identify the most relevant patterns to predict future ones. In this report, we will describe different approaches to solve the task, inspired by the proposal of Barra [5]. The latter is based on Gramian Angular Field: a transformation which can be used to encode Time-Series to images and an ensemble architecture with a majority-based policy. We have used this approach as a baseline and then we tried to improve the performances using a different ensemble architecture and a Deep Reinforcement Learning algorithm for the policy. Eventually, we compared the obtained results with other works on the same task.

*Keywords:* Convolutional Neural Network; Financial Time Series; Gramian Angular Field.

## 1 Introduction to Stock Market

Is there a link between the ancient Greece, the pollen and Wall Street? Bachelier [4] stated that Stock Exchange fluctuations are determined not only by earlier ones, but also by present market position: therefore it is impossible to expect a mathematically exact forecast. The consequence of addiction on the current state implies that the Theory of Probability is not applicable to the movements of quoted prices and only the static states of the Market at a given instant can be dealt with probability frameworks. In 1827, R. Brown, a Scottish botanist, observed a chaotic and incessant motion of pollen grains thus coming to dispel the hypothesis that the movement was due to the fact that the pollen must have been alive. He observed through other experiments using dead plants pollen the same

phenomenon refuting the initial hypothesis. He could not give an explanation and his initial observation went unnoticed. The first completely mechanistic vision of the world dates back to Democritus (460 a.C.): reality is made up of atoms that move incessantly in a vacuum. Atoms are elementary particles, indivisible, different from each other only in quantitative or objective characteristics such as shape, size, order and position, endowed with eternal movement that is inherent to them. Albert Einstein [9] put each piece in its place: he demonstrated that the motion of a particle is a necessary condition for its existence and that the collective motion can be formalized mathematically. All at once, he explained Brown's remark, gave a market model (Brownian Motion) completing the one proposed by Bachelier and he confirmed, at least in part, the intuition of Democritus on the nature of the matter. After this historic discovery, Wiener formalized a rigorous mathematical model of the Bachelier's theory [27] based on the Wiener process which is a continuous time-indexed stochastic process. As Bachelier guessed, if all the price changes of a bond relating to a certain period of time are arranged in a diagram, they are arranged in the usual shape of a bell curve. Fama [10] deduced a more appealing feature which does not differ from the results of Wiener: markets are impossible to predict because they already incorporate all available information. This last statement is a synthesis of the so-called Efficient-Market Hypothesis (EMH) linked to random walk. The price of any stocks is given by:

$$P_t = E_t[M_{t+1}(P_{t+1} + D_{t+1})] \tag{1}$$

according to Fama, where, $E_t$ is the given information at time $t$; $M_{t+1}$ is the stochastic discount factor, and $D_{t+1}$ is the dividend the stock pays next period. This equation implies that stock prices follow a random walk iff the stochastic discount factor is constant. Therefore the EMH can be reduced to the constant stochastic discount factor hypothesis. In the late 1900s, Peters [19] postulates the Market Multimodal Hypothesis: the market does not follow a single rule and behaves in different ways:

- Efficient Mode when the Fama's hypothesis hold
- Transition Mode when the level of coherent investor behaviors increases due to new information and/or other factors, hence resulting in the impact of this information to extend for prolonged periods of time

- Chaotic Mode when the investor sentiment is very strong but variations in groupthink are increased as well
- Choerent Mode when investors act as more or less one unit, resulting in very strong positive or negative market trends

So far this has been found to be a highly realistic view of the actual operations of the market. It should be noted that Peter's hypothesis encapsulates the Fama's one and therefore the more general description consists in representing the market as a system that sometimes follows a random walk (stochastic process without memory like a coin flip) while sometimes it is possible to identify order patterns. More in detail, Peters [19] has empirically shown that the distribution of returns on the stock market follows a fractal distribution, knwon as stable Levy distribution:

$$\varphi(t;\alpha,\beta,\gamma,\delta) = \exp\left(it\delta - |\gamma t|^{\alpha}\left(1 - i\beta\,\mathrm{sgn}(t)\Phi\right)\right)$$
$$\Phi = \begin{cases} \left(|\gamma t|^{1-\alpha} - 1\right)\tan\left(\frac{\pi\alpha}{2}\right) & \alpha \neq 1 \\ -\frac{2}{\pi}\log|\gamma t| & \alpha = 1 \end{cases} \tag{2}$$

This distribution has four characteristics parameters that are continuously changing:

- $\delta$ is the location parameter of the mean
- $\gamma$ is a parameter to adjust for different granularity of data
- $\beta \in [-1, 1]$ is a measure of skewness to the left or the right
- $\alpha$ is a measure of the leptokurtosis of the distribution

for $\alpha = 2$, the distribution generalized the normal distribution. Peters [19] states that $\alpha \in [1, 2]$ in the stock market domain. Noninteger values of $\alpha$ result in fractional Brownian motions of prices, therefore the stock time series has a memory. Future prices are a function of new information as well as the past performance of the stock. However, as the parameters of Levy distibution are varying in time, in accordance with the Multimodal Hypothesis, the stocks have a continuously changing memory, therefore the estimation of the memory window size must be tuned. However, a system capable of predicting the stock market trend must have the following features:

- When the market is in Efficient Mode the system it should answer "I can't tell if the trend will go up or down"
- When the market has a memory it must predict the trend as accurately as possible
- It must be able to generalize and perform equally for different stocks

## 2 Problem Description and SOTA

Since it is possible to invest freely on the market, people with backgrounds belonging to different research fields continue to seek tools and frameworks to predict the stock market trend. The goal of everyone is to correctly direct their investments in order to obtain a profit. The data used are the famous financial time series which describe the so-called open-high-low-close chart.

Figure 1: Example of OLHC chart

where:

- Open is the value index when the market has opened in the specified date;
- Close is the value index when the market has closed in the specified date;
- High is the highest value reached by the index in the specified date;
- Low is the lowest value reached by the index in the specified date;

- Volume is the number of shares that changed hands during in the specified date

The problem commonly treated in the state of the art consists, given a certain date, in predicting whether the difference between Close and Open of the next observation is greater or less than zero. Usually, this difference is used as the label of the observations preceding the one to be predicted.

