Neural Computing and Applications (2020) 32:7897–7914
https://doi.org/10.1007/s00521-019-04236-3

HYBRID ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING TECHNOLOGIES

Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction

Hyejung Chung[1] · Kyung-shik Shin[1]

Received: 26 November 2018 / Accepted: 9 May 2019 / Published online: 20 May 2019

Abstract

Recently, artificial intelligence technologies have received considerable attention because of their practical applications in various fields. The key factor in this prosperity is deep learning which is inspired by the information processing in biological brains. In this study, we apply one of the representative deep learning techniques multi-channel convolutional neural networks (CNNs) to predict the fluctuation of the stock index. Furthermore, we optimize the network topology of CNN to improve the model performance. CNN has many hyper-parameters that need to be adjusted for constructing an optimal model that can learn the data patterns efficiently. In particular, we focus on the optimization of feature extraction part of CNN, because this is the most important part of the computational procedure of CNN. This study proposes a method to systematically optimize the parameters for the CNN model by using genetic algorithm (GA). To verify the effectiveness of our model, we compare the prediction result with standard artificial neural networks (ANNs) and CNN models. The experimental results show that the GA-CNN outperforms the comparative models and demonstrate the effectiveness of the hybrid approach of GA and CNN.

Keywords Convolutional neural network · Genetic algorithm · Deep learning · Stock market prediction

List of symbols

C_t	Closing price at time t
L_t	Low price at time t
H_t	High price at time t
LL_t	The lowest low in the last t days
HH_t	Highest high in the last t days
Up_t	Upward price change at time t
Dw_t	Downward price change at time t
EMA_t	Exponential moving average for t days
M	Typical price which is calculated using $(H_t + L_t + C_t)/3$
m	Simple moving average which is calculated using $\left(\sum_{i=1}^{n} M_{t-i+1}\right)/n$
d	Mean absolute deviation which is calculated using $\left(\sum_{i=1}^{n} \lvert M_{t-i+1} - m_t\rvert\right)/n)$

Abbreviations

AI	Artificial intelligence
CNN	Convolutional neural network
RNN	Recurrent neural network
ANN	Artificial neural network
ILSVRC	ImageNet large-scale visual recognition challenge
GA	Genetic algorithm
SVM	Support vector machine
CBR	Case-based reasoning
EMH	Efficient market hypothesis
ES	Exponential smoothing
ARIMA	Autoregressive integrated moving average
ARCH	Autoregressive conditional heteroscedasticity
GARCH	Generalized autoregressive conditional heteroscedasticity
TOPIX	Tokyo stock exchange price indexes
PNN	Probabilistic neural network
TDNN	Time-delay neural network
LDA	Linear discriminant analysis
QDA	Quadratic discriminant analysis
EBNN	Elman back-propagation neural network
PCA	Principal component analysis
KOSPI	Korea composite stock price index 200
MV	Majority vote
WMV	Weighted majority vote
BC	Borda count

✉ Kyung-shik Shin
ksshin@ewha.ac.kr

[1] School of Business, Ewha Womans University, 52 Ewhayeodae-gil, Seodaemun-gu, Seoul 03760, Korea

WBC	Weighted Borda count
BKS	Behavior-knowledge space
IBB	NASDAQ biotechnology index
RBM	Restricted Boltzmann machine
NMSE	Normalized mean squared error
RMSE	Root-mean-squared error
MAE	Mean absolute error
MI	Mutual information
LSTM	Long short-term memory
MNIST	Mixed national institute of standards and technology
k-NN	k-nearest neighbor
DTW	Dynamic time warping
CIFAR10	Canadian institute for advanced research
PSO	Particle swarm optimization
SVR	Support vector regression
RSI	Relative strength index
MACD	Moving average convergence divergence
CCI	Commodity channel index
ReLU	Rectified linear unit
Adam	Adaptive moment estimation

1 Introduction

Recently, artificial intelligence (AI) technologies have received considerable attention because of their practical applications in various fields including computer vision and natural language processing and have become an active research area. In the history of AI, deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been considered a key factor in this prosperity. Deep learning, also called a deep neural network, is a type of artificial neural networks (ANNs) that has many hidden layers between the input and the output layers. Deep learning not only increases the complexity of the model but also assigns an appropriate role to each neuron for the purpose of analysis; therefore, it can effectively solve the target problems that could not be solved using traditional neural network models [1].

Deep Learning has recently attracted considerable attention with the huge success of the deep learning-based image classification model, AlexNet, which won the ImageNet large-scale visual recognition challenge (ILSVRC) in 2012 [2]. AlexNet considerably improved the classification accuracy of object recognition and proved the potential of deep learning techniques. Considering the huge success of AlexNet, the deep learning models started to have deeper and more complicated architecture. In addition to the advancement of learning capability, deep learning is expanding its application range to various fields.

