

# Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019

Omer Berat Sezer<sup>a</sup>, M. Ugur Gudelek<sup>a</sup>, Ahmet Murat Ozbayoglu<sup>a</sup>

<sup>a</sup>*Department of Computer Engineering, TOBB University of Economics and Technology, Ankara, Turkey*

---

## Abstract

Financial time series forecasting is, without a doubt, the top choice of computational intelligence for finance researchers from both academia and financial industry due to its broad implementation areas and substantial impact. Machine Learning (ML) researchers came up with various models and a vast number of studies have been published accordingly. As such, a significant amount of surveys exist covering ML for financial time series forecasting studies. Lately, Deep Learning (DL) models started appearing within the field, with results that significantly outperform traditional ML counterparts. Even though there is a growing interest in developing models for financial time series forecasting research, there is a lack of review papers that were solely focused on DL for finance. Hence, our motivation in this paper is to provide a comprehensive literature review on DL studies for financial time series forecasting implementations. We not only categorized the studies according to their intended forecasting implementation areas, such as index, forex, commodity forecasting, but also grouped them based on their DL model choices, such as Convolutional Neural Networks (CNNs), Deep Belief Networks (DBNs), Long-Short Term Memory (LSTM). We also tried to envision the future for the field by highlighting the possible setbacks and opportunities, so the interested researchers can benefit.

*Keywords:* deep learning, finance, computational intelligence, machine learning, time series forecasting, CNN, LSTM, RNN

---

## 1. Introduction

The finance industry has always been interested in successful prediction of financial time series data. Numerous studies have been published that were based on ML models with relatively better performances compared to classical time series forecasting techniques. Meanwhile, the widespread application of automated electronic trading systems coupled with increasing demand for higher yields keeps forcing the researchers and practitioners to continue working on searching for better models. Hence, new publications and implementations keep pouring into finance and computational intelligence literature.

In the last few years, DL started emerging strongly as the best performing predictor class within the ML field in various implementation areas. Financial time series forecasting is no exception, as such, an increasing number of prediction models based on various DL techniques were introduced in the appropriate conferences and journals in recent years.

Despite the existence of the vast amount of survey papers covering financial time series forecasting and trading systems using traditional soft computing techniques, to the best of our knowledge, no reviews have been performed in literature for **DL**. Hence, we decided to work on such a comprehensive study focusing on **DL** implementations of financial time series forecasting. Our motivation is two-fold such that we not only aimed at providing the state-of-the-art snapshot of academic and industry perspectives of the developed **DL** models but also pinpointing the important and distinctive characteristics of each studied model to prevent researchers and practitioners to make unsatisfactory choices during their system development phase. We also wanted to envision where the industry is heading by indicating possible future directions.

Our fundamental motivation in this paper was to come up with answers for the following research questions:

- Which **DL** models are used for financial time series forecasting ?
- How is the performance of **DL** models compared with traditional **ML** counterparts ?
- What is the future direction for **DL** research for financial time series forecasting ?

Our focus was solely on **DL** implementations for financial time series forecasting. For other **DL** based financial applications such as risk assessment, portfolio management, etc., interested readers can check the recent survey paper [1]. Since we singled out financial time series prediction studies in our survey, we omitted other time series forecasting studies that were not focused on financial data. Meanwhile, we included time-series research papers that had financial use cases or examples even though the papers themselves were not directly intended for financial time series forecasting. Also, we decided to include algorithmic trading papers that were based on financial forecasting, but ignore the ones that did not have a time series forecasting component.

We reviewed journals and conferences for our survey, however, we also included Masters and PhD theses, book chapters, arXiv papers and noteworthy technical publications that came up in web searches. We decided to only include the articles in the English language.

During our survey through the papers, we realized that most of the papers using the term "deep learning" in their description were published in the last 5 years. However, we also encountered some older studies that implemented deep models; such as **Recurrent Neural Networks (RNNs)**, Jordan-Elman networks. However, at their time of publication, the term "deep learning" was not in common usage. So, we decided to also include those papers.

According to our findings, this will be one of the first comprehensive "financial time series forecasting" survey papers focusing on **DL**. A lot of **ML** reviews for financial time series forecasting exist in the literature, meanwhile, we have not encountered any study on **DL**. Hence, we wanted to fill this gap by analyzing the developed models and applications accordingly. We hope, as a result of this paper, the researchers and model developers will have a better idea of how they can implement **DL** models for their studies.

We structured the rest of the paper as follows. Following this brief introduction, in Section 2, the existing surveys that are focused on **ML** and soft computing studies for financial time series forecasting are mentioned. In Section 3, we will cover the existing **DL**

models that are used, such as CNN, LSTM, Deep Reinforcement Learning (DRL). Section 4 will focus on the various financial time series forecasting implementation areas using DL, namely stock forecasting, index forecasting, trend forecasting, commodity forecasting, volatility forecasting, foreign exchange forecasting, cryptocurrency forecasting. In each subsection, the problem definition will be given, followed by the particular DL implementations. In Section 5, overall statistical results about our findings will be presented including histograms about the yearly distribution of different subfields, models, publication types, etc. As a result, the state-of-the-art snapshot for financial time series forecasting studies will be given through these statistics. At the same time, it will also show the areas that are already mature, compared against promising or new areas that still have room for improvement. Section 6 will provide discussions about what has been done through academic and industrial achievements and expectations through what might be needed in the future. The section will include highlights about the open areas that need further research. Finally, we will conclude in Section 7 by summarizing our findings.

## 2. Financial Time Series Forecasting with ML

Financial time series forecasting and associated applications have been studied extensively for many years. When ML started gaining popularity, financial prediction applications based on soft computing models also became available accordingly. Even though our focus is particularly on DL implementations of financial time series prediction studies, it will be beneficial to briefly mention about the existing surveys covering ML-based financial time series forecasting studies in order to gain historical perspective.

In our study, we did not include any survey papers that were focused on specific financial application areas other than forecasting studies. However, we were faced with some review publications that included not only financial time-series studies but also other financial applications. We decided to include those papers in order to maintain the comprehensiveness of our coverage.

Examples of these aforementioned publications are provided here. There were published books on stock market forecasting [2], trading system development [3], practical examples of forex and market forecasting applications [4] using ML models like Artificial Neural Networks (ANNs), Evolutionary Computations (ECs), Genetic Programming (GP) and Agent-based models [5].

There were also some existing journal and conference surveys. Bahrammirzaee et. al. [6] surveyed financial prediction and planning studies along with other financial applications using various Artificial Intelligence (AI) techniques like ANN, Expert Systems, hybrid models. The authors of [7] also compared ML methods in different financial applications including stock market prediction studies. In [8], soft computing models for the market, forex prediction and trading systems were analyzed. Mullainathan and Spies [9] surveyed the prediction process in general from an econometric perspective.

There were also a number of survey papers concentrated on a single particular ML model. Even though these papers focused on one technique, the implementation areas generally

spanned various financial applications including financial time series forecasting studies. Among those soft computing methods, **EC** and **ANN** had the most overall interest.

For the **EC** studies, Chen wrote a book on **Genetic Algorithms (GAs)** and **GP** in Computational Finance [10]. Later, **Multiobjective Evolutionary Algorithms (MOEAs)** were extensively surveyed on various financial applications including financial time series prediction [11, 12, 13]. Meanwhile, Rada reviewed **EC** applications along with Expert Systems for financial investing models [14].

For the **ANN** studies, Li and Ma reviewed implementations of **ANN** for stock price forecasting and some other financial applications [15]. The authors of [16] surveyed different implementations of **ANN** in financial applications including stock price forecasting. Recently, Elmsili and Outtaj contained **ANN** applications in economics and management research including economic time series forecasting in their survey [17].

There were also several text mining surveys focused on financial applications (which included financial time series forecasting). Mittermayer and Knolmayer compared various text mining implementations that extract market response to news for prediction [18]. The authors of [19] focused on news analytics studies for prediction of abnormal returns for trading strategies in their survey. Nassirtoussi et. al. reviewed text mining studies for stock or forex market prediction [20]. The authors of [21] also surveyed text mining-based time series forecasting and trading strategies using textual sentiment. Similarly, Kumar and Ravi [22] reviewed text mining studies for forex and stock market prediction. Lately, Xing et. al. [23] surveyed natural language-based financial forecasting studies.

Finally, there were application-specific survey papers that focused on particular financial time series forecasting implementations. Among these studies, stock market forecasting had the most interest. A number of surveys were published for stock market forecasting studies based on various soft computing methods at different times [24, 25, 26, 27, 28, 29, 30, 31]. Chatterjee et. al. [32] and Katarya and Mahajan [33] concentrated on **ANN**-based financial market prediction studies whereas Hu et. al. [34] focused on **EC** implementations for stock forecasting and algorithmic trading models. In a different time series forecasting application, researchers surveyed forex prediction studies using **ANN** [35] and various other soft computing techniques [36].

Even though, many surveys exist for **ML** implementations of financial time series forecasting, **DL** has not been surveyed comprehensively so far despite the existence of various **DL** implementations in recent years. Hence, this was our main motivation for the survey. At this point, we would like to cover the various **DL** models used in financial time series forecasting studies.

### 3. Deep Learning

**DL** is a type of **ANN** that consists of multiple processing layers and enables high-level abstraction to model data. The key advantage of **DL** models is extracting the good features of input data automatically using a general-purpose learning procedure. Therefore, in the literature, **DL** models are used in lots of applications: image, speech, video, audio reconstruction, natural language understanding (particularly topic classification), sentiment

analysis, question answering and language translation [37]. The historical improvements on **DL** models are surveyed in [38].

Financial time series forecasting has been very popular among **ML** researchers for more than 40 years. The financial community got a new boost lately with the introduction of **DL** models for financial prediction research and a lot of new publications appeared accordingly. The success of **DL** over **ML** models is the major attractive point for the finance researchers. With more financial time series data and different deep architectures, new **DL** methods will be proposed. In our survey, we found that in the vast majority of the studies, **DL** models were better than **ML** counterparts.

In literature, there are different kinds of **DL** models: Deep Multilayer Perceptron (**DMLP**), **RNN**, **LSTM**, **CNN**, **Restricted Boltzmann Machines (RBMs)**, **DBN**, **Autoencoder (AE)**, and **DRL** [37, 38]. Throughout the literature, financial time series forecasting was mostly considered as a regression problem. However, there were also a significant number of studies, in particular trend prediction, that used classification models to tackle financial forecasting problems. In Section 4, different **DL** implementations are provided along with their model choices.

### 3.1. Deep Multi Layer Perceptron (**DMLP**)

**DMLPs** is one of the first developed **ANNs**. The difference from shallow nets is that **DMLP** contains more layers. Even though particular model architectures might have variations depending on different problem requirements, **DMLP** models consist of mainly three layers: input, hidden and output. The number of neurons in each layer and the number of layers are the hyperparameters of the network. In general, each neuron in the hidden layers has input ( $x$ ), weight ( $w$ ) and bias ( $b$ ) terms. In addition, each neuron has a nonlinear activation function which produces a cumulative output of the preceding neurons. Equation 1 [39] illustrates an output of a single neuron in the **Neural Network (NN)**. There are different types of nonlinear activation functions. Most commonly used nonlinear activation functions are: sigmoid (Equation 2) [40], hyperbolic tangent (Equation 3) [41], Rectified Linear Unit (**ReLU**) (Equation 4) [42], leaky-**ReLU** (Equation 5) [43], swish (Equation 6) [44], and softmax (Equation 7) [39]. The comparison of the nonlinear activations are studied in [44].

$$y_i = \sigma(\sum_i W_i x_i + b_i) \quad (1)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2)$$

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (3)$$

$$R(z) = \max(0, z) \quad (4)$$

$$R(z) = 1(x < 0)(\alpha x) + 1(x \geq 0)(x) \quad (5)$$

$$f(x) = x\sigma(\beta x) \quad (6)$$

$$\text{softmax}(z_i) = \frac{\exp z_i}{\sum_j \exp z_j} \quad (7)$$

**DMLP** models have been appearing in various application areas [45, 37]. Using a **DMLP** model has advantages and disadvantages depending on the problem requirements. Through **DMLP** models, problems such as regression and classification can be solved by modeling the input data [46]. However, if the number of the input features is increased (e.g. image as input), the parameter size in the network will increase accordingly due to the fully connected nature of the model and it will jeopardize the computation performance and create storage problems. To overcome this issue, different types of **Deep Neural Network (DNN)** methods are proposed (such as **CNN**) [37]. With **DMLP**, much more efficient classification and regression processes are performed. In Figure 1, a **DMLP** model, layers, neurons in layers, weights between neurons are shown.

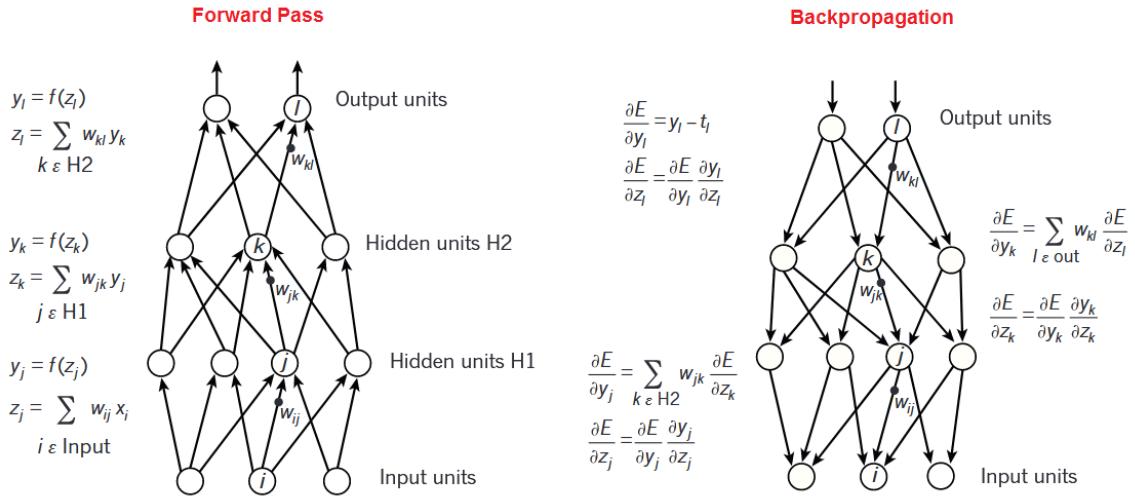


Figure 1: Deep Multi Layer Neural Network Forward Pass and Backpropagation [37]

**DMLP** learning stage is implemented through backpropagation. The amount of error in the neurons in the output layer is propagated back to the preceding layers. Optimization algorithms are used to find the optimum parameters/variables of the **NNs**. They are used to update the weights of the connections between the layers. There are different optimization algorithms that are developed: **Stochastic Gradient Descent (SGD)**, SGD with Momentum, **Adaptive Gradient Algorithm (AdaGrad)**, Root Mean Square Propagation (**RMSProp**), **Adaptive Moment Estimation (ADAM)** [47, 48, 49, 50, 51]. Gradient descent is an iterative method to find optimum parameters of the function that minimizes the cost function. **SGD** is an algorithm that randomly selects a few samples instead of the whole data

set for each iteration [47]. **SGD** with Momentum remembers the update in each iteration that accelerates gradient descent method [48]. **AdaGrad** is a modified **SGD** that improves convergence performance over standard **SGD** algorithm [49]. **RMSProp** is an optimization algorithm that provides the adaptation of the learning rate for each of the parameters. In **RMSProp**, the learning rate is divided by a running average of the magnitudes of recent gradients for that weight [50]. **ADAM** is updated version of **RMSProp** that uses running averages of both the gradients and the second moments of the gradients. **ADAM** combines advantages of the **RMSProp** (works well in online and non-stationary settings) and **AdaGrad** (works well with sparse gradients) [51].

As shown in Figure 1, the effect of the backpropagation is transferred to the previous layers. If the effect of **SGD** is gradually lost when the effect reaches the early layers during backpropagation, this problem is called vanishing gradient problem in the literature [52]. In this case, updates between the early layers become unavailable and the learning process stops. The high number of layers in the neural network and the increasing complexity cause the vanishing gradient problem.

The important issue in the **DMLP** are the hyperparameters of the networks and method of tuning these hyperparameters. Hyperparameters are the variables of the network that affect the network architecture, and the performance of the networks. The number of hidden layers, the number of units in each layer, regularization techniques (dropout, L1, L2), network weight initialization (zero, random, He [53], Xavier [54]), activation functions (Sigmoid, **ReLU**, hyperbolic tangent, etc.), learning rate, decay rate, momentum values, number of epochs, batch size (minibatch size), and optimization algorithms (**SGD**, **AdaGrad**, **RMSProp**, **ADAM**, etc.) are the hyperparameters of **DMLP**. Choosing better hyperparameter values/variables for the network result in better performance. So, finding the best hyperparameters for the network is a significant issue. In literature, there are different methods to find best hyperparameters: **Manual Search (MS)**, **Grid Search (GS)**, **RandomSearch (RS)**, **Bayesian Methods (Sequential Model-Based Global Optimization (SMBGO))**, **The Gaussian Process Approach (GPA)**, **Tree-structured Parzen Estimator Approach (TSPEA)**) [55, 56].