## 2.1 State of The Art

Before the 2000s, the approaches are focused on searching rules to identify pattern in the financial time series. Armstrong [2] used 99 rules developed with financial experts using 18 features of time series. The main problem with these techniques is the poor ability to generalize, resulting not very robust when using Stocks and Indices other than those used for testing. Therefore, since Machine Learning and Deep Learning techniques have established themselves, rule-based methods are no longer used to deal with this type of problem. The the most used techniques today are:

- Statistical: Data Compression, ARIMA, ESN, Regression
- Sentiment Analysis: Alamsyah [1] has shown a correlation between sentiment and Stock Market Trend
- Supervised or Unsupervised Learning
- Pattern Recognition
- Hybrid

Statistical approaches emerged in the post-rules period simultaneously with the diffusion of large datasets. Azhar [3] developed a data compression framework for Stock Market Prediction achieving an average accuracy of 55%. Other Statistical approaches are based on autoregressive integrated moving average (ARIMA) [12]. It is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series [7]. Another widely used technique, especially in competitions, is Sentiment Analysis on social

media. It's not a simple task because it first requires you to filter relevant financial information from unstructured data. Bollen [6] used Twitter data to measure the correlations between the public comments and the changes in DJIA. The authors used an effective filtering technique to remove spam tweets. However, the general assumption that the overall public mood affects the stock prices is naive because not all people who tweet invest in the stock markets [17]. Lee [13] proposed a text analysis model on the company's 8-K reports, which are more relevant compared to social media data. The authors demonstrated that text analysis helps to improve the prediction accuracy by 10%. Despite many works based on sentiment analysis they have obtained excellent results, in terms of accuracy, their robustness and ability to generalize has not been proven. The consequence is that these approaches only work well on certain indices and suffer a lot from overfitting. Supervised and Unsupervised learning methods have been known to be equally strong for stock prediction. Powell [20] drew an effective comparison between SVM and K-means and explained which distance metric for K-means and which kernel function for SVM achieves best results. Both models are tested on S&P 500 data and the results show that both techniques have similar performance, SVM achieves 89.1% and K-means achieves 85.6% respectively. Recently, Lv [14] synthetically evaluated various ML algorithms and observed the daily trading performance of stocks under transaction cost and no transaction cost. They utilized 424 S&P 500 index component stocks (SPICS) and 185 CSI 300 Index Component Stocks (CSICS) between 2010 and 2017 and compared traditional machine learning algorithms with advanced deep neural network (DNN) models. The traditional machine learning algorithms are SVM, Random Forest, Logistic Regression, naïve Bayes, Classification and Regression Tree (CART), and eXtreme Gradient Boosting while the DNN architectures include Multilayer Perceptron (MLP), Deep Belief Network (DBN), Stacked Autoencoders (SAE), RNN, LSTM, and GRU. Their results show that traditional machine learning algorithms have a better performance in most of the directional evaluation indicators without considering the transaction cost, however, DNN models show better performance considering transaction cost. However, among the works cited, all use data samples that are too small to really prove the goodness of an approach. The latest macro-category of techniques to solve the task are based on pattern recognition, which is also the type of solution proposed by us. Experienced traders, who observe financial time-series charts and execute buy and sell orders,

start developing an intuition for market opportunities. The intuition they develop based on their chart observations nearly reflects the recommendations that their state-of-the-art model provides. In this perspective, financial time-series analysis can be thought of as a visual process. That is, when experienced traders look at a time-series data, they process and act upon the image instead of mentally exercising algebraic operations on the sequence of numbers. More recently, it was suggested to approach time-series classification by first encoding the data as images and then utilize the power of computer vision algorithms for classification [18]. In an example, it was suggested to encode the time dependency, implicitly, as Gramian-Angular fields, Markov-Transition fields [26] or make use of recurrence plots [22]. Another work focused on transforming financial data into images to classify candlesticks patterns [23].

# 3 Proposed Approach

Our approach to predict the Stock Market Trend is inspired by the work of Barra [5]. He used an ensemble of CNNs, with the training phase executed over GAF images (particularly, the GADF). The data of the original time series are aggregated according to 4 intervals of time; then, consecutive time frames of 20 observations are extracted from each time series, in order to generate the related set of GADF images. Twenty similar CNNs are trained over these images, and the threshold-driven ensemble approach takes place for deciding which action to perform the day after the observations. We have proposed a different architecture for the ensemble and a deep reinforcment learning method to encapsulate the policy. We've also tried this approach with the candlestick representation of the financial time series as done by Tsai [23]. Eventually, we compared the obtained results.

## 3.1 Data

We used a Kaggle[1] dataset for our experiments. It contains the full historical daily price and volume data for all US-based stocks and ETFs trading on the NYSE, NASDAQ, and NYSE MKT in CSV format. However, considering the huge amount of data (about 100 million) and our limited computational capabilities, we have chosen to use only the first

[1] `https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs`

100 indexes of the ETFs repository. From this subset we have generated an image dataset[2] based on the GADF transformation. For each observation an image is constructed that represents the price trend of the last 100 days using 4 different aggregation periods. Each observation has been labelled according to the difference $Close - Open$ of the day after which is:

- 1, if $Close - Open > 0$
- 0, if $Close - Open \leq 0$

| Items | Negative Items | Positive Items |
|---|---|---|
| 154772 | 82611 | 72260 |

Table 1: Dataset distribution

### 3.1.1 Gramian Angular Field

The Gramian Angular Field (GAF) is a technique to encode time series as images proposed by Wang [26]. The idea is to represent time series in a polar co-ordinate system instead of the typical Cartesian coordinates. In the Gramian matrix, each element is actually the cosine of the summation of angles. Thanks to this transformation it is possible to treat time series with computer vision based approaches instead of the usual recurring neural networks. More in detail: given a time series $X = \{x_1, x_2, ..., x_n\}$ of $n$ real-valued observations, $X$ is rescaled so that all values fall in the interval $[-1, 1]$:

$$\hat{x_i} = \frac{(x_i - max(X)) + (x_i - min(X))}{max(X) - min(X)} \tag{3}$$

Once the rescaled time series $\hat{X}$ is obtained, its representation in a polar coordinate system is computed with the following equations:

$$\begin{cases} \theta = arccos(\hat{x_i}), & \hat{x_i} \in [-1, 1], \;\; \hat{x_i} \in \hat{X} \\ r = \frac{t_i}{N}, & t_i \in N \end{cases} \tag{4}$$