In this study, we apply multi-channel CNN model to predict the fluctuation of the stock index. As interest in financial investment has increased because of the continuous economic recession and low interest rates environment, there has been a steady attempt to apply AI techniques to financial analysis. Despite the fact that many studies have presented various methodologies, the stock market has been considered a very difficult area to predict because of its extreme noise and nonlinear characteristics. Recently, CNN has been applied to various time-series problems of a diverse nature, such as voice recognition and natural language processing, and many studies have demonstrated its efficiency on time-series data [3, 4]. As CNN has the advantage of extracting the local features of the data, it can be an effective prediction methodology for the time-series problems by capturing the temporal property of the given dataset.

Furthermore, in this study, we aim to optimize the network topology of CNN to improve the model performance. Although many studies have shown effectiveness of CNN models in various classification problems, there are several shortcomings in building and using the model. Despite its predictive power, neural network models require longer training time and extra resources as well as careful tuning of various hyper-parameters to reach its full potential [5–7]. In fact, the main drawback of CNN is its difficulty to determine a suitable model that can successfully reflect the characteristics of the problem and efficiently learn the data patterns, due to the numerous learning methods, network architectures, and other controlling parameters [8]. In particular, the architectural factors including kernel size of the convolutional layers, the number of kernels, and the pooling window size are very important, because these parameters directly affect the results of the feature extraction, which is a crucial part of the computational procedure of CNN. However, most of the existing studies tend to depend on trial-and-error-based methods, and such methods are more of an art than a science. Therefore, here, we propose a method to systematically optimize the parameters for the CNN model by using genetic algorithm (GA). We employ GA in the search for CNN's optimal architecture which allows to make use of neural networks' ability to model complex nonlinear functions while automatically solving the optimization problem [7]. Many studies have applied GA in conjunction with other AI and machine learning techniques such as ANN [9–13], support vector machine (SVM) [14–17], and case-based reasoning (CBR) [18, 19]. However, few studies have attempted to integrate GA and CNN, despite a great potential for effective applications in this area.

This paper consists of seven sections. In Sect. 1, we describe the background and the purpose of the research, and in Sect. 2, we examine the research trends of stock

market forecasting and the recent research on the CNN. In Sect. 3, we introduce the main methodology used in this study, namely, CNN and GA. In Sect. 4, we propose a CNN model optimized by GA. In Sect. 5, we explain the experimental design that used in the study. Section 6 presents the experimental results and analysis. Finally, in Sect. 7, conclusions and future research directions are discussed.

2 Related works

2.1 Stock market prediction

The stock market is known to be very difficult to predict and analyze because of the nature of its noisy and nonlinear environment [20]. According to the efficient market hypothesis (EMH), the price of the capital market immediately reflects all the available information; therefore, forecasting and achieving excess returns are impossible [21]. However, stock market forecasting is treated as an important research topic in various academic and practical fields such as mathematics, statistics, and financial engineering, and the results of empirical studies have demonstrated the possibility of stock market prediction [22–24].

The methodology of stock market forecasting is divided into fundamental analysis and technical analysis [20]. The fundamental analysis, on one hand, includes qualitative and quantitative analysis as a way to predict future stock prices by analyzing the intrinsic value of stocks. Qualitative analysis is based on factors that cannot be quantified, such as political situations, economy, and labor issues. Quantitative analysis is based on factors that can be quantified, such as economic indicators and financial statements [25]. On the other hand, technical analysis aims at predicting future market trends on the basis of the historical stock market data, such as past stock prices and trading volumes, without considering the external indicators [26–29].

Traditionally, stock market analysis adopted a variety of statistical approaches including exponential smoothing (ES), autoregressive integrated moving average (ARIMA), autoregressive conditional heteroscedasticity (ARCH) and generalized autoregressive conditional heteroscedasticity (GARCH) [23, 30–33]. However, stock prediction systems using such methods with various statistical assumptions did not achieve satisfactory results, because of the difficulty in meeting statistical assumptions such as linearity and normality postulates and independence among input variables [34].

Because financial markets are considered as chaotic and nonparametric dynamic systems [20], it is essential to develop more flexible models which can learn complex dimensionality. AI and machine learning approaches are highly capable of capturing nonlinear data patterns without the need of any background knowledge about the data sets [35]. Accordingly, AI and machine learning models started to be applied to the financial time-series forecasting problems approving better ability to investigate links between market elements [22, 23, 29, 36–39].