### 3.2. Recurrent Neural Network (RNN)

**RNN** is another type of **DL** network that is used for time series or sequential data, such as language and speech. **RNNs** are also used in traditional **ML** models (**Back Propagation Through Time (BPTT)**, Jordan-Elman networks, etc.), however, the time lengths in such models are generally less than the models used in deep **RNN** models. Deep **RNNs** are preferred due to their ability to include longer time periods. Unlike **Fully Connected Neural Networks (FNNs)**, **RNNs** use internal memory to process incoming inputs. **RNNs** are used in the analysis of time series data in various fields (handwriting recognition, speech recognition, etc. As stated in the literature, **RNNs** are good at predicting the next character in the text, language translation applications, sequential data processing [45, 37].

**RNN** model architecture consists of different number of layers and different type of units in each layer. The main difference between **RNN** and **FNN** is that each **RNN** unit takes the current and previous input data at the same time. The output depends on the previous data in **RNN** model. The **RNNs** process input sequences one by one at any given time, during

their operation. In the units on the hidden layer, they hold information about the history of the input in the “state vector”. When the output of the units in the hidden layer is divided into different discrete time steps, the **RNNs** are converted into a **DMLP** [37]. In Figure 2, the information flow in the **RNN**’s hidden layer is divided into discrete times. The status of the node  $S$  at different times of  $t$  is shown as  $s_t$ , the input value  $x$  at different times is  $x_t$ , and the output value  $o$  at different times is shown as  $o_t$ . Parameter values ( $U, W, V$ ) are always used in the same step.

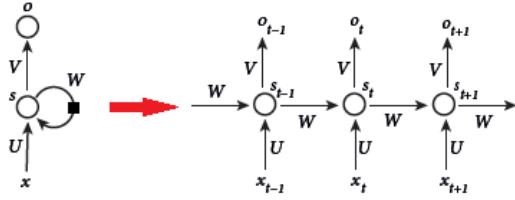


Figure 2: RNN cell through time[37]

**RNNs** can be trained using the **BPTT** algorithm. Optimization algorithms (**SGD**, **RMSProp**, **ADAM**) are used for weight adjustment process. With the **BPTT** learning method, the error change at any  $t$  time is reflected in the input and weights of the previous  $t$  times. The difficulty of training **RNN** is due to the fact that the **RNN** structure has a backward dependence over time. Therefore, **RNNs** become very complex in terms of the learning period. Although the main aim of using **RNN** is to learn long-term dependencies, studies in the literature show that when knowledge is stored for long time periods, it is not easy to learn with **RNN** (training difficulties on **RNN**) [57]. In order to solve this particular problem, **LSTMs** with different structures of **ANN** were developed [37]. Equations 8, 9 illustrate simpler **RNN** formulations. Equation 10 shows the total error which is the sum of each error at time step  $t$ <sup>1</sup>.

$$h_t = Wf(h_{t-1}) + W^{(hx)}x_{[t]} \quad (8)$$

$$y_t = W^{(S)}f(h_t) \quad (9)$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W} \quad (10)$$

Hyperparameters of **RNN** also define the network architecture and the performance of the network is affected by the parameter choices as was in **DMLP** case. The number of hidden layers, the number of units in each layer, regularization techniques, network weight initialization, activation functions, learning rate, momentum values, number of epochs, batch

---

<sup>1</sup>Richard Socher, CS224d: Deep Learning for Natural Language Processing, Lecture Notes

size (minibatch size), decay rate, optimization algorithms, model of **RNN** (Vanilla **RNN**, **Gated-Recurrent Unit (GRU)**, **LSTM**), sequence length for **RNN** are the hyperparameters of **RNN**. Finding the best hyperparameters for the network is a significant issue. In literature, there are different methods to find best hyperparameters: **MS**, **GS**, **RS**, Bayesian Methods (**SMBGO**, **GPA**, **TSPEA**) [55, 56].

### 3.3. Long Short Term Memory (**LSTM**)

**LSTM** [58] is a type of **RNN** where the network can remember both short term and long term values. **LSTM** networks are the preferred choice of many **DL** model developers when tackling complex problems like automatic speech recognition, and handwritten character recognition. **LSTM** models are mostly used with time-series data. It is used in different applications such as **Natural Language Processing (NLP)**, language modeling, language translation, speech recognition, sentiment analysis, predictive analysis, financial time series analysis, etc. [59, 60]. With attention modules and **AE** structures, **LSTM** networks can be more successful on time series data analysis such as language translation [59].

**LSTM** networks consist of **LSTM** units. Each **LSTM** unit merges to form an **LSTM** layer. An **LSTM** unit is composed of cells having input gate, output gate and forget gate. Three gates regulate the information flow. With these features, each cell remembers the desired values over arbitrary time intervals. Equations 11-15 show the form of the forward pass of the **LSTM** unit [58] ( $x_t$ : input vector to the **LSTM** unit,  $f_t$ : forget gate's activation vector,  $i_t$ : input gate's activation vector,  $o_t$ : output gate's activation vector,  $h_t$ : output vector of the **LSTM** unit,  $c_t$ : cell state vector,  $\sigma_g$ : sigmoid function,  $\sigma_c$ ,  $\sigma_h$ : hyperbolic tangent function,  $*$ : element-wise (Hadamard) product,  $W$ ,  $U$ : weight matrices that need to be learned,  $b$ : bias vector parameters that need to be learned) [60].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (11)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (12)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (13)$$

$$c_t = f_t * c_{t-1} + i_t * \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \quad (14)$$

$$h_t = o_t * \sigma_h(c_t) \quad (15)$$

**LSTM** is a specialized version of **RNN**. Therefore, the weight updates and preferred optimization methods are the same. In addition, the hyperparameters of **LSTM** are just like **RNN**: the number of hidden layers, the number of units in each layer, network weight initialization, activation functions, learning rate, momentum values, the number of epochs, batch size (minibatch size), decay rate, optimization algorithms, sequence length for **LSTM**, gradient clipping , gradient normalization, and dropout[61, 60]. In order to find the best hyperparameters of **LSTM**, the hyperparameter optimization methods that are used for **RNN** are also applicable to **LSTM** [55, 56].

### 3.4. Convolutional Neural Networks (CNNs)

**CNN** is a type of **DNN** that consists of convolutional layers that are based on the convolutional operation. Meanwhile, **CNN** is the most common model that is frequently used for vision or image processing based classification problems (image classification, object detection, image segmentation, etc.) [62, 63, 64]. The advantage of the usage of **CNN** is the number of parameters when comparing the vanilla **DL** models such as **DMLP**. Filtering with kernel window function gives an advantage of image processing to **CNN** architectures with fewer parameters that are beneficial for computing and storage. In **CNN** architectures, there are different layers: convolutional, max-pooling, dropout and fully connected **Multilayer Perceptron (MLP)** layer. The convolutional layer consists of the convolution (filtering) operation. Basic convolution operation is shown in Equation 16 ( $t$  denotes time,  $s$  denotes feature map,  $w$  denotes kernel,  $x$  denotes input,  $a$  denotes variable). In addition, the convolution operation is implemented on two-dimensional images. Equation 17 shows the convolution operation of two-dimensional image ( $I$  denotes input image,  $K$  denotes the kernel,  $m$  and  $n$  denote the dimension of images,  $i$  and  $j$  denote variables). Besides, consecutive convolutional and max-pooling layers construct the deep network. Equation 18 provides the details about the **NN** architecture ( $W$  denotes weights,  $x$  denotes input,  $b$  denotes bias,  $z$  denotes the output of neurons). At the end of the network, the softmax function is used to get the output. Equation 19 and 20 illustrate the softmax function ( $y$  denotes output) [39].

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \quad (16)$$

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n). \quad (17)$$

$$z_i = \sum_j W_{i,j} x_j + b_i. \quad (18)$$

$$y = \text{softmax}(z) \quad (19)$$

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (20)$$

The backpropagation process is used for model learning of **CNN**. Most commonly used optimization algorithms (**SGD**, **RMSProp**) are used to find optimum parameters of **CNN**. Hyperparameters of **CNN** are similar to other **DL** model hyperparameters: the number of hidden layers, the number of units in each layer, network weight initialization, activation functions, learning rate, momentum values, the number of epochs, batch size (minibatch size), decay rate, optimization algorithms, dropout, kernel size, and filter size. In order to find the best hyperparameters of **CNN**, usual search algorithms are used: **MS**, **GS**, **RS**, and Bayesian Methods. [55, 56].

### 3.5. Restricted Boltzmann Machines (RBMs)

**RBM** is a productive stochastic **ANN** that can learn probability distribution on the input set [65]. **RBM**s are mostly used for unsupervised learning [66]. **RBM**s are used in applications such as dimension reduction, classification, feature learning, collaborative filtering [67]. The advantage of the **RBM**s is to find hidden patterns with an unsupervised method. The disadvantage of **RBM**s is its difficult training process. “**RBM**s are tricky because although there are good estimators of the log-likelihood gradient, there are no known cheap ways of estimating the log-likelihood itself” [68].

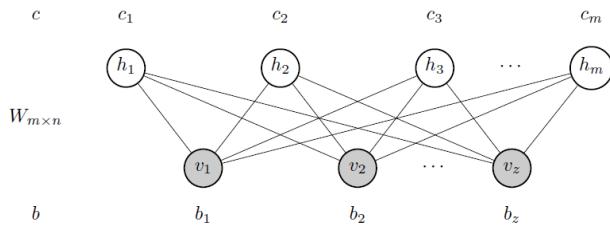


Figure 3: RBM Visible and Hidden Layers [65]

**RBM** is a two-layer, bipartite, and undirected graphical model that consists of two layers; visible and hidden layers (Figure 3). The layers are not connected among themselves. Each cell is a computational point that processes the input and makes stochastic decisions about whether this nerve node will transmit the input. Inputs are multiplied by specific weights, certain threshold values (bias) are added to input values, then calculated values are passed through an activation function. In reconstruction stage, the results in the outputs re-enter the network as the input, then they exit from the visible layer as the output. The values of the previous input and the values after the processes are compared. The purpose of the comparison is to reduce the difference.

Equation 21 illustrates the probabilistic semantics for an **RBM** by using its energy function ( $P$  denotes the probabilistic semantics for an **RBM**,  $Z$  denotes the partition function,  $E$  denotes the energy function,  $h$  denotes hidden units,  $v$  denotes visible units). Equation 22 illustrates the partition function or the normalizing constant. Equation 23 shows the energy of a configuration (in matrix notation) of the standard type of **RBM** that has binary-valued hidden and visible units ( $a$  denotes bias weights (offsets) for the visible units,  $b$  denotes bias weights for the hidden units,  $W$  denotes matrix weight of the connection between hidden and visible units,  $T$  denotes the transpose of matrix,  $v$  denotes visible units,  $h$  denotes hidden units) [69, 70].

$$P(v, h) = \frac{1}{Z} \exp(-E(v, h)) \quad (21)$$

$$Z = \sum_v \sum_h \exp(-E(v, h)) \quad (22)$$

$$E(v, h) = -a^T v - b^T h - v^T Wh \quad (23)$$

The learning is performed multiple times on the network [65]. The training of RBMs is implemented through minimizing the negative log-likelihood of the model and data. **Contrastive Divergence (CD)** algorithm is used for the stochastic approximation algorithm which replaces the model expectation for an estimation using Gibbs Sampling with a limited number of iterations [66]. In the **CD** algorithm, the **Kullback Leibler Divergence (KL-Divergence)** algorithm is used to measure the distance between its reconstructed probability distribution and the original probability distribution of the input [71].

Momentum, learning rate, weight-cost (decay rate), batch size (minibatch size), regularization method, the number of epochs, the number of layers, initialization of weights, size of visible units, size of hidden units, type of activation units (sigmoid, softmax, ReLU, Gaussian units, etc.), loss function, and optimization algorithms are the hyperparameters of RBMs. Similar to the other deep networks, the hyperparameters are searched with **MS**, **GS**, **RS**, and bayesian methods (Gaussian process). In addition to these, **Annealed Importance Sampling (AIS)** is used to estimate the partition function. **CD** algorithm is also used for the optimization of RBMs [55, 56, 72, 73].

### 3.6. Deep Belief Networks (DBNs)

**DBN** is a type of deep **ANN** and consists of a stack of **RBM** networks (Figure 4). **DBN** is a probabilistic generative model that consists of latent variables. In **DBN**, there is no link between units in each layer. **DBNs** are used to find discriminate independent features in the input set using unsupervised learning [69]. The ability to encode the higher-order network structures and fast inference are the advantages of the DBNs [74]. **DBNs** have disadvantages of training like **RBM**s which is mentioned in the **RBM** section, (**DBNs** are composed of **RBM**s).

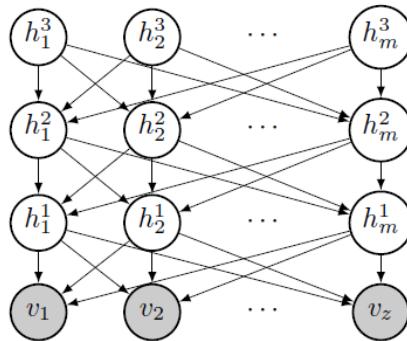


Figure 4: Deep Belief Network [65]

When **DBN** is trained on the training set in an unsupervised manner, it can learn to reconstruct the input set in a probabilistic way. Then the layers on the network begin to detect discriminating features in the input. After this learning step, supervised learning

is carried out to perform the classification [75]. Equation 24 illustrates the probability of generating a visible vector ( $W$ : matrix weight of connection between hidden unit  $h$  and visible unit  $v$ ,  $p(h|W)$ : the prior distribution over hidden vectors) [69].

$$p(v) = \sum_h p(h|W)p(v|h, W) \quad (24)$$

**DBN** training process can be divided into two steps: stacked **RBM** learning and backpropagation learning. In stacked **RBM** learning, iterative **CD** algorithm is used [66]. In backpropagation learning, optimization algorithms (**SGD**, **RMSProp**, **ADAM**) are used to train network [74]. **DBNs**' hyperparameters are similar to RBMs' hyperparameters. Momentum, learning rate, weight-cost (decay rate), regularization method, batch size (minibatch size), the number of epochs, the number of layers, initialization of weights, the number of **RBM** stacks, size of visible units in **RBM**s' layers, size of hidden units in **RBM**s' layer, type of units (sigmoid, softmax, rectified, Gaussian units, etc.), network weight initialization, and optimization algorithms are the hyperparameters of **DBNs**. Similar to the other deep networks, the hyperparameters are searched with **MS**, **GS**, **RS**, and Bayesian methods. **CD** algorithm is also used for the optimization of **DBNs** [55, 56, 72, 73].

### 3.7. Autoencoders (AEs)

**AE** networks are **ANN** types that are used as unsupervised learning models. In addition, **AE** networks are commonly used in **DL** models, wherein they remap the inputs (features) such that the inputs are more representative for classification. In other words, **AE** networks perform an unsupervised feature learning process, which fits very well with the **DL** theme. A representation of a data set is learned by reducing the dimensionality with **AEs**. **AEs** are similar to **Feedforward Neural Networks (FFNNs)**' architecture. They consist of an input layer, an output layer and one or more hidden layers that connect them together. The number of nodes in the input layer and the number of nodes in the output layer are equal to each other in **AEs**, and they have a symmetrical structure. The most notable advantages of **AEs** are dimensionality reduction and feature learning. Meanwhile, reducing dimensionality and feature extraction in **AEs** cause some drawbacks. Focusing on minimizing the loss of data relationship in encoding of **AE** cause the loss of some significant data relationships. Hence, this may be considered as a drawback of **AEs**[76].

In general, **AEs** contain two components: encoder and decoder. The input  $x \in [0, 1]^d$  is converted through function  $f(x)$  ( $W_1$  denotes a weight matrix of encoder,  $b_1$  denotes a bias vector of encoder,  $\sigma_1$  element-wise sigmoid activation function of encoder). Output  $h$  is the encoded part of **AEs** (code), latent variables, or latent representation. The inverse of function  $f(x)$ , called function  $g(h)$ , produces the reconstruction of output  $r$  ( $W_2$  denotes a weight matrix of decoder,  $b_2$  denotes a bias vector of decoder,  $\sigma_2$  element-wise sigmoid activation function of decoder). Equations 25 and 26 illustrate the simple **AE** process [77]. Equation 27 shows the loss function of the **AE**, the **Mean Squared Error (MSE)**. In the literature, **AEs** have been used for feature extraction and dimensionality reduction [39, 77].

$$h = f(x) = \sigma_1(W_1x + b_1) \quad (25)$$

$$r = g(h) = \sigma_2(W_2h + b_2) \quad (26)$$

$$L(x, r) = \|x - r\|^2 \quad (27)$$

**AEs** are a specialized version of **FFNNs**. The backpropagation learning is used for the update of the weights in the network[39]. Optimization algorithms (**SGD**, **RMSProp**, **ADAM**) are used for the learning process of **AEs**. **MSE** is used as a loss function in **AEs**. In addition, recirculation algorithms may also be used for the training of the **AEs** [39]. **AEs**' hyperparameters are similar to **DL** hyperparameters. Learning rate, weight-cost (decay rate), dropout fraction, batch size (minibatch size), the number of epochs, the number of layers, the number of nodes in each encoder layers, type of activation functions, number of nodes in each decoder layers, network weight initialization, optimization algorithms, and the number of nodes in the code layer (size of latent representation) are the hyperparameters of **AEs**. Similar to the other deep networks, the hyperparameters are searched with **MS**, **GS**, **RS**, and Bayesian methods [55, 56].