[2] `https://www.kaggle.com/ddomen/etfscloseopengadf/settings`

Finally, GASF and GADF can be computed as the sum/difference between the points of the time series.

$$\begin{aligned} GASF &= cos(\theta_i + \theta_j) \\ GADF &= sen(\theta_i - \theta_j) \end{aligned} \tag{5}$$

Since our task is to predict if the next day's $Close - Open$ difference is greater or less than zero, our $x_i$ are the difference between $Close$ and $Open$ of the previous days. More in details: let's consider the sample $x_i$ of the time series $X$, its label is $Close > Open$ of the sample $x_{i+1}$. Given $n$, the size of the time window, the image representing the sample $x_i$ is obtained by applying $GASF$ or $GADF$ transformation to the time series $\{x_{i-n}, ..., x_i\}$. The result of one of the two transformations is a $n \times n$ matrix representing an heatmap which can be easily transformed into an RGB image. In Figure 2 an example of $GASF$ image is shown. It represents the difference between $Close$ and $Open$ of the last 20 day taken 1 sample at a time. The minimum period that we can use to represent the samples of our dataset is equal

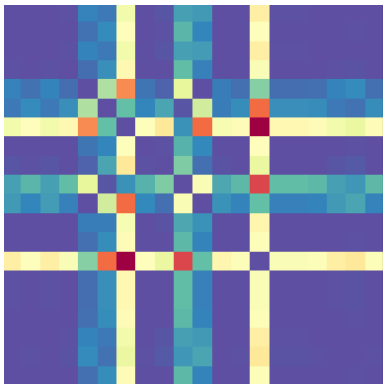

Figure 2: GASF image obtained with time window = 20 and period = 1

to 1 day since this is the minimum period that elapses between two consecutive samples. However, in order to obtain a more generic and "looking more backward" representation, following [5] proposal, we decided to create 3 more $GASF/GADF$ images for each sample using different time periods. Given $n$ and $t$, the size of the time window and the time period respectively, the image representing the sample $x_i$ is obtained by applying $GASF$ or $GADF$ transformation to the time series $\{mean(x_{i-(n\times t)+1}, x_{i-(n\times t)+2}, ..., x_{i-(n\times t)+t}), ...,$ $mean(x_{i-(n\times t)+(n-1)\times t+1}, x_{i-(n\times t)+(n-1)\times t+2}, ..., x_{i-(n\times t)+(n-1)\times t+t})\}$. For instance, let's consider a time window size = 20 and time period = 5: the first element of the aggregated

time series is the mean of the difference between *Close* and *Open* of the days from $x_{i-99}$ to $x_{i-95}$, while, the last element is the mean from $x_{i-4}$ to $x_i$. In Figure 3 an example of concatenated $GASF$ images with four different time periods is shown. In particular:

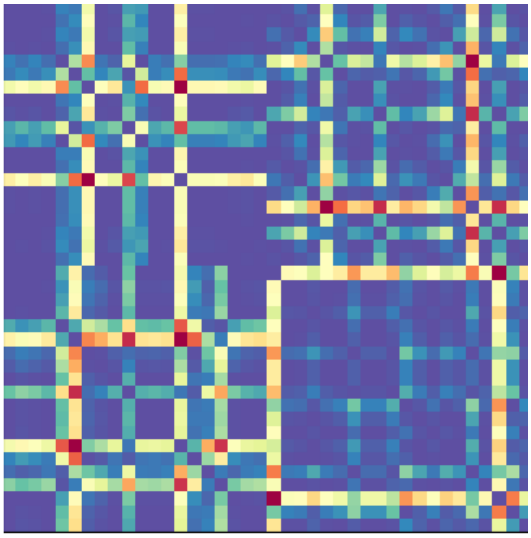

Figure 3: GASF images obtained with time window = 20 and periods = {1, 2, 3, 5}

- The top left $20 \times 20$ pixels is the $GASF$ image obtained by considering a time $window\_size = 20$ and $time\_period = 1$. Note that it is the image shown in Figures 2
- The top right $20 \times 20$ pixels is the $GASF$ image obtained by considering a time $window\_size = 20$ and $time\_period = 13$
- The bottom left $20 \times 20$ pixels is the $GASF$ image obtained by considering a time $window\_size = 20$ and $time\_period = 2$
- The bottom right $20 \times 20$ pixels is the $GASF$ image obtained by considering a time $window\_size = 20$ and $time\_period = 5$

Note that, the choice of the maximum time period to consider and the time window size establishes the size of the global $time\_window$. Considering our choice, for example, the maximum $time\_period = 5$ and $window\_size = 20$, therefore the last $20 \times 5$ days are used to build the whole image considering a certain sample. For this reason, the first 100 samples

of each ETF considered are used to build the first image, therefore the number of labelled items in each ETF is the number of items minus $20 \times 5$. Using what we have just discussed we have a generated a dataset of $40 \times 40$ $GADF$ images labeled with 0 or 1 if the trend of the next day is decreasing or increasing respectively.

### 3.1.2 Candlestick

Japanese start using technical analysis to trade rice in the 17th century [24], for this reason the famous Candlesticks are called Japanese Candlesticks. In Figure 3.1.2 the structure of