Among the various machine learning techniques, ANN and SVM are the most widely used for the stock market prediction. Saad et al. [40] investigated the effectiveness of three neural network models, namely probabilistic neural network (PNN), time-delay neural network (TDNN), and RNN for low false alarm stock trend predictions. They used different predictability analysis techniques including phase diagrams, correlation dimension and Lyapunov exponent and analyzed the neural network results based on a history of the daily closing price. Consequently, the RNN outperformed other considered models. Fernandez-Rodrıguez et al. [36] proposed a technical trading rule based on ANN. To investigate the profitability of this investment strategy, the proposed model was applied to the general index of the Madrid stock market. The experimental results demonstrated the superiority of the ANN-based technical trading rule for a bear market and a stable market.

Some applications of SVM to stock market prediction also have been reported with excellent generalization ability. Huang et al. [22] used the SVM to predict the weekly volatility of the Nikkei 225 stock index futures. They used macroeconomic indicators including interest rate and consumer price index as the input variables and compared the proposed model with statistical methods such as linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) along with the Elman back-propagation neural network (EBNN) to prove the effectiveness of the SVM. Yu et al. [41] developed SVM-based stock selection system. To obtain informative and low-dimensional financial time-series, principal component analysis (PCA) was employed for the feature selection. The experimental results presented that PCA–SVM achieved higher returns of stocks than other benchmark models.

Some researchers hybridized several different AI techniques to improve the prediction accuracy [42, 43]. In particular, many researchers have integrated GA and other AI techniques to enhance the effectiveness of the forecasting procedure. GA has shown good performance in exploring a complex search space, guided by evolution operators such as crossover and mutation. Kim and Han [26] combined the ANN model with GA for feature discretization and optimization of the connection weights between the layers in the ANN. The proposed model was applied to the prediction of Korea composite stock price index 200 (KOSPI 200). The hybrid ANN model with globally searched discretization and connection weights outperformed the conventional back-propagation neural

network. Armano et al. [27] integrated ANN and GA to develop a stock market forecasting system by experimenting on the data of the COMIT (Banca Commerciale Italiana) and the S&P500 stock index. The forecasting system was composed of genetic and neural components, and each part of the model contains different information. The genetic component dealt with inputs to encode information retrieved from a technical analysis, whereas the neural part dealt with the past stock prices. The simulation results demonstrated the superiority of the proposed model and repeatedly outperformed the buy-and-hold strategy. Kim et al. [44] used the GA-based multiple classifier model to predict the KOSPI. They combined the results of multiple classifiers developed by human experts, users, and machine learning techniques through GA. The experimental results of the proposed model were compared with well-known combining algorithms such as the majority vote (MV), the weighted majority vote (WMV), the Borda count (BC), the weighted Borda count (WBC), the Bayesian method, the behavior–knowledge space (BKS), and the Dempster–Shafer theory. The results showed that the GA-based multiple classifier outperformed the individual classifiers and the multiple classifiers that exploited the conventional combining method.

In recent years, with considerable achievements in various prediction tasks, deep learning techniques have been applied in the financial field. Heaton et al. [45] used the deep neural network for financial prediction and classification problems. They autoencoded all the stocks of the NASDAQ biotechnology index (IBB) over the period 2012–2016 and ranked them by the similarity to their own autoencoded version. The proposed model demonstrated that the deep learning method could dramatically improve the prediction performance of the financial market. Chong et al. [29] proposed the deep feature learning-based stock market prediction model. They predicted future market trends with a deep neural network and compared the effectiveness of unsupervised feature extraction techniques including autoencoder, restricted Boltzmann machine (RBM), and PCA. To evaluate the performance of each method, they used normalized mean squared error (NMSE), root-mean-squared error (RMSE), mean absolute error (MAE), and mutual information (MI). Fischer and Krauss [46] used long short-term memory (LSTM) networks to forecast the volatility of the stocks listed on S&P 500 from 1992 to 2015. They demonstrated that LSTM networks outperform the traditional approaches that do not make use of past information such as random forests and logistic regression classifiers. Further, they found one common pattern among the stocks and formalized a rule-based reversal strategy.

Deep learning techniques have shown outstanding performance in various fields, but application to the financial area has not been studied much thus far. The limitation of these studies is that they are restricted to only the basic deep learning techniques and do not extend to the novel models. Moreover, some of these studies did not achieve outstanding performance partly because many researchers still do not seriously consider the adequate network design of deep learning models for stock prediction. Table 1 presents a summary of relevant studies that apply machine learning approaches to stock market forecasting.

2.2 Recent research trend of CNN

CNN is a neural network model that mimics the visual processing in the human brain and is usually applied to image classification problems in the field of computer vision. The research that generated the interest in CNN was the handwriting recognition study of LeCun et al. [47]. LeCun et al. [47] applied LeNet-5, a modern form of CNN, to the handwritten data of mixed national institute of standards and technology (MNIST) and compared the experimental results with other machine learning techniques such as the k-nearest neighbor (k-NN) and SVM. CNN began to be used for image classification tasks in earnest when "AlexNet" of Krizhevsky et al. [2] won the ILSVRC, which is a global visual recognition competition. Krizhevsky et al. [2] proposed a CNN model similar to LeNet-5 but with more hidden layers. With the huge success of AlexNet, many researchers started to develop deeper and more complex neural network models for image recognition and classification tasks such as VGGNet [48], GoogleNet [49] and ResNet [50].