### 3.8. Deep Reinforcement Learning (DRL)

**Reinforcement learning (RL)** is a type of learning method that differs from supervised and unsupervised learning models. It does not need a preliminary data set which is labeled or clustered before. **RL** is an ML approach inspired by learning action/behavior, which deals with what actions should be taken by subjects to achieve the highest reward in an environment. There are different application areas that are used: game theory, control theory, multi-agent systems, operations research, robotics, information theory, managing investment portfolio, simulation-based optimization, playing Atari games, and statistics [78]. Some of the advantages of using **RL** for control problems are that an agent can be easily re-trained to adapt to changes in the environment and that the system is continually improved while training is constantly performed. An **RL** agent learns by interacting with its surroundings and observing the results of these interactions. This learning method mimics the basic way of how people learn.

**RL** is mainly based on **Markov Decision Process (MDP)**. **MDP** is used to formalize the **RL** environment. **MDP** consists of five tuples: state (finite set of states), action (finite set of actions), reward function (scalar feedback signal), state transition probability matrix ( $p(s', r|s, a)$ ,  $s'$  denotes next state,  $r$  denotes reward function,  $s$  denotes state,  $a$  denotes action), discount factor ( $\gamma$ , present value of future rewards). The aim of the agent is to maximize the cumulative reward. The return ( $G_t$ ) is the total discounted reward. Equation 28 illustrates the total return ( $G_t$  denotes total discounted reward,  $R$  denotes rewards,  $t$  denotes time,  $k$  denotes variable in time).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (28)$$

The value function is the prediction of the future values. It informs about how good is state/action. Equation 29 illustrates the formulation of the value function ( $v(s)$  denotes

the value function,  $E[.]$  denotes the expectation function,  $G_t$  denotes the total discounted reward,  $s$  denotes the given state,  $R$  denotes the rewards,  $S$  denotes the set of states,  $t$  denotes time).

$$v(s) = E[G_t|S_t = s] = E[R_{t+1} + \gamma v(S_{t+1})|S_t = s] \quad (29)$$

Policy ( $\pi$ ) is the agent's behavior strategy. It is like a map from state to action. There are two types of value functions to express the actions in the policy: state-value function ( $v_\pi(s)$ ), action-value function ( $q_\pi(s, a)$ ). The state-value function (Equation 30) is the expected return of starting from  $s$  to following policy  $\pi$  ( $E_\pi[.]$  denotes expectation function). The action-value function (Equation 31) is the expected return of starting from  $s$ , taking action  $a$  to following policy  $\pi$  ( $A$  denotes the set of actions,  $a$  denotes the given action).

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right] \quad (30)$$

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] \quad (31)$$

The optimal state-value function (Equation 32) is the maximum value function over all policies. The optimal action-value function (Equation 33) is the maximum action-value function over all policies.

$$v_*(s) = \max(v_\pi(s)) \quad (32)$$

$$q_*(s, a) = \max(q_\pi(s, a)) \quad (33)$$

The **RL** solutions and methods in the literature are too broad to review in this paper. So, we summarized the important issues of **RL**, important **RL** solutions and methods. **RL** methods are mainly divided into two sections: Model-based methods and model-free methods. The model-based method uses a model that is known by the agent before, value/policy and experience. The experience can be real (sample from the environment) or simulated (sample from the model). Model-based methods are mostly used in the application of robotics, and control algorithms [79]. Model-free methods are mainly divided into two groups: Value-based and policy-based methods. In value-based methods, a policy is produced directly from the value function (e.g. epsilon-greedy). In policy-based methods, the policy is parametrized directly. In value-based methods, there are three main solutions for **MDP** problems: **Dynamic Programming (DP)**, **Monte Carlo (MC)**, and **Temporal Difference (TD)**.

In **DP** method, problems are solved with optimal substructure and overlapping subproblems. The full model is known and it is used for planning in **MDP**. There are two iterations (learning algorithms) in **DP**: policy iteration and value iteration. **MC** method learns experience directly by running an episode of game/simulation. **MC** is a type of model-free method that does not need **MDP** transitions/rewards. It collects states, returns and it gets mean of returns for the value function. **TD** is also a model-free method that learns the experience directly by running the episode. In addition, **TD** learns incomplete episodes like the

DP method by using bootstrapping. TD method combines MC and DP methods. SARSA (state, action, reward, state, action;  $S_t, A_t, R_t, S_{t+1}, A_{t+1}$ ) is a type of TD control algorithm. Q-value (action-value function) is updated with the agent actions. It is an on-policy learning model that learns from actions according to the current policy  $\pi$ . Equation 34 illustrates the update of the action-value function in SARSA algorithm ( $S_t$  denotes current state,  $A_t$  denotes current action,  $t$  denotes time,  $R$  denotes reward,  $\alpha$  denotes learning rate,  $\gamma$  denotes discount factor). Q-learning is another TD control algorithm. It is an off-policy learning model that learns from different actions that do not need the policy  $\pi$  at all. Equation 35 illustrates the update of the action-value function in Q-Learning algorithm (The whole algorithms can be reached in [78],  $a'$  denotes action).

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R(t+1) + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (34)$$

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R(t+1) + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)] \quad (35)$$

In the value-based methods, a policy can be generated directly from the value function (e.g. using epsilon-greedy). The policy-based method uses the policy directly instead of using the value function. It has advantages and disadvantages over the value-based methods. The policy-based methods are more effective in high-dimensional or continuous action spaces, and have better convergence properties when compared against the value-based methods. It can also learn the stochastic policies. On the other hand, the policy-based method evaluates a policy that is typically inefficient and has high variance. It typically converges to a local rather than the global optimum. In the policy-based methods, there are also different solutions: Policy gradient, Reinforce (Monte-Carlo Policy Gradient), Actor-Critic [78] (Details of policy-based methods can be reached in [78]).

DRL methods contain NNs. Therefore, DRL hyperparameters are similar to DL hyperparameters. Learning rate, weight-cost (decay rate), dropout fraction, regularization method, batch size (minibatch size), the number of epochs, the number of layers, the number of nodes in each layer, type of activation functions, network weight initialization, optimization algorithms, discount factor, and the number of episodes are the hyperparameters of DRL. Similar to the other deep networks, the hyperparameters are searched with MS, GS, RS and bayesian methods [55, 56].

#### 4. Financial Time Series Forecasting

The most widely studied financial application area is forecasting of a given financial time series, in particular asset price forecasting. Even though some variations exist, the main focus is on predicting the next movement of the underlying asset. More than half of the existing implementations of DL were focused on this area. Even though there are several subtopics of this general problem including Stock price forecasting, Index prediction, forex price prediction, commodity (oil, gold, etc) price prediction, bond price forecasting, volatility forecasting, cryptocurrency price forecasting, the underlying dynamics are the same in all of these applications.

The studies can also be clustered into two main groups based on their expected outputs: price prediction and price movement (trend) prediction. Even though price forecasting is basically a regression problem, in most of the financial time series forecasting applications, correct prediction of the price is not perceived as important as correctly identifying the directional movement. As a result, researchers consider trend prediction, i.e. forecasting which way the price will change, a more crucial study area compared with exact price prediction. In that sense, trend prediction becomes a classification problem. In some studies, only up or down movements are taken into consideration (2-class problem), whereas up, down or neutral movements (3-class problem) also exist.

[LSTM](#) and its variations along with some hybrid models dominate the financial time series forecasting domain. [LSTM](#), by its nature utilizes the temporal characteristics of any time series signal, hence forecasting financial time series is a well-studied and successful implementation of [LSTM](#). However, some researchers prefer to either extract appropriate features from the time series or transform the time series in such a way that, the resulting financial data becomes stationary from a temporal perspective, meaning even if we shuffle the data order, we will still be able to properly train the model and achieve successful out-of-sample test performance. For those implementations, [CNN](#) and [Deep Feedforward Neural Network \(DFNN\)](#) were the most commonly chosen [DL](#) models.

Various financial time series forecasting implementations using [DL](#) models exist in literature. We will cover each of these aforementioned implementation areas in the following subsections. In this survey paper, we examined the papers using the following criteria:

- First, we grouped the articles according to their subjects.
- Then, we grouped the related papers according to their feature set.
- Finally, we grouped each subgroup according to [DL](#) models/methods.

For each implementation area, the related papers will be subgrouped and tabulated. Each table will have the following fields to provide the information about the implementation details for the papers within the group: Article (Art.) and Data Set are trivial, Period refers to the time period for training and testing. Feature Set lists the input features used in the study. Lag has the time length of the input vector (e.g. 30d means the input vector has a 30 day window) and horizon shows how far out into the future is predicted by the model. Some abbreviations are used for these two aforementioned fields: min is minutes, h is hours, d is days, w is weeks, m is months, y is years, s is steps, \* is mixed. Method shows the [DL](#) models that are used in the study. Performance criteria provides the evaluation metrics, and finally the Environment (Env.) lists the development framework/software/tools. Some column values might be empty, indicating there was no relevant information in the paper for the corresponding field.

#### 4.1. Stock Price Forecasting

Price prediction of any given stock is the most studied financial application of all. We observed the same trend within the [DL](#) implementations. Depending on the prediction time

horizon, different input parameters are chosen varying from High Frequency Trading (HFT) and intraday price movements to daily, weekly or even monthly stock close prices. Also, technical, fundamental analysis, social media feeds, sentiment, etc. are among the different parameters that are used for the prediction models.

Table 1: Stock Price Forecasting Using Only Raw Time Series Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[80]	38 stocks in KOSPI	2010-2014	Lagged stock returns	50min	5min	DNN	NMSE, RMSE, MAE, MI	-
[81]	China stock market, 3049 Stocks	1990-2015	OCHLV	30d	3d	LSTM	Accuracy	Theano, Keras
[82]	Daily returns of 'BRD' stock in Romanian Market	2001-2016	OCHLV	-	1d	LSTM	RMSE, MAE	Python, Theano
[83]	297 listed companies of CSE	2012-2013	OCHLV	2d	1d	LSTM, SRNN, GRU	MAD, MAPE	Keras
[84]	5 stock in NSE	1997-2016	OCHLV, Price data, turnover and number of trades.	200d	1..10d	LSTM, RNN, CNN, MLP	MAPE	-
[85]	Stocks of Infosys, TCS and CIPLA from NSE	2014	Price data	-	-	RNN, LSTM and CNN	Accuracy	-
[86]	10 stocks in S&P500	1997-2016	OCHLV, Price data	36m	1m	RNN, LSTM, GRU	Accuracy, Monthly return	Keras, Tensorflow
[87]	Stocks data from S&P500	2011-2016	OCHLV	1d	1d	DBN	MSE, norm-RMSE, MAE	-
[88]	High-frequency transaction data of the CSI300 futures	2017	Price data	-	1min	DNN, ELM, RBF	RMSE, MAPE, Accuracy	Matlab
[89]	Stocks in the S&P500	1990-2015	Price data	240d	1d	DNN, GBT, RF	Mean MDD, ratio	H2O
[90]	ACI Worldwide, Staples, and Seagate in NASDAQ	2006-2010	Daily closing prices	17d	1d	RNN, ANN	RMSE	-
[91]	Chinese Stocks	2007-2017	OCHLV	30d	1..5d	CNN + LSTM	Annualized Return, Mxm Retracement	Python
[92]	20 stocks in S&P500	2010-2015	Price data	-	-	AE + LSTM	Weekly Returns	-
[93]	S&P500	1985-2006	Monthly and daily log-returns	*	1d	DBN+MLP	Validation, Test Error	Theano, Python, Matlab
[94]	12 stocks from SSE Composite Index	2000-2017	OCHLV	60d	1..7d	DWNN	MSE	Tensorflow
[95]	50 stocks from NYSE	2007-2016	Price data	-	1d, 3d, 5d	SFM	MSE	-

In this survey, first, we grouped the stock price forecasting articles according to their feature set such as studies using only the raw time series data (price data, Open, Close, High, Low, Volume (OCHLV)) for price prediction; studies using various other data and papers that used text mining techniques. Regarding the first group, the corresponding DL models were directly implemented using the raw time series for price prediction. Table 1 tabulates the stock price forecasting papers that used only raw time series data in the literature. In Table 1, different methods/models are also listed based on four sub-groups: DNN (networks that are deep but without any given topology details) and LSTM models; multi models; hybrid models; novel methods.

**DNN** and **LSTM** models were solely used in 3 papers. In [80], **DNN** and lagged stock returns were used to predict the stock prices in **The Korea Composite Stock Price Index (KOSPI)**. Chen et. al. [81], Dezsi and Nistor [82] applied the raw price data as the input to **LSTM** models.

Meanwhile, there were some studies implementing multiple **DL** models for performance comparison using only the raw price (**OCHLV**) data for forecasting. Among the noteworthy studies, the authors in [83] compared **RNN**, **Stacked Recurrent Neural Network (SRNN)**, **LSTM** and **GRU**. Hiransha et. al. [84] compared **LSTM**, **RNN**, **CNN**, **MLP**, whereas in [85] **RNN**, **LSTM**, **CNN**, **Autoregressive Integrated Moving Average (ARIMA)** were preferred, Lee and Yoo [86] compared 3 **RNN** models (**SRNN**, **LSTM**, **GRU**) for stock price prediction and then constructed a threshold based portfolio with selecting stocks according to the predictions and Li et. al. [87] implemented **DBN**. Finally, the authors of [88] compared 4 different **ML** models (1 **DL** model - **AE** and **RBM**), **MLP**, **Radial Basis Function Neural Network (RBF)** and **Extreme Learning Machine (ELM)** for predicting the next price in 1-minute price data. They also compared the results with different sized datasets. The authors of [89] used price data and **DNN**, **Gradient Boosted Trees (GBT)**, **Random Forest (RF)** methods for the prediction of the stocks in the **Standard's & Poor's 500 Index (S&P500)**. In Chandra and Chan [90], co-operative neuro-evolution, **RNN** (Elman network) and **DFNN** were used for the prediction of stock prices in **National Association of Securities Dealers Automated Quotations (NASDAQ)** (ACI Worldwide, Staples, and Seagate).

Meanwhile, hybrid models were used in some of the papers. The author of [91] applied **CNN+LSTM** in their studies. Heaton et. al. [92] implemented smart indexing with **AE**. The authors of [93] combined **DBN** and **MLP** to construct a stock portfolio by predicting each stock's monthly log-return and choosing the only stocks that were expected to perform better than the performance of the median stock.

In addition, some novel approaches were adapted in some of the studies. The author of [94] proposed novel **Deep and Wide Neural Network (DWNN)** which is combination of **RNN** and **CNN**. The author of [95] implemented **State Frequency Memory (SFM)** recurrent network in their studies.

In another group of studies, some researchers again focused on **LSTM** based models. However, their input parameters came from various sources including the raw price data, technical and/or fundamental analysis, macroeconomic data, financial statements, news, investor sentiment, etc. Table 2 tabulates the stock price forecasting papers that used various data such as the raw price data, technical and/or fundamental analysis, macroeconomic data in the literature. In Table 2, different methods/models are also listed based on five sub-groups: **DNN** model; **LSTM** and **RNN** models; multiple and hybrid models; **CNN** model; novel methods.

**DNN** models were used in some of the stock price forecasting papers within this group. In [96], **DNN** model and 25 fundamental features were used for the prediction of the Japan Index constituents. Feng et. al. [97] also used fundamental features and **DNN** model for the prediction. **DNN** model, macro economic data such as GDP, unemployment rate, inventories, etc. were used by the authors of [98] for the prediction of the U.S. low-level disaggregated macroeconomic time series.

**LSTM** and **RNN** models were chosen in some of the studies. Kraus and Feuerriegel [99] implemented **LSTM** with transfer learning using text mining through financial news and the stock market data. Similarly, the author of [100] used **LSTM** to predict the stock's next day price using corporate action events and macro-economic index. Zhang and Tan [101] implemented DeepStockRanker, an **LSTM** based model for stock ranking using 11 technical indicators. In another study [102], the authors used the price time series and emotional data from text posts for predicting the stock opening price of the next day with **LSTM** network. Akita et. al. [103] used textual information and stock prices through Paragraph Vector + **LSTM** for forecasting the prices and the comparisons were provided with different classifiers. Ozbayoglu [104] used technical indicators along with the stock data on a Jordan-Elman network for price prediction.

There were also multiple and hybrid models that used mostly technical analysis features as their inputs to the **DL** model. Several technical indicators were fed into **LSTM** and **MLP** networks in [105] for predicting intraday price prediction. Recently, Zhou et. al. [106] used **GAN** for minimizing Forecast error loss and Direction prediction loss (**GAN-FD**) model for stock price prediction and compared their model performances against **ARIMA**, **ANN** and **Support Vector Machine (SVM)**. The authors of [107] used several technical indicator features and time series data with **Principal Component Analysis (PCA)** for dimensionality reduction cascaded with **DNN** (2-layer **FFNN**) for stock price prediction. In [108], the authors used Market microstructures based trade indicators as inputs into **RNN** with Graves **LSTM** detecting the buy-sell pressure of movements in **Istanbul Stock Exchange Index (BIST)** in order to perform the price prediction for intelligent stock trading. In [109], next month's return was predicted and top to be performed portfolios were constructed. Good monthly returns were achieved with **LSTM** and **LSTM-MLP** models.