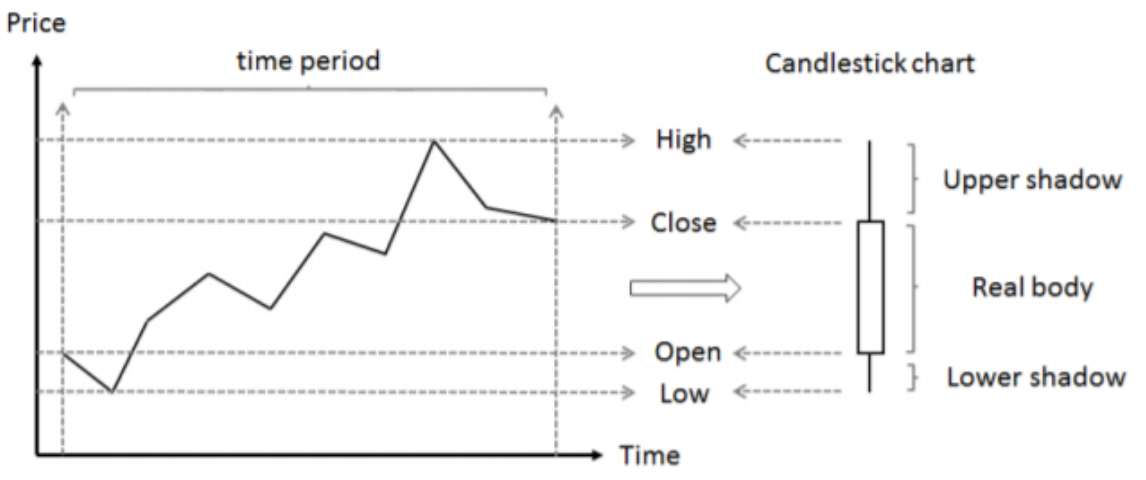


Figure 4: Candlesticks display all the market needed information, such as opening, closing, high, and low prices

a candlestick is shown:

- the real-body is the price difference between the opening and closing prices.
- the upper shadow is the price difference between the highest price and the real-body
- the lower shadow is the price difference between the lowest price and the real-body
- if the open price is higher than the close price the real-body is rendered in red (price is falling)
- if the open price is lower than the close price the real-body is rendered in green (price is raising)

Researchers focus on the topic of candlesticks for many years [16] in order to identify patterns to predict the Stock Market Trend. There is also a specific human figure, called a chartist, who implements trading strategies by observing sequences of Japanese Candlesticks, infact, this type of representation of financial time series is the most intuitive for a human agent. Following the proposal of [23], we decided to create a dataset of Candlestick[3] images, labelled as described for $GADF$ images, to compare this two different types of encoding with the approach proposed by [5].

## 3.2 Trading Strategy

The problem, as it has been described up to now, is a binary classification task. However the authors of [5] decided to add a degree of complexity in order to fit it into the context of real trading. The idea is to design the system to simulate a trading strategy. Specifically, the latter is modeled such that, for each sample (day) the final output of the system is one of the following actions:

- a long action, which consists of buying the stock, and then selling it before the market closes;
- a short action, which consists of selling the stock (using the mechanism of the uncovered sale), and then buying it before the market closes;
- a hold action, which consists of deciding not to invest in that day

Adding an action that gives the system the ability to do nothing is justified by the Multimodal Hypothesis discussed in Section 1. The ideal target of this strategy requires the system to choose the action which maximize the profit, given the prediction about the Stock Price Trend. A long action is performed when the system predicts that the trend is increasing; a short action if the trend will decrease; while, the hold action is performed when the system is not confident about its prediction. In order to represent the confidence of a prediction, we have developed the threshold-based approach proposed by [5] and a simple Deep Reinforcement Learning algorithm based on a deep Q Network.

[3] `https://www.kaggle.com/ddomen/etfscandlestick100/settings`

## 3.3 Architecture

Once we have described the data used we can pass the architecture of the neural networks that we used for our experiments. First we reimplemented the proposal of [5] which consists of a simplified version of the VGG-16 network [21]. Later we tried to implement a simplified ResNet [11], trying two different types of stem layers, with the same number of layers of the simplified version of the VGG-16. We have implemented also a Non Local Layer [25] that you can optionally insert in our architectures. For each architecture we have developed an Ensemble configuration which involves the training of twenty convolutional neural networks, each with a different weight initialization method.

### 3.3.1 Simplified VGG-16

The simplified version of VGG-16 is composed by 5 convolutional layers with $ReLU$ activation and a fully-connected one. There are not Batch Norm layers according to standard VGG design, however there is a Dropout layer with $p = 0.25$ after the first two Max Pool layers. In Figure 5 the global representation of the architecture is shown. Note that this

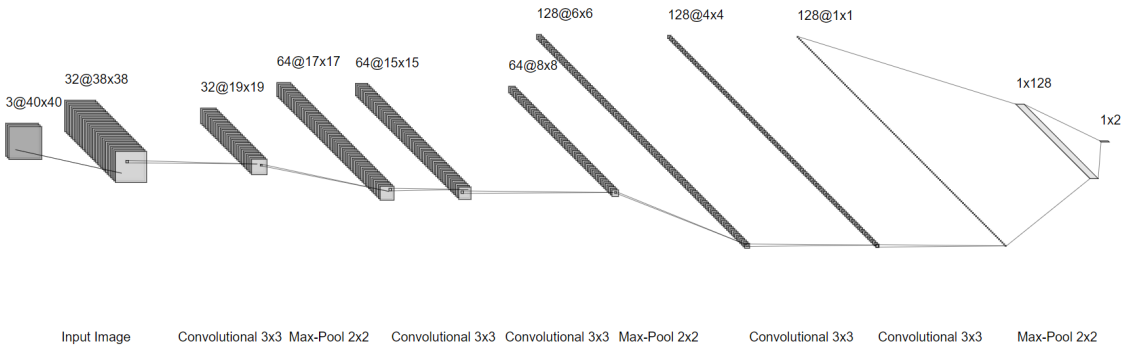


Figure 5: Simplified VGG-16

structure does not differentiate the sub-images that make up the global $GADF$ image. The kernels of the various convolutional layers are also applied pixels belonging to different sub-images (in the edges of the latter). This could be a problem for the extraction of the most semantically relevant features, why correlate the pixels of the right edge of the $GADF$ image obtained with $time\_period = 1$ with the pixels of the left edge of the $GADF$ image obtained with $time\_period = 3$? At the beginning, in the development of more performing

architectures than [5] proposal, we did not consider the latter a problem. However, the architecture that has obtained the best results respects the constraint of not passing the convolution kernel between two different sub-images and uses a Non Local Layer [25] to extract the global non local features.