Furthermore, CNN has achieved good results in time-series problems. When convolution kernels that share the same weights are applied to local signals at different time segments, a type of translation invariance is obtained. Since the temporal location of an event has a significant effect on the outcomes of the time-series analysis, CNN can efficiently identify the periodic patterns while decreasing the unnecessary operation with the convolution kernels. Furthermore, significant information can be extracted through the pooling layers. When applying the CNN to the time-series data, usually one-dimensional (1D) kernels are used instead of the conventional two-dimensional (2D) or three-dimensional (3D) kernels to perform the convolution operations.

Kim [3] performed a sentence-level sentiment analysis using CNN with pre-trained word vectors including word2vec. Each channel of CNN was composed of different word embedding methods, and the kernel size was varied to extract the features from different angles. The proposed model showed better performance than baseline models such as a model with a randomly initialized word vector and the single-channel CNN.

Table 1 A summary of previous studies on stock market prediction

Researchers (year)	Data type	Goal	Method(s)	Benchmark	Performance measure
Saad, Prokhorov, and Wunsch (1998) [40]	US four stocks of NYSE and NASDAQ	Compare neural network models (TDNN, RNN, and PNN)	TDNN, RNN, and PNN	Fisher's linear classifier	False alarm rate
Fernandez-Rodrıguez, Gonzalez-Martel, and Sosvilla-Rivero (2000) [36]	Spain MADX	Investigate the profitability of technical trading rule	ANN	Buy-and-hold strategy	Trading simulation
Huang, Nakamori, and Wang (2005) [22]	Japan NIKKEI 225 index	Predict weekly volatility of stock index futures	SVM	LDA, QDA, and EBNN	Directional accuracy
Yu, Chen, and Zhang (2014) [41]	China SSEC	Construct stock selection system	SVM + PCA	–	Directional accuracy
Kim and Han (2000) [26]	Korea KOSPI 200	Predict direction of stock movement	ANN + GA	ANN	Directional accuracy
Armano, Marchesi, and Murru (2005) [27]	Italy COMIT and US S&P 500	Forecast next-day price of stock market indexes	ANN + GA	Buy-and-hold strategy	Percentage of annualized returns
Kim, Min, and Han (2006) [44]	Korea KOSPI	Predict direction of stock movement	ANN + human-driven classifier + GA	MV, WMV, Bayesian, BC, WBC, BKS, and Dempster–Shafer theory	Directional accuracy
Heaton, Polson, and Witte (2017) [45]	NASDAQ biotechnology index (IBB)	Find a selection of investment strategy	DNN and LSTM network	–	MSE
Chong, Han, and Park (2017) [29]	Korea 38 stocks of KOSPI	Compare deep feature learning-based stock market prediction model	DNN	Autoregressive model and ANN	NMSE, RMSE, MAE, and MI
Fischer and Krauss (2017) [46]	US S&P 500	Predict direction of stock movement	LSTM network	Random forest and logistic regression	Directional accuracy

Zheng et al. [4] conducted experiments on multivariate time-series data sets that belong to different domains by using a multi-channel CNN. Each channel of the CNN took a single dimension of the multivariate time-series as the input and learned the features individually. The proposed model combined the features that were learned from an individual channel and fed them into the fully connected layer to perform the classification. The experimental results showed that the proposed multi-channel CNN model outperformed the nearest neighbor classification (particularly 1-NN) combined with dynamic time warping (DTW) and ANN.

Although CNN has achieved satisfactory results in various fields, there are only a few studies on the optimization of the models' hyper-parameters. There are various hyper-parameters in the CNN that can affect the performance of the model, such as the kernel size, the number of kernels, and so on. However, most of the existing studies designed neural network models based on the decision of experts or experimental methods instead of systematic methods.

Simonyan and Zisserman [48] used a number of small kernels, such as 3×3 and 1×1, to demonstrate the effectiveness of small kernels. They compared the experimental results with those of the CNN models with relatively larger kernel size, such as 11×11 and 7×7. They achieved outstanding results through these small kernels and insisted that the small kernels could augment the nonlinearity and increase the learning effectiveness.