Meanwhile, in some of the papers, **CNN** models were preferred. The authors of [110] used 250 features: order details, etc for the prediction of the private brokerage company's real data of risky transactions. They used **CNN** and **LSTM** for stock price forecasting. The authors of [111] used **CNN** model, fundamental, technical and market data for the prediction.

Novel methods were also developed in some of the studies. In [112], FI-2010 dataset: bid/ask and volume were used as the feature set for the forecast. In the study, they proposed **Weighted Multichannel Time-series Regression (WMTR)**, **Multilinear Discriminant Analysis (MDA)**. The authors of [113] used 57 characteristic features such as Market equity, Market Beta, Industry momentum, Asset growth, etc. as inputs to a Fama-French n-factor model **DL** for predicting monthly US equity returns in **New York Stock Exchange (NYSE)**, **American Stock Exchange (AMEX)**, or **NASDAQ**.

Table 2: Stock Price Forecasting Using Various Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[96]	Japan Index constituents from WorldScope	1990-2016	25 Fundamental Features	10d	1d	<b>DNN</b>	Correlation, Accuracy, <b>MSE</b>	Tensorflow
[97]	Return of <b>S&amp;P500</b>	1926-2016	Fundamental Features:	-	1s	<b>DNN</b>	<b>MSPE</b>	Tensorflow

Table 2: Stock Price Forecasting Using Various Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[98]	U.S. low-level disaggregated macroeconomic time series	1959-2008	GDP, Unemployment rate, Inventories, etc.	-	-	DNN	R <sup>2</sup>	-
[99]	CDEX stock market data	2010-2013	Financial news, stock market data	20d	1d	LSTM	MSE, MAE, AUC, RMSE, Accuracy	TensorFlow, Theano, Python, Scikit-Learn
[100]	Stock of Tsugami Corporation	2013	Price data	-	-	LSTM	RMSE	Keras, Tensorflow
[101]	Stocks in China's A-share	2006-2007	11 technical indicators	-	1d	LSTM	AR, IR, IC	-
[102]	SCI prices	2008-2015	OCHL of change rate, price	7d	-	EmotionalAnalysis + LSTM	MSE	-
[103]	10 stocks in Nikkei 225 and news	2001-2008	Textual information and Stock prices	10d	-	Paragraph Vector + LSTM	Profit	-
[104]	TKC stock in NYSE and QQQQ ETF	1999-2006	Technical indicators, Price	50d	1d	RNN (Jordan-Elman)	Profit, MSE	Java
[105]	10 Stocks in NYSE	-	Price data, Technical indicators	20min	1min	LSTM, MLP	RMSE	-
[106]	42 stocks in China's SSE	2016	OCHLV, Technical Indicators	242min	1min	GAN (LSTM, CNN)	RMSRE, DPA, GAN-F, GAN-D	-
[107]	Google's daily stock data	2004-2015	OCHLV, Technical indicators	20d	1d	(2D) <sup>2</sup> PCA + DNN	SMAPE, PCD, MAPE, RMSE, HR, TR, R <sup>2</sup>	R, Matlab
[108]	GarantiBank in BIST, Turkey	2016	OCHLV, Volatility, etc.	-	-	PLR, Graves LSTM	MSE, RMSE, MAE, RSE, R <sup>2</sup>	Spark
[109]	Stocks in NYSE, AMEX, NASDAQ, TAQ intraday trade	1993-2017	Price, 15 firm characteristics	80d	1d	LSTM+MLP	Monthly return, SR	Python, Keras, Tensorflow in AWS
[110]	Private brokerage company's real data of risky transactions	-	250 features: order details, etc.	-	-	CNN, LSTM	F1-Score	Keras, Tensorflow
[111]	Fundamental and Technical Data, Economic Data	-	Fundamental, technical and market information	-	-	CNN	-	-
[112]	The LOB of 5 stocks of Finnish Stock Market	2010	FI-2010 dataset: bid/ask and volume	-	*	WMTR, MDA	Accuracy, Precision, Recall, F1-Score	-
[113]	Returns in NYSE, AMEX, NASDAQ	1975-2017	57 firm characteristics	*	-	Fama-French n-factor model DL	R <sup>2</sup> , RMSE	Tensorflow

There were a number of research papers that also used text mining techniques for the feature extraction, but used non-LSTM models for the stock price prediction. Table 3 tabulates the stock price forecasting papers that used text mining techniques. In Table 3, different methods/models are clustered into three sub-groups: CNN and LSTM models; GRU, LSTM, and RNN models; novel methods.

CNN and LSTM models were adapted in some of the papers. In [114], events were detected from Reuters and Bloomberg news through text mining and that information was used for the price prediction and stock trading through the CNN model. Vargas et. al. [115] used text mining on S&P500 index news from Reuters through a LSTM+CNN hybrid model for price prediction and intraday directional movement estimation together. The authors of

[116] used the financial news data and implemented word embedding with Word2vec along with MA and stochastic oscillator to create inputs for Recurrent CNN (RCNN) for stock price prediction. The authors of [117] also used sentiment analyses through text mining and word embeddings from analyst reports and used sentiment features as inputs to DFNN model for stock price prediction. Then different portfolio selections were implemented based on the projected stock returns.

GRU, LSTM, and RNN models were preferred in the next group of papers. Das et. al. [118] implemented sentiment analysis on Twitter posts along with the stock data for price forecasting using RNN. Similarly, the authors of [119] used sentiment classification (neutral, positive, negative) for the stock open or close price prediction with various LSTM models. They compared their results with SVM and achieved higher overall performance. In [120], text and price data were used for the prediction of the SSE Composite Index (SCI) prices.

Some novel approaches were also found in some of the papers. The authors of [121] used word embeddings for extracting information from web pages and then combined with the stock price data for stock price prediction. They compared Autoregressive (AR) model and RF with and without news. The results showed embedding news information improved the performance. In [122], financial news and ACE2005 Chinese corpus were used. Different event-types on Chinese companies were classified based on a novel event-type pattern classification algorithm in [122], also next day stock price change was predicted using additional inputs.

Table 3: Stock Price Forecasting Using Text Mining Techniques for Feature Extraction

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[114]	S&P500 Index, stocks in S&P500	15	2006-2013 News from Reuters and Bloomberg	-	-	CNN	Accuracy, MCC	-
[115]	S&P500 index news from Reuters	2006-2013	Financial news titles, Technical indicators	1d	1d	RCNN	Accuracy	-
[116]	TWSE index, stocks in TWSE	4	2001-2017 Technical indicators, Price data, News	15d	-	CNN + LSTM	RMSE, Profit	Keras, Python, TALIB
[117]	Analyst reports on the TSE and Osaka Exchange	2016-2018	Text	-	-	LSTM, CNN, Bi-LSTM	Accuracy, squared	R, Python, MeCab
[118]	Stocks of Google, Microsoft and Apple	2016-2017	Twitter sentiment and stock prices	-	-	RNN	-	Spark, Flume, Twitter API,
[119]	Stocks of CSI300 index, OCHLV of CSI300 index	2009-2014	Sentiment Posts, Price data	1d	1d	Naive Bayes + LSTM	Precision, Recall, F1-score, Accuracy	Python, Keras
[120]	SCI prices	2013-2016	Text data and Price data	7d	1d	LSTM	Accuracy, F1-Measure	Python, Keras
[121]	Stocks from S&P500	2006-2013	Text (news) and Price data	7d	1d	LAR+News, RF+News	MAPE, RMSE	-
[122]	News from Sina.com, ACE2005 Chinese corpus	2012-2016	A set of news text	-	-	Their unique algorithm	Precision, Recall, F1-score	-

#### 4.2. Index Forecasting

Instead of trying to forecast the price of a single stock, several researchers preferred to predict the stock market index. Indices generally are less volatile than individual stocks, since they are composed of multiple stocks from different sectors and are more indicative of the overall momentum and general state of the economy.

In the literature, different stock market index data were used for the experiments. Most commonly used index data can be listed as follows: [S&P500](#), [China Securities Index \(CSI\)300](#), [National Stock Exchange of India \(NIFTY\)](#), [Tokyo Nikkei Index \(NIKKEI\)225](#), [Dow Jones Industrial Average \(DJIA\)](#), [Shanghai Stock Exchange \(SSE\)180](#), [Hong Kong Hang Seng Index \(HSI\)](#), [Shenzhen Stock Exchange Composite Index \(SZSE\)](#), [London Financial Times Stock Exchange Index \(FTSE\)100](#), [Taiwan Capitalization Weighted Stock Index \(TAIEX\)](#), [BIST](#), [NASDAQ](#), [Dow Jones Industrial Average 30 \(DOW30\)](#), [KOSPI](#), [S&P500 Volatility Index \(VIX\)](#), [NASDAQ100 Volatility Index \(VXN\)](#), [Brazilian Stock Exchange \(Bovespa\)](#), [Stockholm Stock Exchange \(OMX\)](#), [NYSE](#). The authors of [123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 114] used [S&P500](#) as their dataset. The authors of [123, 124, 135, 136, 137] used [NIKKEI](#) as their dataset. [KOSPI](#) was used in [135, 131, 132]. [DJIA](#) was used as the dataset in [123, 136, 137, 138, 139]. Besides, the authors of [123, 135, 137, 131] used [HSI](#) as the dataset in their studies. [SZSE](#) is used in studies of [140, 135, 141, 142].

In addition, in the literature, there were different methods for the prediction of the index data. While some of the studies used only the raw time series data, some others used various other data such as technical indicators, index data, social media feeds, news from Reuters, Bloomberg, the statistical features of data (standard deviation, skewness, kurtosis, omega ratio, fund alpha). In this survey, first, we grouped the index forecasting articles according to their feature set such as studies using only the raw time series data (price/index data, [OCHLV](#)); then in the second group we clustered the studies using various other data. Table 4 tabulates the index forecasting papers using only the raw time series data. Moreover, different methods (models) were used for index forecasting. [MLP](#), [RNN](#), [LSTM](#), [DNN](#) (most probably [DFNN](#), or [DMLP](#)) methods were the most used methods for index forecasting. In Table 4, these various methods/models are also listed as four sub-groups: [ANN](#), [DNN](#), [MLP](#), and [Fuzzy Deep Direct Reinforcement Learning \(FDDR\)](#) models; [RL](#) and [DL](#) models; [LSTM](#) and [RNN](#) models; novel methods.

Table 4: Index Forecasting Using Only Raw Time Series Data

Art.	Data Set	Period	Feature Set	Lag	Horizon Method	Performance Criteria	Env.
[124]	<a href="#">S&amp;P500</a> , Nikkei225, USD Exchanges	2011-2015	Index data	-	1d, 5d, 7d, 10d with Firefly-Harmony Search	<a href="#">LRNFIS</a>	RMSE, MAPE, -
[125]	<a href="#">S&amp;P500</a> Index	1989-2005	Index data, Volume	240d	1d	<a href="#">LSTM</a> Return, STD, SR, Accuracy	Python, Tensor-Flow, Keras, R, H2O
[127]	<a href="#">S&amp;P500</a> , <a href="#">VIX</a>	2005-2016	Index data	*	1d	uWN, cWN MASE, RMSE	HIT, -

Table 4: Index Forecasting Using Only Raw Time Series Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[128]	S&P500 Index	2010-2017	Index data	10d 30d	1d, 30d	Stacked LSTM, Bi-LSTM	MAE, RMSE, R-squared	Python, Keras, Tensorflow
[131]	S&P500, KOSPI, HSI, and EuroStoxx50	1987-2017	200-days stock price	200d	1d	Deep Q-Learning and DNN	Total profit, Correlation	-
[132]	S&P500, KOSPI200, 10-stocks	2000-2017	Index data	20d	1d	ModAugNet: LSTM	MSE, MAPE, MAE	Keras
[133]	S&P500, Bovespa50, OMX30	2009-2017	Autoregressive part of the time series	-	1d	LSTM	MSE, Accuracy	Tensorflow, Keras, R
[134]	S&P500	2000-2017	Index data	-	1..4d, 1w, 1..3m	GLM, LSTM+RNN	MAE, RMSE	Python
[136]	Nikkei225, IXIC, HSI, GSPC, DJIA	1985-2018	OCHLV	5d	1d	LSTM	RMSE	Python, Keras, Theano
[138]	DJIA	-	Index data	-	-	Genetic Deep Neural Network	MSE	Java
[139]	Log returns of the DJIA	1971-2002	Index data	20d	1d	RNN	TR, sign rate, PT/HM test, MSFE, SR, profit	-
[140]	Shanghai A-shares composite index, SZSE	2006-2016	OCHLV	10d	-	Embedded layer + LSTM	Accuracy, MSE	Python, Matlab, Theano
[141]	300 stocks from SZSE, Commodity	2014-2015	Index data	-	-	FDDR, DNN + RL	Profit, return, SR, profit-loss curves	Keras
[142]	Shanghai composite index and SZSE	1990-2016	OCHLV	20d	1d	Ensembles of ANN	Accuracy	-
[143]	TUNINDEX	2013-2017	Log returns of index data	-	5min	DNN with hierarchical input	Accuracy, MSE	Java
[144]	Singapore Stock Market Index	2010-2017	OCHL of last 10 days of index	10d	3d	Feed-forward DNN	RMSE, MAPE, Profit, SR	-
[145]	BIST	1990-2002	Index data	7d	1d	MLP, RNN, MoE	HIT, positive/negative HIT, MSE, MAE	-
[146]	SCI	2012-2017	OCHLV, Index data	-	1..10d	Wavelet + LSTM	MAPE, theil unequal coefficient	-
[147]	S&P500	1950-2016	Index data	15d	1d	LSTM	RMSE	Keras
[148]	ISE100	1987-2008	Index data	-	2d, 4d, 8d, 12d, 18d	TAR-VEC-MLP, TAR-VEC-RBF, TAR-VEC-RHE	RMSE	-
[149]	VIX, VXN, VXD	2002-2014	First five autoregressive lags	5d	1d, 22d	HAR-GASVR	MAE, RMSE	-

ANN, DNN, MLP, and FDDR models were used in some of the studies. In [143], log returns of the index data was used with DNN with hierarchical input for the prediction of the TUNINDEX data. The authors of [144] used deep FFNN and Open,Close,High, Low (OCHL) of the last 10 days of index data for prediction. In addition, MLP and ANN were used for the prediction of index data. In [145], the raw index data was used with MLP, RNN, Mixture of Experts (MoE) and Exponential GARCH (EGARCH) for the forecast. In [142], ensembles of ANN with OCHLV of the data were used for the prediction of the

Shanghai composite index.

Furthermore, **RL** and **DL** methods were used together for the prediction of the index data in some of the studies. In [141], **FDDR**, **DNN** and **RL** methods were used to predict 300 stocks from **SZSE** index data and commodity prices. In [131], Deep Q-Learning and **DNN** methods and 200-days stock price dataset were used together for the prediction of **S&P500** index.

Most of the preferred methods for prediction of the index data using the raw time series data were based on **LSTM** and **RNN**. In [139], **RNN** was used for prediction of the log returns of **DJIA** index. In [125], **LSTM** was used to predict **S&P500** Index data. The authors of [128] used stacked **LSTM**, **Bidirectional LSTM (Bi-LSTM)** methods for **S&P500** Index forecasting. The authors of [146] used **LSTM** network to predict the next day closing price of Shanghai stock Index. In their study, they used wavelet decomposition to reconstruct the financial time series for denoising and better learning. In [140], **LSTM** was used for the prediction of Shanghai A-shares composite index. The authors of [136] used **LSTM** to predict **NIKKEI225**, **IXIC**, **HIS**, **GSPC** and **DJIA** index data. In [147] and [132], **LSTM** was also used for the prediction of **S&P500** and **KOSPI200** index. The authors of [132] developed an **LSTM** based stock index forecasting model called ModAugNet. The proposed method was able to beat **Buy and Hold (B&H)** in the long term with an overfitting prevention mechanism. The authors of [134] compared different **ML** models (linear model), **Generalized Linear Model (GML)** and several **LSTM**, **RNN** models for stock index price prediction. In [133], **LSTM** and autoregressive part of the time series index data were used for prediction of **S&P500**, **Bovespa50**, **OMX30** indices.

Also, some studies adapted novel approaches. In [138], genetic **DNN** was used for **DJIA** index forecasting. The authors of [127] proposed a new **DNN** model which is called Wavenet convolutional net for time series forecasting. The authors of [148] proposed a (**Threshold Autoregressive (TAR)-Vector Error Correction model (VEC)-Recurrent Hybrid Elman (RHE)**) model for forex and stock index of return prediction and compared several models. The authors of [124] proposed a method that is called **Locally Recurrent Neuro-fuzzy Information System (LRNFIS)** with **Firefly Harmony Search Optimization (FHSO)** Evolutionary Algorithm (EA) to predict **S&P500**, **NIKKEI225** indices and USD Exchange price data. The authors of [149] proposed a **Heterogeneous Autoregressive Process (HAR)** with a **GA** with a **SVR (GASVR)** model that was called **HAR-GASVR** for prediction of **VIX**, **VXN**, Dow Jones Industrial Average Volatility Index (**VXD**) indices.