### 3.3.2 Simplified ResNet

Our first proposal is a simplified version of the ResNet architecture which is often used as a baseline. This type of architecture was very successful when it was proposed because the use of Residual Blocks has been shown to be a useful component for optimizing large neural networks. The use of skip connection allows the network to skip the training of the layers; in a sense it is as if the problem of tuning the number of layers was done by the neural network itself. An other interpretation is that for different training data points, different parts of networks will be trained at different rates based on how the error flows backwards in the network. This can be thought of as training an ensemble of different models on the dataset and getting the best possible accuracy. Since [5] showed that using Ensemble is particularly good for this task, we are proposing to stress this last point by using in a sense an ensemble of ensemble. We used a Residual Block with skip connection with a one-dimensional convolutional layer like the one shown in the Figure 6. We also tried two different types of Stem Layer:

- $Stem\_1$: a classic one with $kernel\_size = 7$ and $stride = 2$ + Batch Norm + $ReLU$ activation (Figure 7)
- $Stem\_2$: $kernel\_size = 4$ and $stride = 4$ + Batch Norm + $ReLU$ activation (Figure 8)

The reasons for choosing the $Stem\_2$ derive from the structure of $GADF$ images, in fact using a $kernel\_size = 4$ and $stride = 4$ the kernel operates locally on each sub-image. Remember that the global image is the result of the concatenation of four $20 \times 20$ images, therefore, the application of the $Stem\_2$ layer produces an image with 32 channels and dimension $10 \times 10$, where the top/bottom left/right $5 \times 5$ images are produced from the initial sub-images. The goal of this approach is to create a deep representation of the images

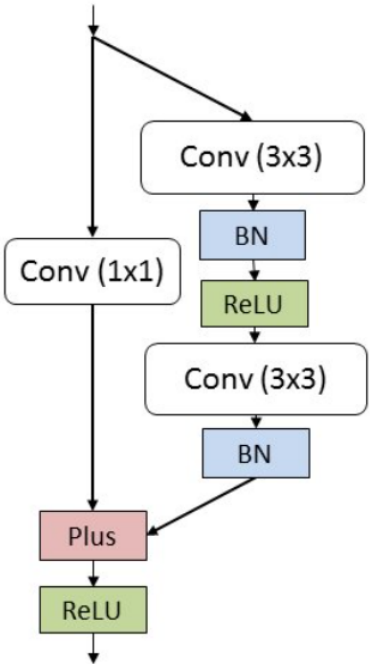


Figure 6: Residual Block

representing the Stock Market Trend, aggregated considering different periods, and then to create a global representation in the subsequent layers.

### 3.3.3 Non Local Block

Convolutional operations are building blocks that process one local neighborhood at a time. Wang [25] proposed a non-local operations as a generic family of building blocks for capturing long-range dependencies. This non-local operation computes the response at a position as a weighted sum of the features at all positions. It is inspired by Non-local means [8] is a classical filtering algorithm that computes a weighted mean of all pixels in an image. A generic non-local operation can be represented as follows:

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j) g(x_j) \tag{6}$$

In this equation $i$ is the index of the out position and $j$ is the index that enumerates all possible positions. A pairwise function $f$ computes a scalar (representing relationship such as affinity) between $i$ and all $j$. The unary function $g$ computes a representation

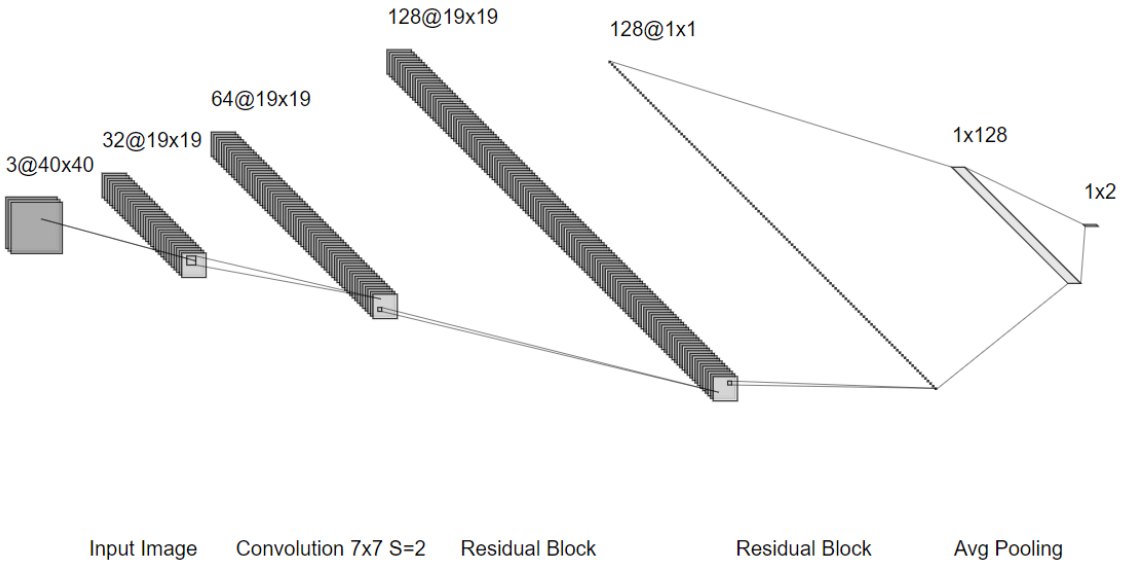


Figure 7: Simplified ResNet with $Stem\_1$

of the input signal at the position $j$. The response is normalized by a factor $C(x)$. The non-local behaviour is due to the fact that all positions $j$ are considered to compute the response for the pixel at position $i$. The non-local operation is also different from a Fully Connected Layer (FC) because the responses based are based on relationships between different locations, whereas FC uses learned weights. In other words, the relationship between $x_j$ and $x_i$ is not a function of the input data in FC, unlike in non-local layers. In Figure 3.3.3 a space-time Non Local Block is shown. The pairwise function used is the Embedded Gaussian while $C(x)$ is the normalization:

$$\begin{aligned} f(x_i, x_j) &= e^{\theta(x_i)^T \phi(x_j)} \\ C(x) &= \sum_{\forall j} f(x_i, x_j) \end{aligned} \tag{7}$$

The ratio between this two terms is computed with $softmax$ while $g(x_j) = W_j x_j$ is a linear embedding that can be implemented as a 1 Dimensional Convolution. The Non-Local Block is an optional component that can be easily added in several architecture because it produces in output a tensor with the same size before applying the non-local operations. However, in our experiments we have inserted it after the stemming layer.