2.3 Hyper-parameter optimization of neural network models

Hyper-parameters of neural network models, such as number and size of hidden layers, activation thresholds, learning rate, dropout rate and so on must be defined before actual training of the model. There are many methods for hyper-parameter tuning of neural network models, such as Bayesian optimization-based methods [51], gradient search [52], grid search and random search [53]. In addition to those approaches, various metaheuristic algorithms such as GA, particle swarm optimization (PSO) and tabu search

have also been applied to hyper-parameter optimization problems in different applications. Jaddi et al. [54] proposed the combined method GA and ANN which allows the exploration of different number of nodes for each layer, number of hidden layers, weights and biases. Tian et al. [55] proposed a constraint-based GA to find optimal architectures of two hidden layers for detecting loss of a nuclear power plant's coolant accident. The GA used in this study created an initial population of architectural factors of neural networks by using a proposed constraint satisfaction algorithm; random walk. The experimental results demonstrated that the GA-optimized ANN outperformed random search, exhaustive search and a support vector regression (SVR) in terms of generalization performance. Ciancio et al. [56] compared the performances of four optimization techniques; GA, tabu search, Taguchi and decision trees. They investigated the optimal architectures of ANN for manufacturing problems such as extrusion process, the rolling process and the shearing process. For each problem, the four algorithms searched for the optimal value of the number of hidden layers, the number of neurons of each hidden layer, the types of the activation functions and the type of the training algorithm. As the result, GA outperformed other methods when applied to the problems of extrusion and shearing processes.

Although many studies have attempted to apply GA for the designing of ANN architecture, literature remains limited for the CNN model.

3 Methodology

3.1 Convolutional neural network (CNN)

CNN is a subtype of ANN for processing data with a grid-like structure, such as images, which are 2D grids of pixels, and time-series data, which are 1D grids taking samples at regular time intervals. The basic mechanism of CNN is inspired by the organization of the visual cortex of animals. CNN is composed of one or more convolutional layers and fully connected layers; additionally, pooling layers can be used. The conceptual schematic representation of the basic CNN structure is depicted in Fig. 1.

CNN is based on three major concepts that help to efficiently learn important patterns from the input data: local connectivity, weight sharing, and sub-sampling [57].

In traditional ANN models, neurons in the lower layer are connected to all the neurons in the upper layer, and the relationship between the neurons is derived using matrix multiplication [58]. The local connectivity of CNN means that the neurons in the lower layer are connected only to the local receptive field of the higher layer; the other part is not included in the operation, thus reducing the computational complexity with a sparse interaction where the convolution kernels move over the entire input data [58]. Figure 2 is an example of local connectivity in CNN.

Like other ANN models, CNN has weights and biases, which are updated during the learning process. However, CNN applies the same weights and biases to all the neurons in the corresponding layer. This makes it possible to detect the same features in different areas of the input data, and the model shows excellent learning performance with fewer parameters than other neural network models [1]. The CNN topology uses the spatial relationships between neurons to reduce the number of parameters for the learning process, and the performance is therefore improved by using the standard back-propagation algorithms [58].

Sub-sampling refers to the process of extracting a sample after separating each layer. By reducing the number of features, we can reduce the computational complexity and avoid overfitting.

The recent developments in CNN have increased the fitting capacity considerably. As shown in Fig. 1, CNN is a neural network model with multiple hidden layers consisting of two different types of layers, namely convolutional and pooling layers. The main computational layers of CNN and their operating process are as follows.

3.1.1 Convolutional layer

In general, convolutional layers extract features that have local correlations when the positional relationships between the local features are determined. In the convolutional layer, the kernel (or filter) extracts features while moving on the input data at regular intervals. As shown in Fig. 3, the corresponding elements in the kernel and the input data are multiplied and the results are summed. The elements constituting the kernel applied to the multiplication operation correspond to the weight matrix of the conventional ANN. The trainable kernels at all the possible offsets are convolved with input data and produce feature maps in the convolutional layer. The training process of a convolutional layer is resembling that for a conventional ANN using back-propagation. However, unlike standard ANN, CNN only needs to train the kernels of each convolutional layer.

3.1.2 Pooling layer

In the pooling layer, the samples of the most representative features are extracted from the convolutional layer. The sampling methods include max-pooling and average pooling. Sampling is conducted by extracting the maximum or the average value of each interval. The pooling procedure