In the literature, some of the studies used various input data such as technical indicators, index data, social media news, news from Reuters, Bloomberg, the statistical features of data (standard deviation, skewness, kurtosis, omega ratio, fund alpha). Table 5 tabulates the index forecasting papers using these aforementioned various data. **DNN**, **RNN**, **LSTM**, **CNN** methods were the most commonly used models in index forecasting. In Table 5, different methods/models are also listed within four sub-groups: **DNN** model; **RNN** and **LSTM** models; **CNN** model; novel methods.

**DNN** was used as the classification model in some of the papers. In [150], **DNN** and some of the feature of the data (Return, **Sharpe-ratio (SR)**, **Standard Deviation (STD)**, Skewness, Kurtosis, Omega ratio, Fund alpha) were used for the prediction. In [126], **DNN**, **RNN** and

technical indicators were used for the prediction of **FTSE100**, **OMX30**, **S&P500** indices.

In addition, **RNN** and **LSTM** models with various other data were also used for the prediction of the indices. The authors of [137] used **RNN** and **OCHLV** of indices, technical indicators to predict **DJIA**, **FTSE**, **Nikkei**, **TAIEX** indices. The authors of [151] used **GASVR**, **LSTM** for the forecast. The authors of [152] used four **LSTM** models (technical analysis, attention mechanism and market vector embedded) for the prediction of the daily return ratio of **HSI300** index. In [135], **LSTM** with wavelet denoising and index data, volume, technical indicators were used for the prediction of the **HSI**, **SSE**, **SZSE**, **TAIEX**, **NIKKEI**, **KOSPI** indices. The authors of [153] used MODRL+**LSTM** method to predict Chinese stock-IF-IH-IC contract indices. The authors of [123] used stacked **AEs** to generate deep features using **OCHL** of the stock prices, technical indicators and macroeconomic conditions to feed to **LSTM** to predict the future stock prices.

Table 5: Index Forecasting Using Various Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[114]	<b>S&amp;P500</b> Index, 15 stocks in <b>S&amp;P500</b>	2006-2013	News from Reuters and Bloomberg	-	-	<b>CNN</b>	Accuracy, <b>MCC</b>	-
[116]	<b>TWSE</b> index, 4 stocks in <b>TWSE</b>	2001-2017	Technical indicators, Index data, News	15d	-	<b>CNN</b> + <b>LSTM</b>	<b>RMSE</b> , Profit	Keras, Python, <b>TALIB</b>
[123]	<b>CSI300</b> , <b>NIFTY50</b> , <b>HSI</b> , <b>NIKKEI225</b> , <b>S&amp;P500</b> , <b>DJIA</b>	2010-2016	<b>OCHLV</b> , Technical Indicators	-	1d	<b>WT</b> , Stacked autoencoders, <b>LSTM</b>	MAPE, Correlation coefficient, <b>THEIL-U</b>	-
[126]	FTSE100, OMXS 30, SP500, Commodity, Forex	1993-2017	Technical indicators	60d	1d	<b>DNN</b> , <b>RNN</b>	Accuracy, p-value	-
[129]	<b>S&amp;P500</b> , <b>DOW30</b> , <b>NASDAQ100</b> , Commodity, Forex, Bitcoin	2003-2016	Index data, Technical indicators	-	1w, 1m	<b>CNN</b>	Accuracy	Tensorflow
[130]	<b>BSE</b> , <b>S&amp;P500</b>	2004-2012	Index data, technical indicators	5d	1d..1m	<b>PSO</b> , <b>HMRPSO</b> , <b>DE</b> , <b>RCEFLANN</b>	<b>RMSE</b> , <b>MAPE</b>	-
[135]	<b>HSI</b> , <b>SSE</b> , <b>SZSE</b> , <b>TAIEX</b> , <b>NIKKEI</b> , <b>KOSPI</b>	2010-2016	Index data, volume, technical indicators	2d..512d	1d	<b>LSTM</b> with wavelet denoising	Accuracy, <b>MAPE</b>	-
[137]	<b>DJIA</b> , <b>FTSE</b> , <b>NIKKEI</b> , <b>TAIEX</b>	1997-2008	<b>OCHLV</b> , Technical indicators	26d	1d	<b>RNN</b>	<b>RMSE</b> , <b>MAE</b> , <b>THEIL-U</b>	<b>C</b>
[150]	Hedge fund monthly return data	1996-2015	Return, <b>SR</b> , <b>STD</b> , Skewness, Kurtosis, Omega ratio, Fund alpha	12m	3m, 6m, 12m	<b>DNN</b>	Sharpe ratio, Annual return, Cum. return	-
[151]	Stock of National Bank of Greece (ETE).	2009-2014	<b>FTSE100</b> , <b>DJIA</b> , <b>GDAX</b> , <b>NIKKEI225</b> , EUR/USD, Gold	1d, 2d, 5d, 10d	1d	<b>GASVR</b> , <b>LSTM</b>	Return, volatility, <b>SR</b> , Accuracy	Tensorflow
[152]	Daily return ratio of <b>HS300</b> index	2004-2018	<b>OCHLV</b> , Technical indicators	-	-	Market Vector + Tech. ind. + <b>LSTM</b> + Attention	<b>MSE</b> , <b>MAE</b>	Python, Tensorflow
[153]	Chinese stock-IF-IH-IC contract	2016-2017	Decisions for index change	240min	1min	<b>MODRL+LSTM</b>	Profit and loss, <b>SR</b>	-

Table 5: Index Forecasting Using Various Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[154]	HS300	2015-2017	Social media news, Index data	1d	1d	RNN-Boost with LDA	Accuracy, MAE, MAPE, RMSE	Python, Scikit-learn

Besides, different **CNN** implementations with various data (technical indicators, news, index data) were used in the literature. In [129], **CNN** and index data, technical indicators were used for the **S&P500**, **DOW30**, **NASDAQ100** indices and Commodity, Forex, Bitcoin prices. In [114], **CNN** model with news from Reuters and Bloomberg were used for the prediction of **S&P500** Index and 15 stocks' prices in **S&P500**. In [116], **CNN + LSTM** and technical indicators, index data, news were used for the forecasting of **Taiwan Stock Exchange (TWSE)** index and 4 stocks' prices in **TWSE**.

In addition, there were some novel methods proposed for the index forecasting. The authors of [130] used **RNN** models, **Recurrent Computationally Efficient Functional Link Neural Network (RCEFLANN)** and **Functional Link Neural network (FLANN)**, with their weights optimized using various **EA** like **Particle Swarm Optimization (PSO)**, HMRPSO and **PSO** for time series forecasting. The authors of [154] used social media news to predict the index price and index direction with **RNN-Boost with Latent Dirichlet Allocation (LDA)** features.

#### 4.3. Commodity Price Forecasting

There were a number of studies particularly focused on the price prediction of any given commodity, such as gold, silver, oil, copper, etc. With increasing number of commodities that are available for public trading through online stock exchanges, interest in this topic will likely grow in the following years.

In the literature, there were different methods that were used for commodity price forecasting. **DNN**, **RNN**, **FDDR**, **CNN** were the most used models to predict the commodity prices. Table 6 provides the details about the commodity price forecasting studies with **DL**.

In [129], the authors used **CNN** for predicting the next week and next month price directional movement. Meanwhile, **RNN** and **LSTM** models were used in some of the commodity forecasting studies. In [155], **DNN** was used for Commodity forecasting. In [126], different datasets (Commodity, forex, index) were used as datasets. **DNN** and **RNN** were used to predict the prices of the time series data. Technical indicators were used as the feature set which consist of **Relative Strength Index (RSI)**, **Williams Percent Range (William%R)**, **Commodity Channel Index (CCI)**, **Percentage Price Oscillator (PPOS)**, momentum, **Exponential Moving Average (EMA)**. In [156], the authors used Elman **RNN** to predict COMEX copper spot price (through **New York Mercantile Exchange (NYMEX)**) from daily close prices.

Hybrid and novel models were adapted in some studies. In [157], **FNN** and **Stacked Denoising Autoencoders (SDAE)** deep models were compared against **Support Vector Regressor (SVR)**, **Random Walk (RW)** and **Markov Regime Switching (MRS)** models for WTI oil price forecasting. As performance criteria, accuracy, **Mean Absolute Percentage Error (MAPE)**,

Root Mean Square Error (RMSE) were used. In [158], authors tried to predict WTI crude oil prices using several models including combinations of DBN, LSTM, Autoregressive Moving Average (ARMA) and RW. MSE was used as the performance criteria. In [141], the authors used FDDR for stock price prediction and trading signal generation. They combined DNN and RL. Profit, return, SR, profit-loss curves were used as the performance criteria.

Table 6: Commodity Price Forecasting

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[129]	S&P500, DOW30, NASDAQ100, Commodity, Forex, Bitcoin	2003-2016	Price data, Technical indicators	-	1w, 1m	CNN	Accuracy	Tensorflow
[155]	Commodity, FX future, ETF	1991-2014	Price Data	100*5min	5min	DNN	SR, capability ratio, return	C++, Python
[126]	FTSE100, OMX30, S&P500, Commodity, Forex	1993-2017	Technical indicators	60d	1d	DNN, RNN	Accuracy, p-value	-
[156]	Copper prices from NYMEX	2002-2014	Price data	-	-	Elman RNN	RMSE	R
[157]	WTI crude oil price	1986-2016	Price data	1m	1m	SDAE, Bootstrap aggregation	Accuracy, MAPE, RMSE	Matlab
[158]	WTI Crude Oil Prices	2007-2017	Price data	-	-	ARMA + DBN, RW + LSTM	MSE	Python, Keras, Tensorflow
[141]	300 stocks from SZSE, Commodity	2014-2015	Price data	-	-	FDDR, DNN + RL	Profit, return, SR, profit-loss curves	Keras

#### 4.4. Volatility Forecasting

Volatility is directly related with the price variations in a given time period and is mostly used for risk assesment and asset pricing. Some researchers implemented models for accurately forecasting the underlying volatility of any given asset.

In the literature, there were different methods that were used for volatility forecasting. LSTM, RNN, CNN, MM, Generalised Auto-Regressive Conditional Heteroscedasticity (GARCH) models were shown as some of these methods. Table 7 summarizes the studies that were focused on volatility forecasting. In Table 7, different methods/models are also represented as three sub-groups: CNN model; RNN and LSTM models; hybrid and novel models.

CNN model was used in one volatility forecasting study based on HFT data [159].

Meanwhile, RNN and LSTM models were used in some of the researches. In [160], the authors used financial time series data to predict volatility changes with Markov Models and Elman RNN for profitable straddle options trading. The authors of [161] used the price data and different types of Google Domestic trends with LSTM. The authors of [162] used CSI300, 28 words of the daily search volume based on Baidu as the dataset with LSTM to predict the index volatility. The authors of [163] developed several LSTM models integrated with GARCH for the prediction of volatility.

Hybrid and novel approaches were also adapted in some of the researches. In [164], RMDN with a GARCH (RMDN-GARCH) model was proposed. In addition, several models

including traditional forecasting models and **DL** models were compared for the estimation of volatility. The authors of [149] proposed a novel method that is called **HAR with a GASVR (HAR-GASVR)** for volatility index forecasting.

Table 7: Volatility Forecasting

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[159]	London Stock Exchange	2007-2008	Limit order book state, trades, buy/sell orders, order deletions	-	-	CNN	Accuracy, kappa	Caffe
[160]	DAX, FTSE100, call/put options	1991-1998	Price data	*	*	MM, RNN	Ewa-measure, iv, daily profits' mean and std	-
[161]	S&P500	2004-2015	Price data, Google Domestic trend dimensions	25	1d	LSTM	MAPE, RMSE	-
[162]	CSI 300, 28 words of the daily search volume based on Baidu	2006-2017	Price data and text	5d	5d	LSTM	MSE, MAPE	Python, Keras
[163]	KOSPI200, Korea Treasury Bond interest rate, AA-grade corporate bond interest rate, gold, crude oil	2001-2011	Price data	22d	1d	LSTM GARCH	+ MAE, MSE, HMAE, HMSE	-
[164]	DEM/GBP exchange rate	-	Returns	-	-	RMDN-GARCH	NMSE, NMAE, HR, WHR	-
[149]	VIX, VZN, VXD	2002-2014	First five autoregressive lags	5d	1d, 22d	HAR-GASVR	MAE, RMSE	-

#### 4.5. Bond Price Forecasting

Some financial experts follow the changes in the bond prices to analyze the state of the economy, claiming bond prices represent the health of the economy better than the stock market [165]. Historically, long term rates are higher than the short term rates under normal economic expansion times, whereas just before recessions short term rates pass the long term rates, i.e. the inverted yield curve. Hence, accurate bond price prediction is very useful. However, **DL** implementations for bond price prediction is very scarce. In one study [166], excess bond return was predicted using several **ML** models including **RF**, **AE** and **PCA** network and a 2-3-4-layer **DFNN**. 4 layer **NN** outperformed the other models.

#### 4.6. Forex Price Forecasting

Foreign exchange market has the highest volume among all existing financial markets in the world. It is open 24/7 and trillions of dollars worth of foreign exchange transactions happen in a single day. According to the Bank for International Settlements, foreign-exchange trading had a volume of more than 5 trillion USD a day [167]. In addition, there are a large number of online forex trading platforms that provide leveraged transaction opportunities to their subscribers. As a result, there is a huge interest for profitable trading strategies by traders. Hence, there were a number of forex forecasting and trading studies that were based on **DL** models. Since most of the global financial transactions were based on US

Dollar, almost all forex prediction research papers include USD in their analyses. However, depending on regional differences and intended research focus, various models were developed accordingly.

In the literature, there were different methods that were used for forex price forecasting. RNN, LSTM, CNN, DBN, DNN, AE, MLP methods were shown as some of these methods. Table 8 provides details about these implementations. In Table 8, different methods/models are listed as four sub-groups: Continuous-valued Deep Belief Networks (CDBN), DBN, DBN+RBM, and AE models; DNN, RNN, Psi-Sigma Network (PSN), and LSTM models; CNN models; hybrid models.

CDBN, DBN, DBN+RBM, and AE models were used in some of the studies. In [168], Fuzzy information granulation integrated with CDBN was applied for predicting EUR/USD and GBU/USD exchange rates. They extended DBN with Continuous Restricted Boltzman machine (CRBM) to improve the performance. In [169], weekly GBP/USD and INR/USD prices were predicted, whereas in [170], CNY/USD and INR/USD was the main focus. In both cases, DBN was compared with FFNN. Similarly, the authors in [171] implemented several different DBN networks to predict weekly GBP/USD, BRL/USD and INR/USD exchange rate returns. The researchers in [172] combined Stacked AE and SVR for predicting 28 normalized currency pairs using the time series data of (USD, GBP, EUR, JPY, AUD, CAD, CHF).

DNN, RNN, PSN, and LSTM models were preferred in some of the researches. In [155], multiple DMLP models were developed for predicting AD and BP futures using 5-minute data in a 130 day period. The authors of [173] used MLP, RNN, GP and other ML techniques along with traditional regression methods for also predicting EUR/USD time series. They also integrated Kalman filter, LASSO operator and other models to further improve the results in [174]. They further extended their analyses by including PSN and providing comparisons along with traditional forecasters like ARIMA, RW and STAR [175]. To improve the performance they also integrated hybrid time-varying volatility leverage. In [176], the authors implemented RMB exchange rate forecasting against JPY, HKB, EUR and USD by comparing RW, RNN and FFNN performances. In [177], the authors predicted various Forex time series and created portfolios consisted of these investments. Each network used LSTM (RNN EVOLINO) and different risk appetites for users have been tested. The authors of [178] also used EVOLINO RNN + orthogonal input data for predicting USD/JPY and XAU/USD prices for different periods.

Different CNN models were used in some of the studies. In [179], EUR/USD was once again forecasted using multiple DL models including MLP, CNN, RNN and Wavelet+CNN. The authors of [180] implemented forex trading (GBP/PLN) using several different input parameters on a multi-agent based trading environment. One of the agents was using AE+CNN as the prediction model and outperformed all other models.

Hybrid models were also adapted in some of the researches. The authors of [148] developed several (TAR-VEC-RHE) models for predicting monthly returns for TRY/USD and compared model performances. In [164], the authors compared several models including traditional forecasting models and DL models for DEM/GBP prediction. The authors in [124] predicted AUD, CHF, MAX and BRL against USD currency time series data using

LRNFIS and compared it with different models. Meanwhile, instead of using LMS based error minimization during the learning, they used **FHSO**.