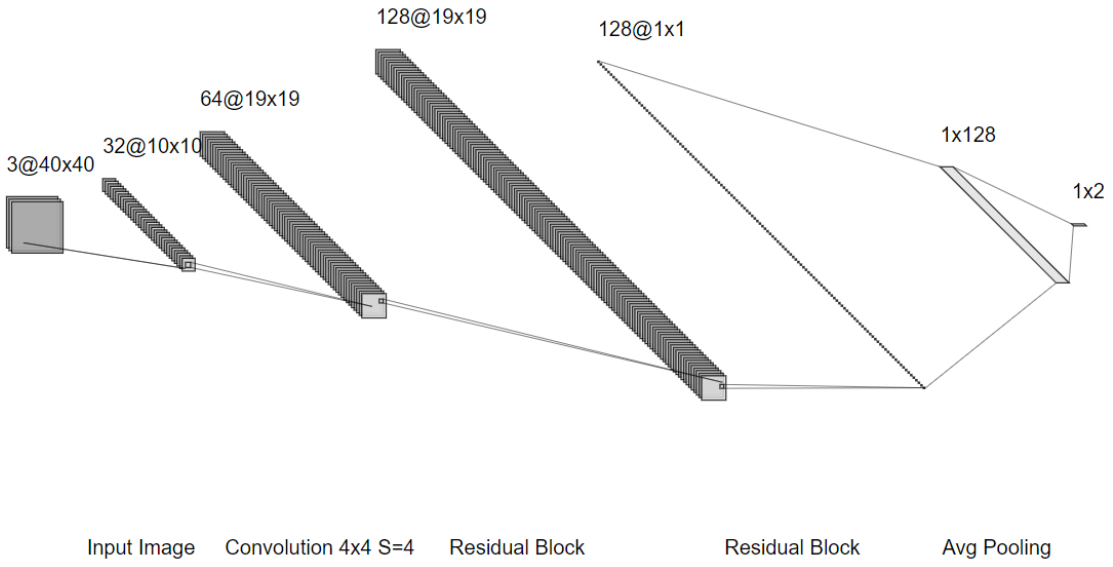


Figure 8: Simplified ResNet with $Stem\_2$

## 3.4 Policy

The ensemble configuration consists of driving twenty CNNs with different initializations, each of these chooses a long or short action given a $GADF$ image. For this purpose we have implemented the proposal of [5], based on threshold and we have implemented a Reinforcement Learning algorithm based on Deep Q learning [15] to encapsulate the policy.

### 3.4.1 Threshold-based approach

Collective behavior consists by applying a majority voting based approach, which returns a specific output $r \in \{0, 1\}$, according to the percentage of networks which agree with the same classification. For this reason a threshold is used such that if the percentage of CNN according to an action is greater than the threshold then the action is returned to the output. Instead, if the majority does not agree on the action to be performed, given the threshold, a *hold* action is performed.

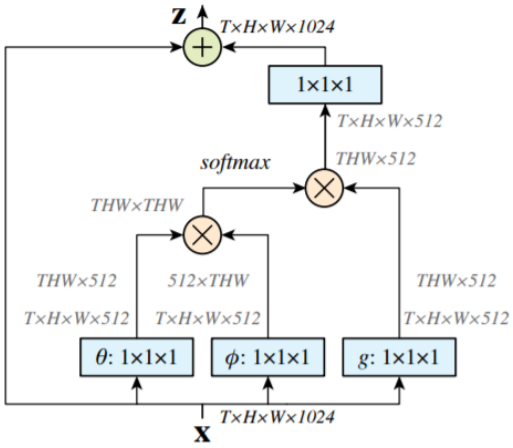


Figure 9: A space-time Non Local Block

### 3.4.2 Deep Q Learning

We think that trading can be formalized as a Reinforcement Learning problem because the components needed to describe it fit very well:

- The environment is the market
- The observations of the environment are the $GADF$ images (or an other encoding of the Financial Time Series)
- The actions are: $Long$, $Short$, $Hold$ (however this is a simplification as $Long$ and $Short$ should be quantified)
- The rewards are the gains or losses

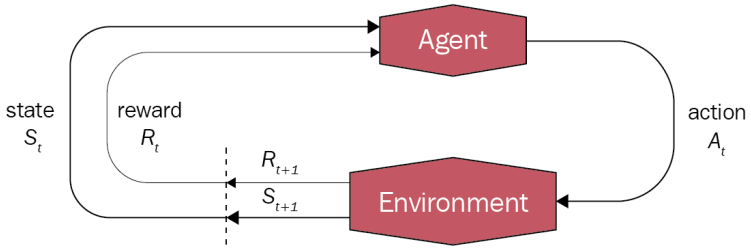


Figure 10: Reinforcement Learning framework

We have created a simplified reward system. Given the observation of a $GADF$ image:

- if the agent performs a *Long* action and the trend will decrease therefore it will receive a reward equal to 1, -1 otherwise
- if the agent performs a *Short* action and the trend will increase therefore it will receive a reward equal to 1, -1 otherwise
- if the agent performs an *Hold* action the reward is 0

We have also simulated a wallet in order to simulate the simplified trading environment, such that, given an observation, the agent can choose to bet 1 dollar to receive 2 if he guesses or lose it if he is wrong; or he can decide not to play. This choice is linked to the Efficient Market Hypothesis of [10]: if the market follows a random walk, in the long term, the starting wallet should be like the initial wallet. From an implementation point of view, developing a Reinforcement Learning algorithm to encapsulate the policy described above, consists in adding a FC with three outputs (one for each possible action) that takes as input the outputs of each CNN of the ensemble. In this case the training of the individual CNNs is collective, the common goal is to maximize cumulative rewards (which represent gains and losses following the application of an action on the environment). The agent is trained with a Deep Q Learning algorithm.

# 4 Experimental Settings

At this point we provide some details on the two proposed approaches:

- In the classification-based one, each CNN is trained independently from the others. We have used the Cross Entropy Loss, as usual for the classification tasks
- In the Reinforcement Learning-based one, CNNs are trained as a collective. We have used the Smooth L1 Loss as usual for DQN

However, for both approaches, we used Adam optimizer with $lr = 0.0007$ and $batch\_size = 30$. The threshold is set to 0.8 since with this value the best results were obtained in [5].