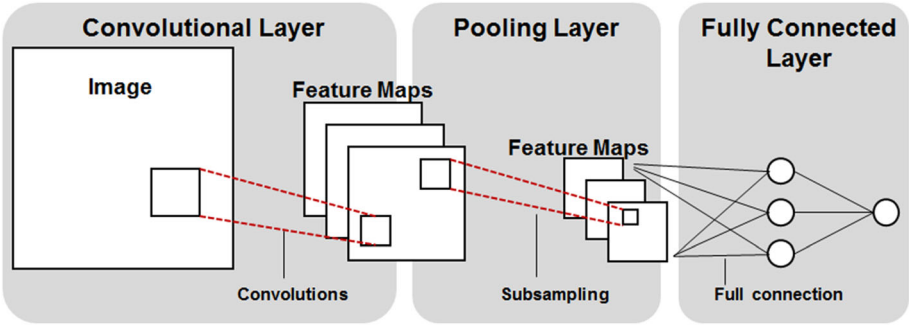

Fig. 1 The basic architecture of CNN

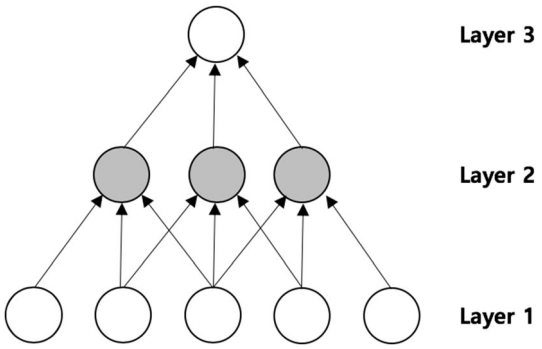

Fig. 2 Local connectivity in CNN

can remove the unnecessary detailed information, thereby ensuring that only the important information is selected for the learning procedure. Through this process, the influence of the noise data can be reduced and better robustness is gained.

3.1.3 Fully connected layer

The fully connected layer links the neurons generated in the convolutional layer and the pooling layer to all the neurons of the higher layer and performs the prediction and classification operations. It has the same structure and performs the same role as those of the standard ANN models.

3.2 Genetic algorithm (GA)

GA is a parallel and global search algorithm inspired by Darwin's theory of "survival of the fittest" [59]. GA can effectively solve complex problems by simulating the process of natural evolution. In general, if the problem is too computationally complicated to be solved, GA can be used as a method to obtain a solution close to the optimal one. GA can be characterized by the genetic operators that imitate the biological process of reproduction. The chromosomes which are possible solutions to a computational problem are generated randomly, and the chromosomes

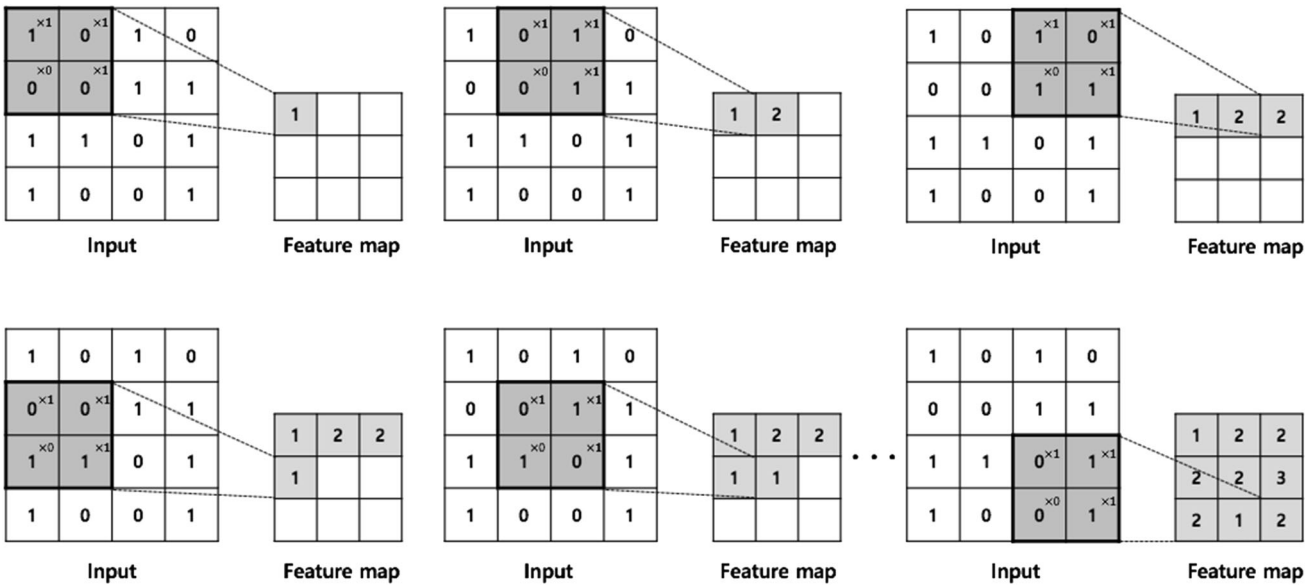

Fig. 3 The process of convolution operation

with higher fitness get better opportunities to be reproduced [27]. GA increasingly produces better solutions by gradually transforming the chromosomes that are the candidate solutions to the given problem. In this process, the solution can be expressed as a gene, and the process of creating a better solution by transformation can be understood as evolution.