Table 8: Forex Price Forecasting

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[168]	EUR/USD, GBP/USD	2009-2012	Price data	*	1d	<b>CDBN-FG</b>	Profit	-
[169]	GBP/USD, INR/USD	1976-2003	Price data	10w	1w	<b>DBN</b>	RMSE, MAPE, DA, PCC	-
[170]	CNY/USD,INR/USD	1997-2016	Price data	-	1w	<b>DBN</b>	MAPE, squared	R- -
[171]	GBP/USD, BRL/USD, INR/USD	1976-2003	Price data	10w	1w	<b>DBN + RBM</b>	RMSE, MAE, MAPE, accuracy, PCC	-
[172]	Combination of USD, GBP, EUR, JPY, AUD, CAD, CHF	2009-2016	Price data	-	-	Stacked + SVR	MAE, MSE, RMSE	Matlab
[155]	Commodity, FX fu- ture, <b>ETF</b>	1991-2014	Price Data	100*5min	5min	<b>DNN</b>	SR, capability ra- tio, return	C++, Python
[126]	FTSE100, OMX30, <b>S&amp;P500</b> , Commodity, Forex	1993-2017	Technical indica- tors	60d	1d	<b>DNN, RNN</b>	Accuracy, p-value	-
[173]	EUR/USD	2001-2010	Close data	11d	1d	RNN and more	MAE, MAPE, RMSE, THEIL-U	-
[174]	EUR/USD	2002-2010	Price data	13d	1d	RNN, MLP, PSN	MAE, MAPE, RMSE, THEIL-U	-
[175]	EUR/USD, EUR/GBP, EUR/JPY, EUR/CHF	1999-2012	Price data	12d	1d	RNN, MLP, PSN	MAE, MAPE, RMSE, THEIL-U	-
[176]	RMB against USD, EUR, JPY, HKD	2006-2008	Price data	10d	1d	<b>RNN, ANN</b>	RMSE, MAE, MSE	-
[177]	EUR/USD, EUR/JPY, USD/JPY, EUR/CHF, XAU/USD, XAG/USD, QM, QG	2011-2012	Price data	-	-	Evolino <b>RNN</b>	Correlation be- tween predicted, real values	-
[178]	USD/JPY	2009-2010	Price data, Gold	-	5d	EVOLINO RNN + orthogonal input data	RMSE	-
[179]	<b>S&amp;P500</b> , EUR/USD	1950-2016	Price data	30d, 30d*min	1d, 1min	Wavelet+ <b>CNN</b>	Accuracy, log- loss	Keras
[180]	USD/GBP, <b>S&amp;P500</b> , FTSE100, oil, gold	2016	Price data	-	5min	<b>AE + CNN</b>	SR, % volatility, avg return/trans, rate of return	H2O
[148]	ISE100, TRY/USD	1987-2008	Price data	-	2d, 4d, 8d, 12d, 18d	TAR-VEC- MLP, TAR- VEC-RBF, TAR-VEC- RHE	RMSE	-
[164]	DEM/GBP change rate	ex-	Returns	-	-	RMDN- GARCH	NMSE, NMAE, HR, WHR	-
[124]	<b>S&amp;P500</b> , NIKKEI225, Exchanges	USD	Price data	-	1d, 5d, 7d, 10d	LRNFIS with <b>FHSO</b>	RMSE, MAPE, MAE	-

#### 4.7. Cryptocurrency Price Forecasting

Since cryptocurrencies became a hot topic for discussion in the finance world, lots of studies and implementations started emerging in recent years. Most of the cryptocurrency studies were focused on price forecasting.

The rise of bitcoin from 1000 USD in January 2017 to 20,000 USD in January 2018 has attracted a lot of attention not only from the financial world, but also from ordinary people on the street. Recently, some papers have been published for price prediction and trading strategy development for bitcoin and other cryptocurrencies. Given the attention that the underlying technology has attracted, there is a great chance that some new studies will start appearing in the near future.

In the literature, **DNN**, **LSTM**, **GRU**, **RNN**, Classical methods (**ARMA**, **ARIMA**, **Autoregressive Conditional Heteroscedasticity (ARCH)**, **GARCH**, etc) were used for cryptocurrency price forecasting. Table 9 tabulates the studies that utilize these methods. In [181], the author combined the opinion market and price prediction for cryptocurrency trading. Text mining combined with 2 models **CNN** and **LSTM** were used to extract the opinion. Bitcoin, Litecoin, StockTwits were used as the dataset. **OCHLV** of prices, technical indicators, and sentiment analysis were used as the feature set. In [182], the authors compared Bayesian optimized **RNN**, **LSTM** and **ARIMA** to predict bitcoin price direction. Sensitivity, specificity, precision, accuracy, RMSE were used as the performance metrics.

Table 9: Cryptocurrency Price Prediction

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[181]	Bitcoin, Litecoin, StockTwits	2015-2018	<b>OCHLV</b> , technical indicators, sentiment analysis	-	30min, 4h, 1d	<b>CNN</b> , <b>LSTM</b> , State Frequency Model	<b>MSE</b>	Keras, Tensorflow
[182]	Bitcoin	2013-2016	Price data	100d	30d	Bayesian optimized <b>RNN</b> , <b>LSTM</b>	Sensitivity, specificity, precision, accuracy, <b>RMSE</b>	Keras, Python, Hyperas

#### 4.8. Trend Forecasting

Even though trend forecasting and price forecasting share the same input characteristics, some researchers prefer to predict the price direction of the asset instead of the actual price. This alters the nature of the problem from regression to classification and the corresponding performance metrics also change. However, it is worth to mention that these two approaches are not really different, the difference is in the interpretation of the output.

In the literature, there were different methods for trend forecasting. In this survey, we grouped the articles according to their feature set such as studies using only the raw time series data (only price data, **OCHLV**); studies using technical indicators & price data & fundamental data at the same time; studies using text mining techniques and studies using other various data. Table 10 tabulates the trend forecasting using only the raw time series data.

Table 10: Trend Forecasting Using Only Raw Time Series Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[183]	S&P500 stock indices	1963-2016	Price data	30d	1d	NN	Accuracy, precision, recall, F1-score, AUROC	R, H2o, Python, Tensorflow
[184]	SPY ETF, 10 stocks from S&P500	2014-2016	Price data	60min	30min	FNN	Cumulative gain	MatConvNet, Matlab
[142]	Shanghai composite index and SZSE	1990-2016	OCHLV	20d	1d	Ensembles of ANN	Accuracy	-
[185]	10 stocks from S&P500	-	Price data			TDNN, RNN, PNN	Missed opportunities, false alarms ratio	-
[186]	GOOGL stock daily price data	2012-2016	Time window of 30 days of OCHLV	22d, 50d, 70d	*	LSTM, GRU, RNN	Accuracy, Logloss	Python, Keras
[133]	S&P500, Bovespa50, OMX30	2009-2017	Autoregressive part of the price data	30d	1..15d	LSTM	MSE, Accuracy	Tensorflow, Keras, R
[187]	HSI, DAX, S&P500	1991-2017	Price data	-	1d	GRU, GRU-SVM	Daily return %	Python, Tensorflow
[188]	Taiwan Stock Index Futures	2001-2015	OCHLV	240d	1..2d	CNN with GAF, MAM, Candlestick	Accuracy	Matlab
[189]	ETF and Dow30	1997-2007	Price data			CNN with feature imaging	Annualized return	Keras, Tensorflow
[190]	SSEC, NASDAQ, S&P500	2007-2016	Price data	20min	7min	EMD2FNN	MAE, MAPE	RMSE, -
[191]	23 cap stocks from the OMX30 index in Nasdaq Stockholm	2000-2017	Price data and returns	30d	*	DBN	MAE	Python, Theano

Different methods and models were used for trend forecasting. In Table 10, these are divided into three sub-groups: ANN, DNN, and FFNN models; LSTM, RNN, and Probabilistic NN models; novel methods. ANN, DNN, DFNN, and FFNN methods were used in some of the studies. In [183], NN with the price data were used for prediction of the trend of S&P500 stock indices. The authors of [184] combined deep FNN with a selective trading strategy unit to predict the next price. The authors of [142] created an ensemble network of several Backpropagation and ADAM models for trend prediction.

In the literature, LSTM, RNN, Probabilistic Neural Network (PNN) methods with the raw time series data were also used for trend forecasting. In [185], the authors compared Timedelay Neural Network (TDNN), RNN and PNN for trend detection using 10 stocks from S&P500. The authors of [186] compared 3 different RNN models (basic RNN, LSTM, GRU) to predict the movement of Google stock price. The authors of [133] used LSTM (and other classical forecasting techniques) to predict the trend of the stocks prices. In [187], GRU and GRU-SVM models were used for the trend of HSI, The Deutscher Aktienindex (DAX), S&P500 indices.

There were also novel methods that used only the raw time series price/index data in the literature. The author of [188] proposed a method that used CNN with Gramian Angular Field (GAF), Moving Average Mapping (MAM), Candlestick with converted image data. In [189], a novel method, CNN with feature imaging was proposed for the prediction of the buy/sell/hold positions of the Exchange-Traded Funds (ETFs)' prices and Dow30 stocks'

prices. The authors of [190] proposed a method that uses **Empirical Mode Decomposition and Factorization Machine based Neural Network (EMD2FNN)** models to forecast the stock close prices' direction accurately. In [191], **DBN** with the price data were used for the prediction of the trend of 23 large cap stocks from the **OMX30** index.

Table 11: Trend Forecasting Using Technical Indicators & Price Data & Fundamental Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[192]	KSE100 index	-	Price data, several fundamental data	-	-	ANN, SLP, MLP, RBF, DBN, SVM	Accuracy	-
[193]	Stocks in Dow30	1997-2017	RSI (Technical Indicators)	200d	1d	DMLP with genetic algorithm	Annualized return	Spark MLlib, Java
[194]	SSE Composite Index, FTSE100, PingAnBank	1999-2016	Technical indicators, OCHLV price	24d	1d	RBM	Accuracy	-
[195]	Dow30 stocks	2012-2016	Price data, several technical indicators	40d	-	LSTM	Accuracy	Python, Keras, Tensorflow, TALIB
[196]	Stock price from IBOVESPA index	2008-2015	Technical indicators, OCHLV of price	-	15min	LSTM	Accuracy, Precision, Recall, F1-score, % return, Maximum drawdown	Keras
[197]	20 stocks from NASDAQ and NYSE	2010-2017	Price data, technical indicators	5d	1d	LSTM, GRU, SVM, XG-Boost	Accuracy	Keras, Tensorflow, Python
[198]	17 ETF	2000-2016	Price data, technical indicators	28d	1d	CNN	Accuracy, MSE, Profit, AUROC	Keras, Tensorflow
[199]	Stocks in Dow30 and 9 Top Volume ETF	1997-2017	Price data, technical indicators	20d	1d	CNN with feature imaging	Recall, precision, F1-score, annualized return	Python, Keras, Tensorflow, Java
[200]	Borsa Istanbul 100 Stocks	2011-2015	75 technical indicators, OCHLV of price	-	1h	CNN	Accuracy	Keras

In the literature, some of the studies used technical indicators & price data & fundamental data at the same time. Table 11 tabulates the trend forecasting papers using technical indicators, price data, fundamental data. In addition, these studies are clustered into three sub-groups: **ANN**, **MLP**, **DBN**, and **RBM** models; **LSTM** and **GRU** models; novel methods. **ANN**, **MLP**, **DBN**, and **RBM** methods were used with technical indicators, price data and fundamental data in some of the studies. In [192], several classical, **ML** models and **DBN** were compared for trend forecasting. In [193], technical analysis indicator's (**RSI**) buy & sell limits were optimized with **GA** which was used for buy-sell signals. After optimization, **DMLP** was also used for function approximation. The authors of [194] used technical analysis parameters, **OCHLV** of prices and **RBM** for stock trend prediction.

Besides, **LSTM** and **GRU** methods with technical indicators & price data & fundamental data were also used in some of the papers. In [195], the crossover and **Moving Average Convergence and Divergence (MACD)** signals were used to predict the trend of the Dow 30

stocks prices. The authors of [196] used **LSTM** for stock price movement estimation. The author of [197] used stock prices, technical analysis features and four different **ML** Models (**LSTM**, **GRU**, **SVM** and **eXtreme Gradient Boosting (XGBoost)**) to predict the trend of the stocks prices.

In addition, there were also novel and new methods that used **CNN** with the price data and technical indicators. The authors of [198] converted the time series of price data to 2-dimensional images using technical analysis and classified them with deep **CNN**. Similarly, the authors of [199] also proposed a novel technique that converted financial time series data that consisted of technical analysis indicator outputs to 2-dimensional images and classified these images using **CNN** to determine the trading signals. The authors of [200] proposed a method that used **CNN** with correlated features combined together to predict the trend of the stocks prices.

Besides, there were also studies that used text mining techniques in the literature. Table 12 tabulates the trend forecasting papers using text mining techniques. Different methods/models are represented within four sub-groups in that table: **DNN**, **DMLP**, and **CNN** with text mining models; **GRU** model; **LSTM**, **CNN**, and **LSTM+CNN** models; novel methods. In the first group of studies, **DNN**, **DMLP**, **CNN** with text mining were used for trend forecasting. In [201], the authors used different models that included **Hidden Markov Model (HMM)**, **DMLP** and **CNN** using Twitter moods to predict the next days' move. In [202], the authors used the combination of text mining and word embeddings to extract information from financial news and **DNN** model for prediction of the stock trends.

Moreover, **GRU** methods with text mining techniques were also used for trend forecasting. The authors of [203] used financial news from Reuters, Bloomberg and stock prices data and **Bidirectional Gated Recurrent Unit (Bi-GRU)** model to predict the stock movements in the future. The authors of [204] used Stock2Vec and **Two-stream GRU (TGRU)** models to generate input data from financial news and stock prices. Then, they used the sign difference between the previous close and next open for the classification of the stock prices. The results were better than the state-of-the-art models.

**LSTM**, **CNN** and **LSTM+CNN** models were also used for trend forecasting. The authors of [205] combined news data with financial data to classify the stock price movement and assessed them with certain factors. They used **LSTM** model as the **NN** architecture. The authors of [206] proposed a novel method that used character-based neural language model using financial news and **LSTM** for trend prediction. In [207], sentiment/mood prediction and price prediction based on sentiment, price prediction with text mining and **DL** models (**LSTM**, **NN**, **CNN**) were used for trend forecasting. The authors of [208] proposed a method that used two separate **LSTM** networks to construct an ensemble network. One of the **LSTM** models was used for word embeddings with word2Vec to create a matrix information as input to **CNN**. The other one was used for price prediction using technical analysis features and stock prices.

In the literature, there were also novel and different methods to predict the trend of the time series data. In [209], the authors proposed a novel method that uses a combination of **RBM**, **DBN** and word embedding to create word vectors for **RNN-RBM-DBN** network to predict the trend of stock prices. The authors of [210] proposed a novel method (called

DeepClue) that visually interpreted text-based **DL** models in predicting stock price movements. In their proposed method, financial news, charts and social media tweets were used together to predict the stock price movement. The authors of [211] proposed a method that performed information fusion from several news and social media sources to predict the trend of the stocks. The authors of [212] proposed a novel method that used text mining techniques and Hybrid Attention Networks based on financial news for the forecast of the trend of stocks. The authors of [213] combined technical analysis and sentiment analysis of social media (related financial topics) and created **Deep Random Subspace Ensembles (DRSE)** method for classification. The authors of [214] proposed a method that used **Deep Neural Generative Model (DGM)** with news articles using Paragraph Vector algorithm to create the input vector for the prediction of the trend of stocks. The authors of [215] implemented intraday stock price direction classification using financial news and stocks prices.

Table 12: Trend Forecasting Using Text Mining Techniques

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[201]	S&P500, NYSE Composite, DJIA, NASDAQ Composite	2009-2011	Twitter moods, index data	7d	1d	DNN, CNN	Error rate	Keras, Theano
[202]	News from Reuters and Bloomberg, Historical stock security data	2006-2013	News, price data	5d	1d	DNN	Accuracy	-
[203]	News from Reuters, Bloomberg	2006-2013	Financial news, price data	-	1d, 2d, 5d, 7d	Bi-GRU	Accuracy	Python, Keras
[204]	News about Apple, Airbus, Amazon from Reuters, Bloomberg, S&P500 stock prices	2006-2013	Price data, news, technical indicators	-	-	Two-stream GRU, stock2vec	Accuracy, precision, AUROC	Keras, Python
[205]	NIFTY50 Index, NIFTY Bank/Auto/IT/Energy Index, News	2013-2017	Index data, news	1d, 2d, 5d	1d	LSTM	MCC, Accuracy	-
[206]	News from Reuters, Bloomberg, stock price/index data from S&P500	2006-2013	News and sentences	-	1h, 1d	LSTM	Accuracy	-
[207]	30 DJIA stocks, S&P500, DJI, news from Reuters	2002-2016	Price data and features from news articles	1m	1d	LSTM, NN, CNN and word2vec	Accuracy	VADER
[208]	APPL from S&P500 and news from Reuters	2011-2017	News, OCHLV, Technical indicators	-	1d	CNN + LSTM, CNN+SVM	Accuracy, F1-score	Tensorflow
[209]	News, Nikkei Stock Average and 10-Nikkei companies	1999-2008	News, MACD	-	1d	RNN, RBM+DBN	Accuracy, P-value	-
[210]	News from Reuters and Bloomberg for S&P500 stocks	2006-2015	Financial news, price data	1d	1d	DeepClue	Accuracy	Dynet software
[211]	Price data, index data, news, social media data	2015	Price data, news from articles and social media	1d	1d	Coupled matrix and tensor	Accuracy, MCC	Jieba
[212]	News and Chinese stock data	2014-2017	Selected words in a news	10d	1d	HAN	Accuracy, Annual return	-

Table 12: Trend Forecasting Using Text Mining Techniques

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[213]	Sina Weibo, Stock market records	2012-2015	Technical indicators, sentences	-	-	DRSE	F1-score, precision, recall, accuracy, AU-ROC	Python
[214]	Nikkei225, S&P500, news from Reuters and Bloomberg	2001-2013	Price data and news	1d	1d	DGM	Accuracy, MCC, %profit	-
[215]	News, stock prices from Hong Kong Stock Exchange	2001	Price data and TF-IDF from news	60min	(1..6)*5mi	ELM, DLR, PCA, BELM, KELM, NN	Accuracy	Matlab

Moreover, there were also studies that used different data variations in the literature. Table 13 tabulates the trend forecasting papers using these various data clustered into two sub-groups: **LSTM**, **RNN**, **GRU** models; **CNN** model.