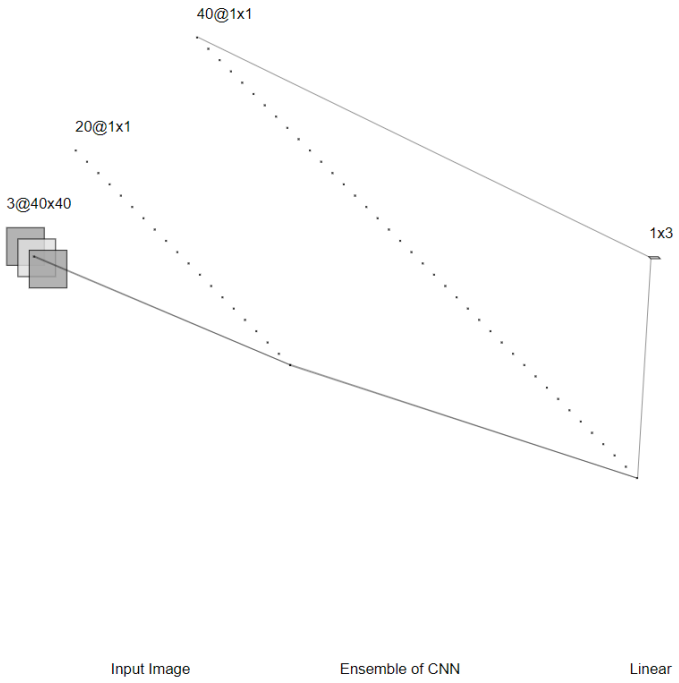


Figure 11: Our DQN architecture

## 4.1 Initialization

An important aspect proposed in [5] is the initialization of the weights of the CNNs. The choice of defining each network with different weight initialization methods arises from the need of excluding most of the randomness factors which can affect the final prediction. This ensures that the answer of the ensemble originates from a proper convergence of the weights, no matters how they are initialized.In Figure 4.1 the inizialization configuration is shown.

## 4.2 Rotation and Rhombus

We have also developed a transformation that can be added to the image pre-processing pipeline. Since $GAF$ images are symmetrical with respect to the diagonal there are redundant patterns. Using in combination rotation and the application of a diamond as a patch on the image it is possible to remove redundant patterns as shown in the Figure 4.2.

| Init. method | Configuration parameters | seed |
|---|---|---|
| Orthogonal | gain=1.0 | - |
| lecun_uniform | - | - |
| VarianceScaling | scale=1, mode='fan_in', distr.='normal' | - |
| RandomNormal | mean=0.0, stddev=0.05 | - |
| RandomUniform | minval=-0.05, maxval=0.05 | - |
| TruncatedNormal | mean=0.0, stddev=0.05 | - |
| glorot_normal | - | - |
| glorot_uniform | - | - |
| he_normal | - | - |
| he_uniform | - | - |
| Orthogonal | gain=1.0 | 42 |
| lecun_uniform | - | 42 |
| VarianceScaling | scale=1, mode='fan_in', distr.='normal' | 42 |
| RandomNormal | mean=0.0, stddev=0.05 | 42 |
| RandomUniform | minval=-0.05, maxval=0.05 | 42 |
| TruncatedNormal | mean=0.0, stddev=0.05 | 42 |
| glorot_normal | - | 42 |
| glorot_uniform | - | 42 |
| he_normal | - | 42 |
| he_uniform | - | 42 |

Figure 12: Initialization configuration

## 4.3 Walks

With respect to the common cross-validation approaches, which are typically applied when dealing with image classification, like the Leave-One-Out cross validation (LOOCV) or the k-fold cross validation, time series data need a more specific purpose approach, since they need to consider the semantic linking between the observation at time $t$ and the one at time $t + 1$. The representation of the training / validation phase is shown in the Figure 4.3. The goal of this approach is to simulate the work of a trader: the model is trained with four batches and then it is validated over the next batch, with respect the to training ones, from a temporal point of view. This can be seen as an artificial agent watching the Stock Market Trend for 120 days (let's recall that $batch\_size = 30$) and then it acts in the next month. However it is important to emphasize that this approach represents a very simplified strategy: the moment in which the agent acts (is validated) is not necessarily the best, but is forced to do so after being towed for 120 days. Adding a $Hold$ action, which the agent can choose when he is not confident about the prediction, should also help the agent deal with this problem. An interesting aspect of this approach is that it allows us to evaluate the agent for each Stock and therefore makes it easier to evaluate the robustness

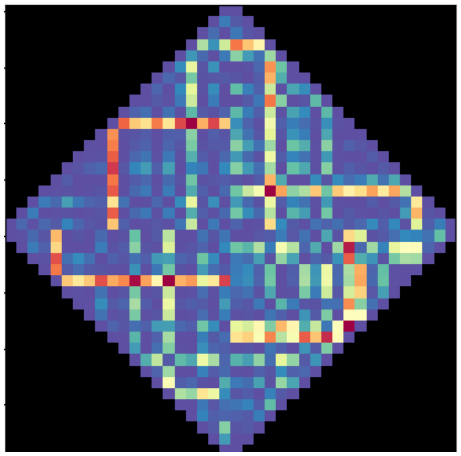

Figure 13: Rhombus Transformation

of the solution and how it performs for different indices.