The genetic representation and the definition of the fitness function are the two major requirements of a GA process. Genetic representation is a way to express the potential solutions (individuals) of the target problem. It is important because the result of the genetic representation affects all GA operations, such as crossover and mutation; thus, it should be able to reflect the nature of the target problem well. The fitness function is a type of objective function used to guide simulations toward optimal design solutions. This function measures the performance of each chromosome, and through this evaluation process, GA finds the optimal solution for the problem [60].

Six stages are considered in GA: initialization, fitness calculation, selection, crossover, mutation, and termination condition check [61], as shown in Fig. 4.

In the initialization stage, a GA randomly initializes a population of candidate solutions that will perform the genetic operations. The size of the population usually depends on the characteristics of the target problem and ranges to several hundreds or thousands of possible solutions.

Once the population is initialized, the fitness of each chromosome is calculated in every generation. The predefined fitness function is used to select good solutions [60]. In the GA process, the determination of the fitness function is crucial step, because it can seriously affect the performance of model. If the definition of the fitness function is inappropriate for the target problem, GA can converge on the wrong solution.

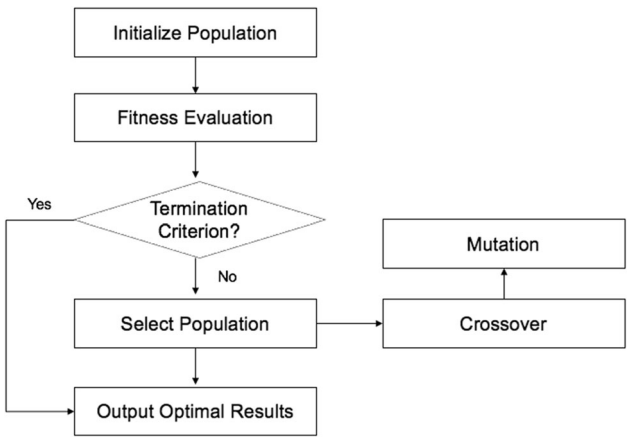

Fig. 4 The operation process of GA

Selection allocates reproductive opportunities to each chromosome. In the process of calculating the fitness of each solution, only solutions with higher fitness values are selected for the recombination process. The main idea of selection is to choose better solutions and let superior ones pass their genes to the next generation.

Crossover is the most important phase in GA operation. The selected chromosomes generate offspring by combining genetic information of two parents. Two of the chosen chromosomes swap the corresponding parts of the string and change the gene combinations to create new candidate solutions. Figure 5 illustrates the process of crossover operation.

In the mutation process, randomly chosen chromosomes flip the individual bits as shown in Fig. 6. The crossover process is limited in that completely new information cannot be generated. However, this limitation can be overcome by the mutation operation, because it alters each bit of a chromosome to a completely new value. The aim of this process is to maintain the diversity in the population and prevent the local optimum problem.

The newly produced chromosomes through the generational process are evaluated to check if the termination condition has been reached. The GA process is over when the population has converged. However, the whole process is iterated until at least one of the termination conditions is satisfied. The general termination conditions are as follows [54–56]:

- A solution which meets the minimum criteria is derived.
- The predefined number of generations is reached.
- A successive state that no longer produces better results is reached.

4 GA-optimized multi-channel CNN model

4.1 Financial time-series analysis using multi-channel deep CNN

The term "time-series" refers to the series of sequential data points according to the flow of time. As the temporal changes of these data points are caused by various reasons, the prediction and utilization of time-series data are intricate. Despite these difficulties, a time-series analysis is significant because of the many opportunities of its application in a broad range of real-world scenarios. In fact, multivariate time-series can provide more information and patterns of the same underlying phenomena and help improve the performance of prediction models when compared to univariate time-series. Therefore, we use multivariate financial time-series data to predict the