**LSTM**, **RNN**, **GRU** methods with various data representations were used in some trend forecasting papers. In [216], the authors used the limit order book time series data and **LSTM** method for trend prediction. The authors of [217] proposed a novel method that used limit order book flow and history information for the determination of the stock movements using **LSTM**. The results of the proposed method were remarkably stationary. The authors of [154] used social media news, **LDA** features and **RNN** model to predict the trend of the index price. The authors of [218] proposed a novel method that used expert recommendations (Buy, Hold or Sell), ensemble of **GRU** and **LSTM** to predict the trend of the stocks prices.

**CNN** models with different data representations were also used for trend prediction. In [219], the authors used the last 100 entries from the limit order book to create images for the stock price prediction using **CNN**. Using the limit order book data to create 2D matrix-like format with **CNN** for predicting directional movement was innovative. In [159], HFT microstructures forecasting with **CNN** was implemented.

Table 13: Trend Forecasting Using Various Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[216]	Nasdaq Nordic (Kesko Oyj, Outokumpu Oyj, Sampo, Rautaruukki, Wartsila Oyj)	2010	Price and volume data in <b>LOB</b>	100s	10s, 20s, 50s	<b>LSTM</b>	Precision, Recall, F1-score, Cohen's k	-
[217]	High-frequency record of all orders	2014-2017	Price data, record of all orders, transactions	2h	-	<b>LSTM</b>	Accuracy	-
[154]	Chinese, The Shanghai-Shenzhen 300 Stock Index (HS300)	2015-2017	Social media news (Sina Weibo), price data	1d	1d	<b>RNN-Boost with LDA</b>	Accuracy, MAE, MAPE, RMSE	Python, Scikit learn
[218]	ISMIS 2017 Data Mining Competition dataset	-	Expert identifier, class predicted by expert	-	-	<b>LSTM</b> + <b>GRU</b> + <b>FCNN</b>	Accuracy	-

Table 13: Trend Forecasting Using Various Data

Art.	Data Set	Period	Feature Set	Lag	Horizon	Method	Performance Criteria	Env.
[219]	Nasdaq (Kesko Oyj, Outokumpu Oyj, Sampo, Rautaruukki, Wartsila Oyj)	Nordic Oyj, 2010	Price, Volume data, 10 orders of the LOB	-	-	CNN	Precision, Recall, F1-score, Cohen's k	Theano, Scikit learn, Python
[159]	London Stock Exchange	2007-2008	Limit order book state, trades, buy/sell orders, order deletions	-	-	CNN	Accuracy, kappa	Caffe

## 5. Current Snapshot of The Field

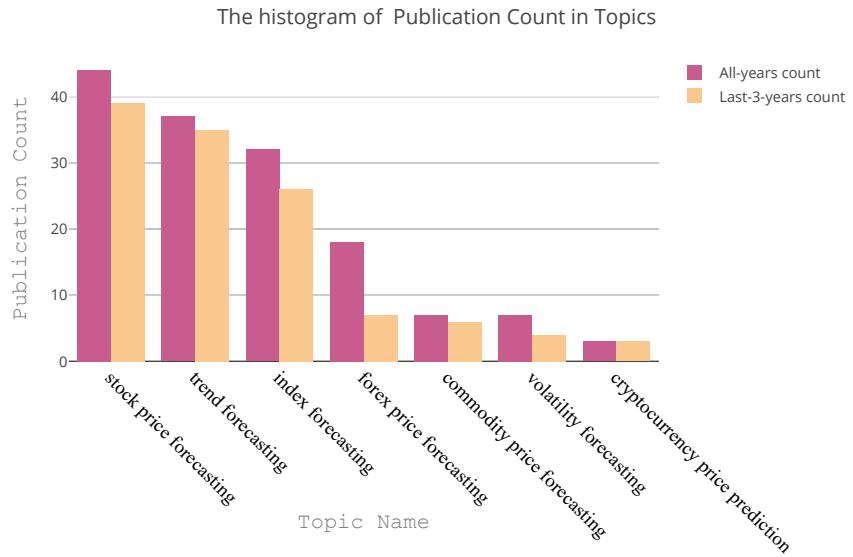


Figure 5: The histogram of Publication Count in Topics

After reviewing through all the research papers specifically targeted for financial time series forecasting implementations using **DL** models, we are now ready to provide some overall statistics about the current state of the studies. The number of papers that we were able to locate to be included in our survey was 140. We categorized the papers according to their forecasted asset type. Furthermore, we also analyzed the studies through their **DL** model choices, frameworks for the development environment, data sets, comparable benchmarks, and some other differentiating criteria like feature sets, number of citations, etc. which we were not able to include in the paper due to space constraints. We will now summarize our notable observations to provide important highlights for the interested researchers within the field.

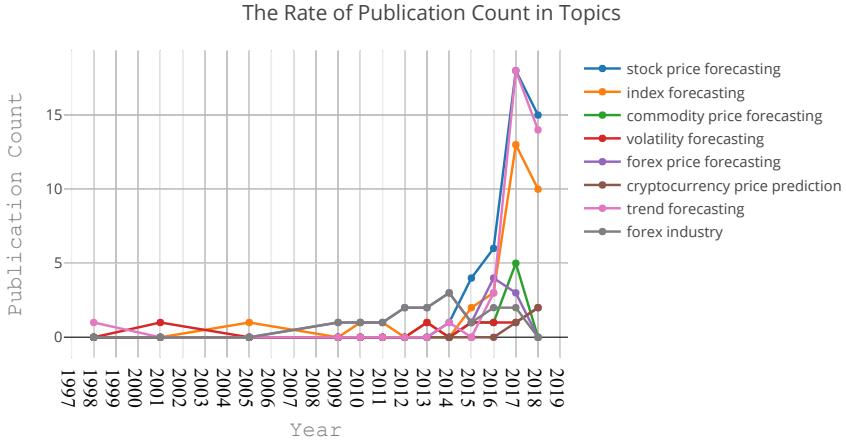


Figure 6: The rate of Publication Count in Topics

Figure 5 presents the various asset types that the researchers decided to develop their corresponding forecasting models for. As expected, stock market-related prediction studies dominate the field. Stock price forecasting, trend forecasting and index forecasting were the top three picks for the financial time series forecasting research. So far, 46 papers were published for stock price forecasting, 38 for trend forecasting and 33 for index forecasting, respectively. These studies constitute more than 70% of all studies indicating high interest. Following those include 19 papers for forex prediction and 7 papers for volatility forecasting. Meanwhile cryptocurrency forecasting has started attracting researchers, however, there were just 3 papers published yet, but this number is expected to increase in coming years [220]. Figure 6 highlights the rate of publication counts for various implementation areas throughout the years. Meanwhile Figure 7 provides more details about the choice of DL models over various implementation areas.

Figure 8 illustrates the accelerating appetite in the last 3 years by researchers for developing DL models for the financial time series implementations. Meanwhile, as Figure 9 indicates, most of the studies were published in journals (57 of them) and conferences (49 papers) even though a considerable amount of arXiv papers (11) and graduate theses (6) also exist.

One of the most important questions for a researcher is where he/she can publish their research findings. During our review of the papers, we also carefully investigated where each paper was published. We tabulated our results for the top journals for financial time series forecasting in Fig 10. According to these results, the journals with the most published papers include Expert Systems with Applications, Neurocomputing, Applied Soft Computing, The Journal of Supercomputing, Decision Support Systems, Knowledge-based Systems,

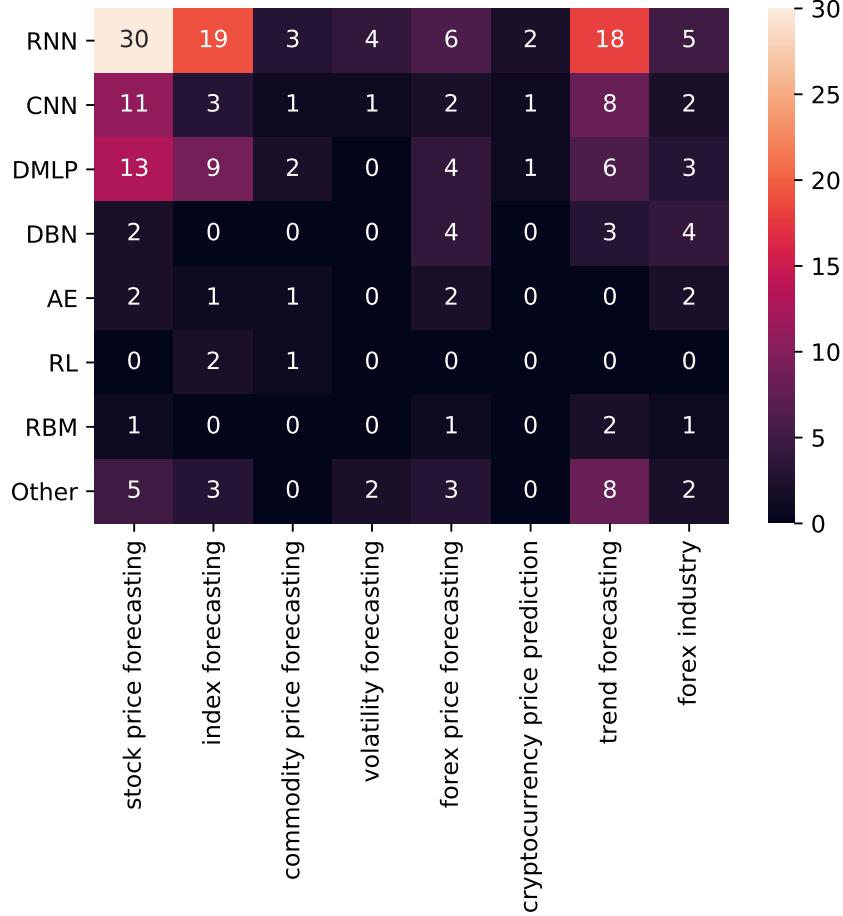


Figure 7: Topic-Model Heatmap

European Journal of Operational Research and IEEE Access. The interested researchers should also consider the trend within the last 3 years, as tendencies can be slightly varying depending on the particular implementation areas.

Carefully analyzing Figure 11 clearly validates the dominance of **RNN** based models (65 papers) among all others for **DL** model choices, followed by **DMLP** (23 papers) and **CNN** (20 papers). The inner-circle represents all years considered, meanwhile the outer circle just provides the studies within the last 3 years. We should note that **RNN** is a general model with several versions including **LSTM**, **GRU**, etc. Within **RNN**, the researchers mostly prefer **LSTM** due to its relative easiness of model development phase, however, other types of **RNN** are also common. Figure 12 provides a snapshot of the **RNN** model distribution. As mentioned above, **LSTM** had the highest interest among all with 58 papers, while Vanilla **RNN** and **GRU** had 27 and 10 papers respectively. Hence, it is clear that **LSTM** was the most popular **DL** model for financial time series forecasting or regression studies.

Meanwhile, **DMLP** and **CNN** generally were preferred for classification problems. Since the time series data generally consists of temporal components, some data preprocessing

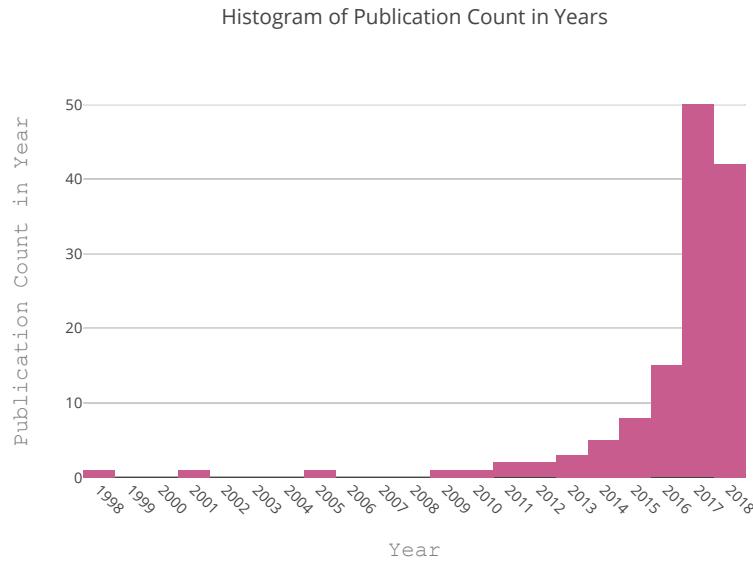


Figure 8: The histogram of Publication Count in Years

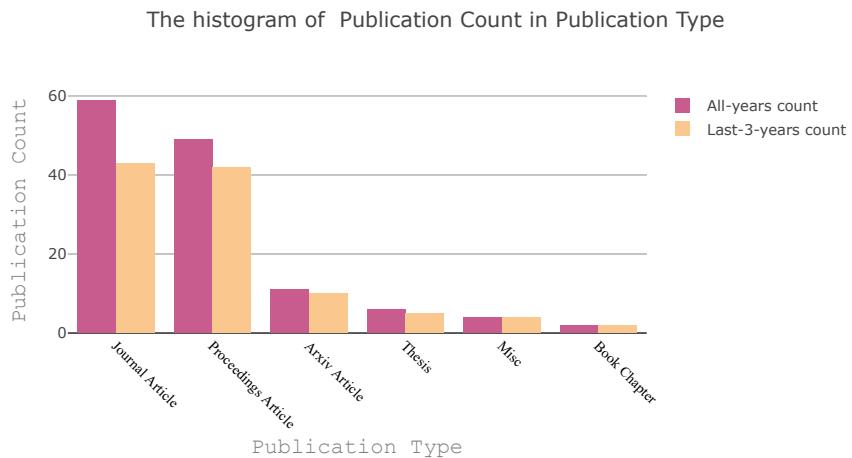


Figure 9: The histogram of Publication Count in Publication Types

might be required before the actual classification can occur. Hence, a lot of these implementations utilize feature extraction, selection techniques along with possible dimensionality reduction methods. A lot of researchers decided to use [DMLP](#) mostly due to the fact that its shallow version [MLP](#) has been used extensively before and has a proven successful track record for many different financial applications including financial time series forecasting.

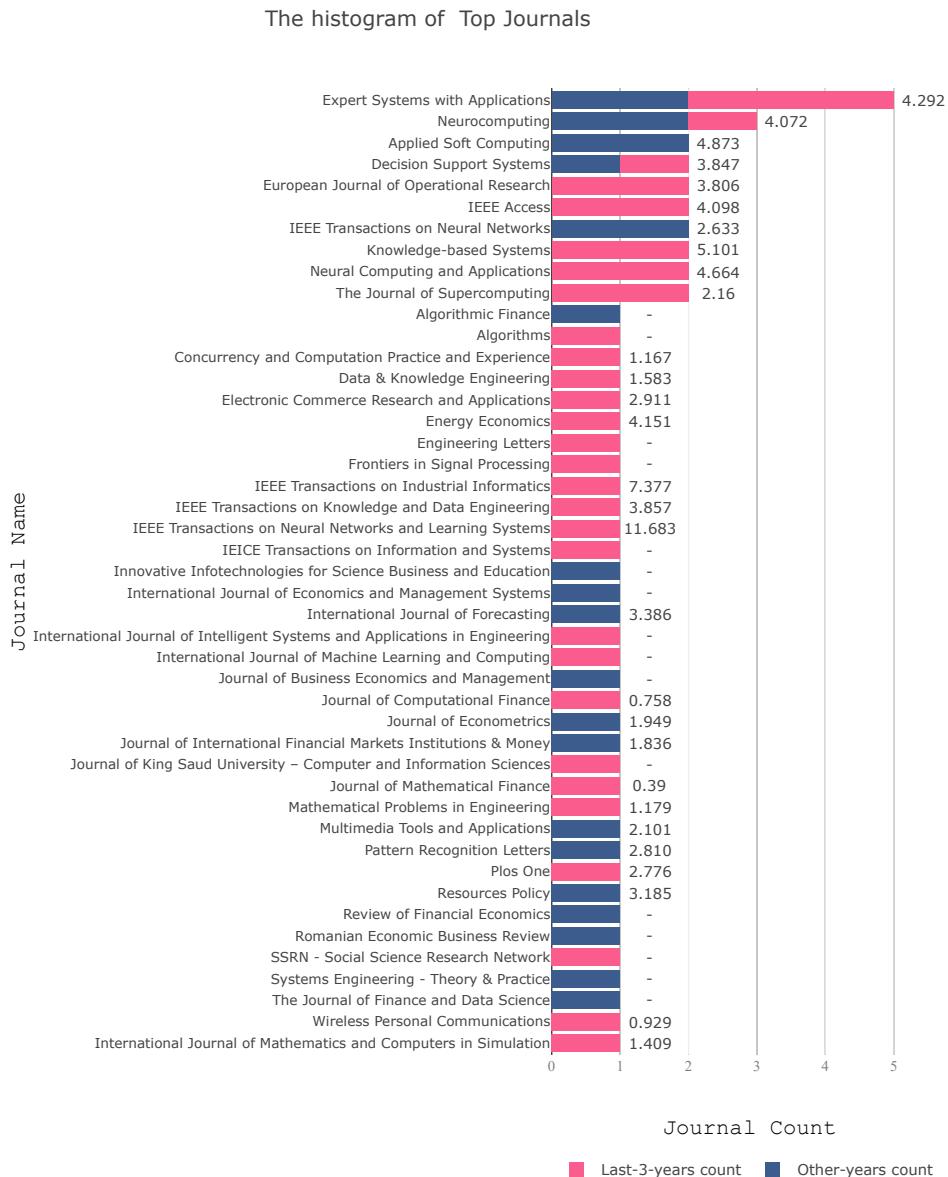


Figure 10: Top Journals - corresponding numbers next to the bar graph are representing the impact factor of the journals

Consistent with our observations, [DMLP](#) was also mostly preferred in the stock, index or in particular trend forecasting, since it is by definition, a classification problem with two (uptrend or downtrend) and three (uptrend, stationary or downtrend) class instances.