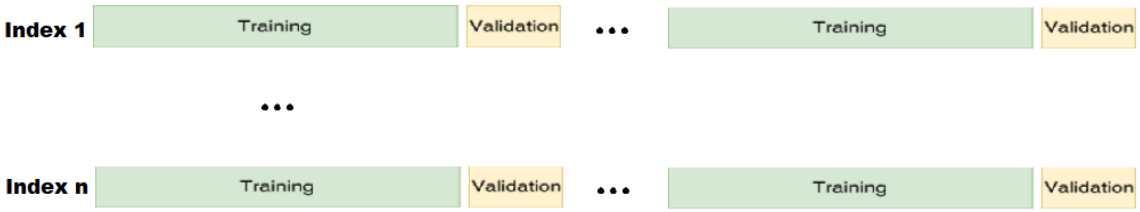


Figure 14: Walks definition

# 5 Results and Discussions

In this section, the quantitative and qualitative results are shown and discussed. In the following we report the metrics we have used:

- *Accuracy* is the ratio between the number of correct predictions (actions) and the number of answers (the total number of not $Hold$ choices).
- *Coverage* is the ratio between the number of $Hold$ actions and the number of samples
- $Starting\_Wallet$ represents the initial number of tokens available to the agent.

| | Accuracy (%) | Coverage (%) | Starting_Wallet | Final_Wallet |
|---|---|---|---|---|
| VGG | 54.33 | 84.47 | 100 | 2129 |
| VGG-Rhombus | 54.36 | 81.80 | 100 | 2109 |
| VGG-DQN | 55.19 | 39.86 | 100 | 1201 |
| ResNet-Stem_1 | 54.58 | 84.47 | 100 | 2323 |
| ResNet-Stem_2 | 54.78 | 81.27 | 100 | 2392 |
| ResNet-Stem_2-NonLocal | 54.97 | 82.57 | 100 | 2449 |
| ResNet-Stem_2-NonLocal-Rhombus | 54.31 | 84.96 | 100 | 2202 |
| ResNet-Stem_2-NonLocal-DQN | 55.59 | 62.71 | 100 | 2002 |

Table 2: The results of our experiments

- $Final\_Wallet$ represents the final number of tokens available to the agent. Every time the agent makes a correct prediction he receives 1 token, when he makes a mistake he loses 1, when he chooses no action he receives and loses nothing.

In Table 2 the results of our experiments are shown. The architecture proposed by [5] has reached 54.33% of Accuracy on this dataset which is a little less than those achieved in his paper. However, he used a dataset much smaller than ours (about 5000 items) and also used data sampled every 5 minutes while ours are daily. The first observation that can be made is that this task is very difficult, an accuracy close to 50% is not the best for a binary classification problem. The Rhombus transformation improves a little the accuracy of VGG but it has not the same effect on ResNet. The approach based on Reinforcement Learning improves the *Accuracy* a little but greatly decreases the *Coverage* and consequently the $Final\_Wallet$. Finally, all our proposals perform better than [5] proposal; $ResNet - Stem\_2 - NonLocal$ brings an improvement of 0.64% to *Accuracy* and 320 to the $final\_wallet$. Note that small variations in accuracy cause large variations in profit. this gives us great satisfaction because VGG has generated a profit of $82.000 in [5].

# 6 Conclusion

In this work we have developed an alternative architecture for the ensemble proposed in [5]. In addition we have added an approach based on Reinforcement Learning to develop a flexible policy. Even if we have not obtained the best results with the latter we believe it is the way to follow the domain of Trading seems to us to be tailor-made for DRL approaches. We believe that the first thing to improve is the reward system since we have proposed one that is too simplified. Obviously we believe that the best is the real one: assign the profits and/or losses obtained during real trading as rewards. In parallel we would like to improve our knowledge of the sector in order to design more specific and efficient architectures. The addition of Non Local Blocks has been shown to have a positive impact on performances, therefore we believe that in the financial domain there are correlations between features that are not temporally local, so we will investigate this last aspect better. The results achieved in this work give us hope because we got an improvement in the accuracy of 0.69% compared to an approach that had achieved a profit of $82,000 in real trading.

# References


[1] A. Alamsyah. Exploring relationship between headline news sentiment and stock return. *7th International Conference on Information and Communication Technology*, 2019.

[2] J. S. Armstrong. Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations. *Management Science*, 1992.

[3] S. Azhar. Data compression techniques for stock market prediction. *Proceedings of the Data Compression Conference*, 1994.

[4] L. Bachelier. Theory of speculation. *Annales scientifiques de l'Ecole Normale Sup ´ erieure*, 1900.

[5] S. Barra. Deep learning and time series-to-image encoding for financial forecasting. *Journal of Automatica*, 2019.

[6] J. Bollen. Twitter mood predicts the stock market. *Journal of Computational Science*, 2011.

[7] G. E. P. Box. Time series analysis: Forecasting and control. 2015.

[8] A. Buades. A non-local algorithm for image denoising. *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[9] A. Einstein. On the movement of small suspended particles in a stationary liquid according to the molecular kinetic theory of heat. *Annalen der Physik*, 1905.

[10] E. Fama. The behavior of stock-market prices. *The Journal of Business*, 1965.

[11] K. He. Deep residual learning for image recognition. 2015.

[12] M. E. Hiransha. Nse stock market prediction using deep-learning models. *Procedia Computer Science*, 2018.

[13] Lee. On the importance of text analysis for stock price prediction. *9th International Conference on Language Resources and Evaluation*, 2014.

[14] Lv. An empirical study of machine learning algorithms for stock daily trading strategy. *Mathematical Problems in Engineering*, 2019.

[15] V. Mnih. Human-level control through deep reinforcement learning. *Nature*, 2015.

[16] S. Nison. Japanese candlestick charting techniques: a contemporary guide to the ancient investment techniques of the far east. *Penguin*, 2001.

[17] Pagolu. Sentiment analysis of twitter data for predicting stock market movements. *International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 2016.

[18] D. S. Park. Specaugment: A simple data augmentation method for automatic speech recognition. *Mathematical Problems in Engineering*, 2019.

[19] E. E. Peters. *Chaos and order in the capital markets*. 1991.

[20] N. Powell. Supervised and unsupervised methods for stock trend forecasting. *40th Southeastern Symposium on System Theory (SSST 2008)*, 2008.

[21] Simonyan. Very deep convolutional networks for large-scale image recognition. *Penguin*, 2014.

[22] V. Souza. Extracting texture features for time series classification. *22nd International Conference on Pattern Recognition*, 2014.

[23] Y.-C. Tsai. Encoding candlesticks as images for patterns classification using convolutional neural networks. 2019.

[24] G. S. Wagner. Trading applications of japanese candlestick charting. 1994.

[25] X. Wang. Non-local neural networks. 2018.

[26] Z. Wang. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[27] N. Wiener. Differential-space. *Journal of Mathematics and Physics*, 1923.