In addition to [DMLP](#), [CNN](#) was also a popular choice for classification type financial time series forecasting implementations. Most of these studies appeared within the last 3 years.

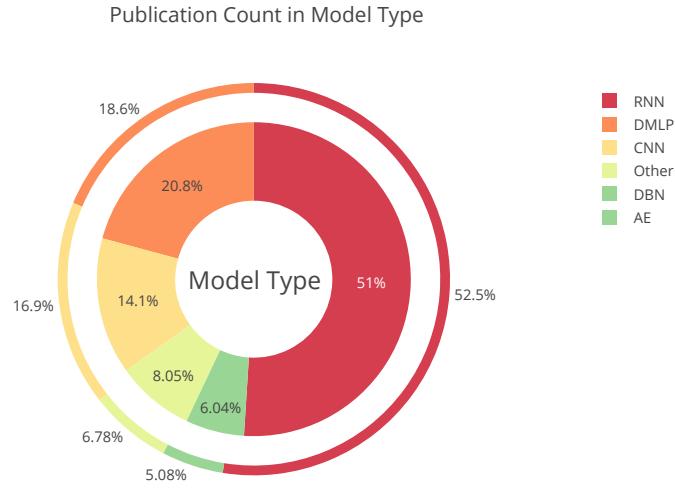


Figure 11: The Piechart of Publication Count in Model Types

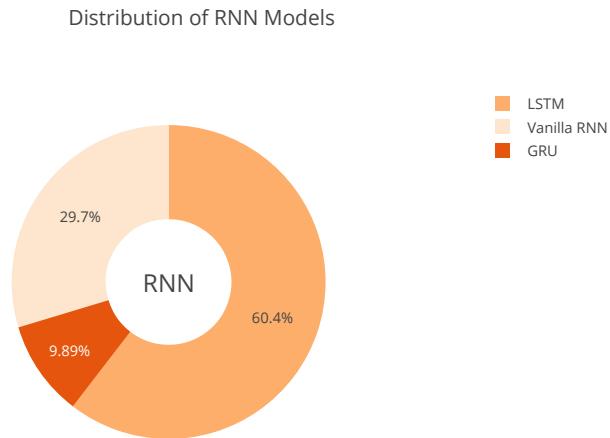


Figure 12: Distribution of RNN Models

As mentioned before, in order to convert the temporal time-varying sequential data into a more stationary classifiable form, some preprocessing might be necessary. Even though some 1-D representations exist, the 2-D implementation for [CNN](#) was more common, mostly inherited through image recognition applications of [CNN](#) from computer vision implementations. In some studies [[188](#), [189](#), [193](#), [199](#), [219](#)], innovative transformations of financial time

series data into an image-like representation has been adapted and impressive performance results have been achieved. As a result, [CNN](#) might increase its share of interest for financial time series forecasting in the next few years.

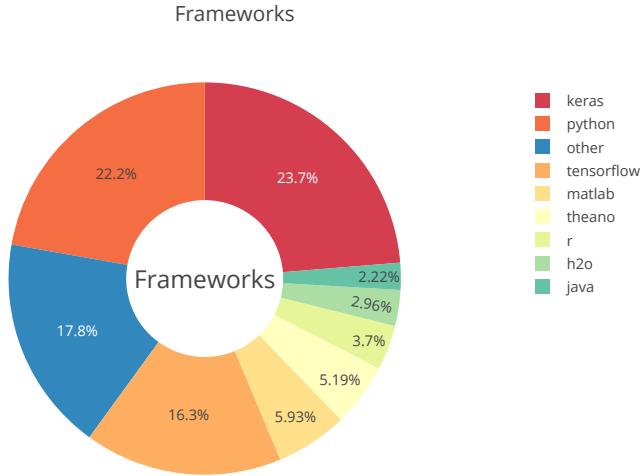


Figure 13: The Preferred Development Environments

As one final note, Figure 13 shows which frameworks and platforms the researchers and developers used while implementing their work. We tried to extract this information from the papers to the best of our effort. However, we need to keep in mind that not every publication provided their development environment. Also in most of the papers, generally, the details were not given preventing us from a more thorough comparison chart, i.e. some researchers claimed they used Python, but no further information was given, while some others mentioned the use of Keras or TensorFlow providing more details. Also, within the "Other" section the usage of Pytorch is on the rise in the last year or so, even though it is not visible from the chart. Regardless, Python-related tools were the most influential technologies behind the implementations covered in this survey.

## 6. Discussion and Open Issues

From an application perspective, even though financial time series forecasting has a relatively narrow focus, i.e. the implementations were mainly based on price or trend prediction, depending on the underlying [DL](#) model, very different and versatile models exist in literature. We need to keep in mind that, even though financial time series forecasting is a subset of time-series studies, due to the embedded profit-making expectations through successful prediction models, some differences exist, such that higher prediction accuracy sometimes might not reflect a profitable model. Hence, the risk and reward structure must also be

taken into consideration. At this point, we will try to elaborate on our observations about these differences in various model designs and implementations.

### 6.1. DL Models for financial time series forecasting

According to the publication statistics, **LSTM** was the preferred choice of most researchers for financial time series forecasting. **LSTM** and its variations utilized the time-varying data with feedback embedded representations, resulting in higher performances for time series prediction implementations. Since most of the financial data, one way or another, included time-dependent components, **LSTM** was the natural choice in financial time series forecasting problems. Meanwhile, **LSTM** is a special **DL** model derived from a more general classifier family, namely **RNN**.

Careful analysis of Figure 11 illustrates the dominance of **RNN** (which is highly consisted of **LSTM**). As a matter of fact, more than half of the published papers for time series forecasting studies fall into the **RNN** model category. Regardless of its problem type, price or trend prediction, the ordinal nature of the data representation forced the researchers to consider **RNN**, **GRU** and **LSTM** as viable preferences for their model choices. Hence, **RNN** models were chosen, at least for benchmarking, in a lot of studies for performance comparison against other developed models.

Meanwhile, other models were also used for time series forecasting problems. Among those, **DMLP** had the most interest due to the market dominance of its shallow cousin, **MLP** and its wide acceptance and long history within **ML** society. However, there is a fundamental difference in how **DMLP** and **RNN** based models were used for financial time series prediction problems.

**DMLP** fits well for both regression and classification problems. However, in general, data order independence must be preserved for better utilizing the internal working dynamics of such networks, even though through the learning algorithm configuration, some adjustments can be performed. In most cases, either trend components of the data need to be removed from the underlying time series, or some data transformations might be needed so that the resulting data becomes stationary. Regardless, some careful preprocessing might be necessary for the **DMLP** model to be successful. In contrast, **RNN** based models can directly work with time-varying data, making it easier for researchers to develop **DL** models.

As a result, most of the **DMLP** implementations had embedded data preprocessing before the learning stage. However, this inconvenience did not prevent the researchers to use **DMLP** and its variations during their model development process. Instead, a lot of versatile data representations were attempted in order to achieve higher overall prediction performances. A combination of fundamental and/or technical analysis parameters along with other features like financial sentiment through text mining was embedded into such models. In most of the **DMLP** studies, the corresponding problem was treated as classification, especially in trend prediction models, whereas **RNN** based models directly predicted the next value of the time series. Both approaches had some success in beating the underlying benchmark; hence it is not possible to claim victory of one model type over the other. However, for the general rule of thumb, researchers prefer **RNN** based models for time series regression and **DMLP** for trend classification (or buy-sell point identification)

Another model that started becoming popular recently is **CNN**. **CNN** also works better for classification problems and unlike **RNN** based models, it is more suitable for either non-time varying or static data representations. The comments for **DMLP** are also mostly valid for **CNN**. Furthermore, unlike **DMLP**, **CNN** mostly requires locality within the data representation for better-performing classification results. One particular implementation area of **CNN** is image-based object recognition problems. In recent years, **CNN** based models dominated this field, handily outperforming all other models. Meanwhile, most financial data is time-varying and it might not be easy to implement **CNN** directly for financial applications. However, in some recent studies, various independent research groups followed an innovative transformation of 1-D time-varying financial data into 2-D mostly stationary image-like data so that they could utilize the power of **CNN** through adaptive filtering and implicit dimensionality reduction. Hence, with that approach, they were able to come up with successful models.

There is also a rising trend to use deep **RL** based financial algorithmic trading implementations; these are mostly associated with various agent-based models where different agents interact and learn from their interactions. This field even has more opportunities to offer with advancements in financial sentiment analysis through text mining to capture investor psychology; as a result, behavioral finance can benefit from these particular studies associated with **RL** based learning models coupled with agent-based studies.

Other models including **DBN**, **AE** and **RBM** also were used by several researchers and superior performances were reported in some of their work; but the interested readers need to check these studies case by case to see how they were modelled both from the data representation and learning point of view.

### *6.2. Discussions on Selected Features*

Regardless of the underlying forecasting problem, somehow the raw time series data was almost always embedded directly or indirectly within the feature vector, which is particularly valid for **RNN**-based models. However, in most of the other model types, other features were also included. Fundamental analysis and technical analysis features were among the most favorable choices for stock/index forecasting studies.

Meanwhile, in recent years, financial text mining is particularly getting more attention, mostly for extracting the investor/trader sentiment. The streaming flow of financial news, tweets, statements, blogs allowed the researchers to build better and more versatile prediction and evaluation models integrating numerical and textual data. The general methodology involves in extracting financial sentiment analysis through text mining and combining that information with fundamental/technical analysis data to achieve better overall performance. It is logical to assume that this trend will continue with the integration of more advanced text and **NLP** techniques.

### *6.3. Discussions on Forecasted Asset Types*

Even though forex price forecasting is always popular among the researchers and practitioners, stock/index forecasting has always had the most interest among all asset groups.

Regardless, price/trend prediction and algo-trading models were mostly embedded with these prediction studies.

These days, one other hot area to financial time series forecasting research is involved with cryptocurrencies. Cryptocurrency price prediction has an increasing demand from the financial community. Since the topic is fairly new, we might see more studies and implementations coming in due to high expectations and promising rewards.

There were also a number of publications in commodity price forecasting research, in particular, the price of oil. Oil price prediction is crucial due to its tremendous effect on world economic activities and planning. Meanwhile, gold is considered a safe investment and almost every investor, at one time, considers allocating some portion of their portfolios for gold-related investments. In times of political uncertainties, a lot of people turn to gold for protecting their savings. Even though we have not encountered a noteworthy study for gold price forecasting, due to its historical importance, there might be opportunities in this area for the years to come.

#### *6.4. Open Issues and Future Work*

Despite the general motivation for financial time series forecasting remaining fairly unchanged, the means of achieving the financial goals vary depending on the choices and trade-off between the traditional techniques and newly developed models. Since our fundamental focus is on the application of **DL** for financial time series studies, we will try to assess the current state of the research and extrapolate that into the future.

##### *6.4.1. Model Choices for the Future*

The dominance of **RNN**-based models for price/trend prediction will probably not disappear anytime soon, mainly due to their easy adaptation to most asset forecasting problems. Meanwhile, some enhanced versions of the original **LSTM** or **RNN** models, generally integrated with hybrid learning systems started becoming more common. Readers need to check individual studies and assess their performances to see which one fits the best for their particular needs and domain requirements.

We have observed the increasing interest in 2-D **CNN** implementations of financial forecasting problems through converting the time series into an image-like data type. This innovative methodology seems to work quite satisfactorily and provides promising opportunities. More studies of this kind will probably continue in the near future.

Nowadays, new models are generated through older models via modifying or enhancing the existing models so that better performances can be achieved. Such topologies include **Generative Adversarial Network (GAN)**, Capsule networks, etc. They have been used in various non-financial studies, however, financial time series forecasting has not been investigated for those models yet. As such, there can be exciting opportunities both from research and practical point of view.

Another **DL** model that is not investigated thoroughly is Graph **CNN**. Graphs can be used to represent portfolios, social networks of financial communities, fundamental analysis data, etc. Even though graph algorithms can directly be applied to such configurations, different graph representations can also be implemented for the time series forecasting problems. Not

much has been done on this particular topic, however, through graph representations of the time series data and implementing graph analysis algorithms, or implementing [CNN](#) through these graphs are among the possibilities that the researchers can choose.

As a final note for the future models, we believe deep [RL](#) and agent-based models offer great opportunities for the researchers. [HFT](#) algorithms, robo-advisory systems highly depend on automated algorithmic trading systems that can decide what to buy and when to buy without any human intervention. These aforementioned models can fit very well in such challenging environments. The rise of the machines will also lead to a technological (and algorithmic) arms race between Fintech companies and quant funds to be the best in their neverending search for “achieving alpha”. New research in these areas can be just what the doctor ordered.

#### *6.4.2. Future Projections for Financial Time Series Forecasting*

Most probably, for the foreseeable future, the financial time series forecasting will have a close research cooperation with the other financial application areas like algorithmic trading and portfolio management, as it was the case before. However, changes in the available data characteristics and introduction of new asset classes might not only alter the forecasting strategies of the developers, but also force the developers to look for new or alternative techniques to better adapt to these new challenging working conditions. In addition, metrics like [Continuous Ranked Probability Score \(CRPS\)](#) for evaluating probability distributions might be included for more thorough analysis.

One rising trend, not only for financial time series forecasting, but for all intelligent decision support systems, is the human-computer interaction and [NLP](#) research. Within that field, text mining and financial sentiment analysis areas are of particular importance to financial time series forecasting. Behavioral finance may benefit from the new advancements in these fields.

In order to utilize the power of text mining, researchers started developing new data representations like Stock2Vec [204] that can be useful for combining textual and numerical data for better prediction models. Furthermore, [NLP](#) based ensemble models that integrate data semantics with time-series data might increase the accuracy of the existing models.

One area that can benefit a lot from the interconnected financial markets is the automated statistical arbitrage trading model development. It has been used in forex and commodity markets before. In addition, a lot of practitioners currently seek arbitrage opportunities in the cryptocurrency markets [220], due to the existence of the huge number of coins available on various marketplaces. Price disruptions, high volatility, bid-ask spread variations cause arbitrage opportunities across different platforms. Some opportunists develop software models that can track these price anomalies for the instant materialization of profits. Also, it is possible to construct pairs trading portfolios across different asset classes using appropriate models. It is possible that [DL](#) models can learn (or predict) these opportunities faster and more efficient than classical rule-based systems. This will also benefit [HFT](#) studies that are constantly looking for faster and more efficient trading algorithms and embedded systems with minimum latency. In order to achieve that, Graphics Processing Unit (GPU) or Field Programmable Gate Array (FPGA) based hardware solutions embedded with [DL](#) models

can be utilized. There is a lack of research accomplished on this hardware aspect of financial time series forecasting and algorithmic trading. As long as there is enough computing power available, it is worth investigating the possibilities for better algorithms, since the rewards are high.

### 6.5. Responses to our Initial Research Questions

We are now ready to go back to our initially stated research questions. Our question and answer pairs, through our observations, are as follows:

- Which DL models are used for financial time series forecasting ?

Response: RNN based models (in particular LSTM) are the most commonly used models. Meanwhile, CNN and DMLP have been used extensively in classification type implementations (like trend classification) as long as appropriate data processing is applied to the raw data.

- How is the performance of DL models compared with traditional machine learning counterparts ?

Response: In the majority of the studies, DL models were better than ML. However, there were also many cases where their performances were comparable. There were even two particular studies ([82, 175] where ML models performed better than DL models. Meanwhile, appetite for preferrance of DL implementations over ML models is growing. Advances in computing power, availability of big data, superior performance, implicit feature learning capabilities and user friendly model development environment for DL models are among the main reasons for this migration.

- What is the future direction for DL research for financial time series forecasting ?

Response: NLP, semantics and text mining-based hybrid models ensembled with time-series data might be more common in the near future.

## 7. Conclusions

Financial time series forecasting has been very popular among ML researchers for more than 40 years. The financial community got a new boost lately with the introduction of DL implementations for financial prediction research and a lot of new publications appeared accordingly. In our survey, we wanted to review the existing studies to provide a snapshot of the current research status of DL implementations for financial time series forecasting. We grouped the studies according to their intended asset class along with the preferred DL model associated with the problem. Our findings indicate, even though financial forecasting has a long research history, overall interest within the DL community is on the rise through utilizing new DL models; hence, a lot of opportunities exist for researchers.