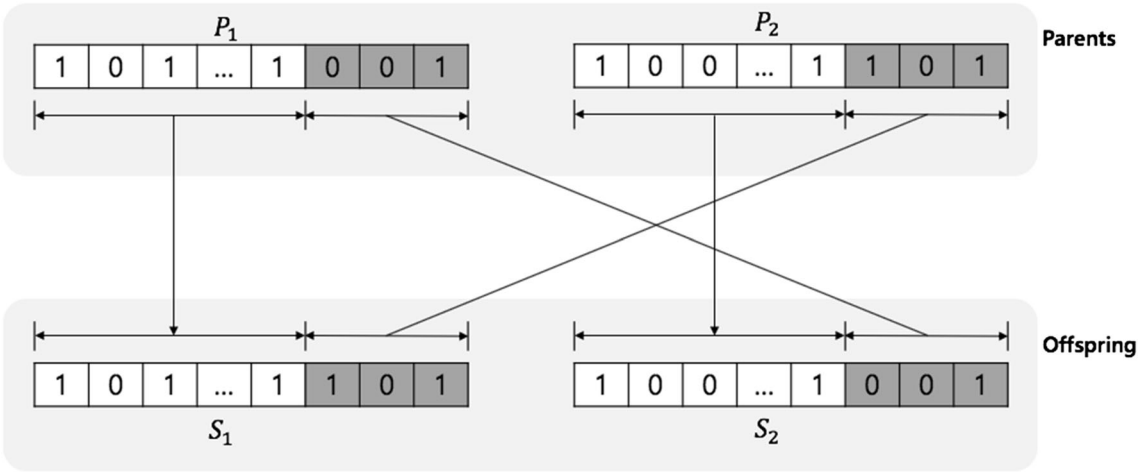

Fig. 5 Crossover operation

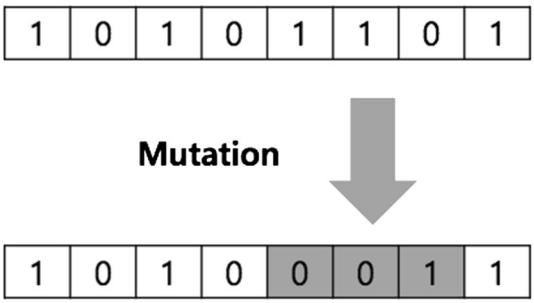

Fig. 6 The mutation operation

direction of the stock index, and adopt a multi-channel CNN model which can learn the temporal patterns of multivariate time-series more effectively.

In this study, we are inspired by the CNN models for the image and time-series classification problems and use a deep multi-channel CNN framework for a multivariate time-series classification. The conventional multi-layer perceptron usually ignores the local information or the structure of the input data. However, most real-world data such as image, speech, and time-series data have a strong 2D or 1D local structure, and the variables (pixels in the case of image data) nearby are spatially or temporally highly correlated. CNN has advantage in that aspect since convolutional layers can capture these local correlations by extracting and combining local features [57]. The modified CNN model proposed in this study is applied to the multivariate time-series classification task in two stages:

Stage 1 We divide multivariate time-series into univariate ones where each univariate time-series comprises an individual channel. Each channel of CNN is composed of the single dimension of multivariate time-series. The signals of input variables have local characteristics along the time axis, which means temporal patterns exist in the data spectrum. These local trends of input sequence become the crucial evidence to distinguish the direction of stock index. CNN has shown great ability of modeling these local structures in time-series data by allowing each unit of the convolutional layer to process input signals only for its receptive field. For the purpose of local processing of input data, the input spectrum is divided into the size of local kernels. As depicted in Fig. 7, the predefined size of kernel sweeps over the input signal with shared weights and each kernel learns the local temporal information. After the convolution operation, the learned features go through a nonlinear activation function. The output values from the convolutional layer which are called as feature maps are the input of the pooling layer. The feature maps are pooled by means of max-pooling or average pooling to reduce the features dimension and improve computational efficiency. Several authors have stipulated that max-pooling shows a better performance than average pooling while allowing faster convergence by selecting superior invariant features and avoiding the overfitting phenomena [62]. Hence, we adopted the max-pooling method for further improvement of the prediction performance.

Stage 2 At the end of feature extraction, we concatenate the trainable fully connected part to perform further

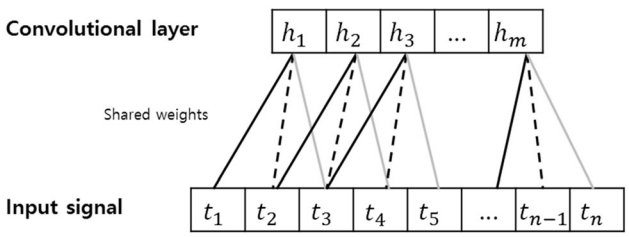

Fig. 7 The feature learning process in convolutional layer

classification. These fully connected layers which are connected to the two output classes (up/down trend) are anticipated to combine local information extracted from the lower layers. The feature maps of each channel are flattened and combined as the input of fully connected layers. Then the fully connected layers map the features from convolutional and pooling operations to target labels. In other words, the classification is conducted based on the learned features from the convolutional and pooling layers. For each feature, the weight is given through the training process and finally the total score for target classes is calculated. The input data is considered to belong to the class with the higher score.

In this study, the CNN model includes 7-channel inputs. For each channel, convolution kernels that extract the features move over the univariate time-series and learn hierarchical features. The CNN model of this study utilizes 2 convolutional layers and 1 pooling layer.

4.2 Optimizing the architecture of multi-channel CNN with evolutionary algorithm

There has been a dramatic improvement in the use of deep learning techniques in various areas of research. Despite the fact that deep learning can learn features and optimize weight parameters in a data-driven manner, the selection of the model architecture still remains a manual process in a highly intuitive manner. The network topology for the input and the output variables and the learning parameters are examples of the part that require expert intervention for tuning and adjustment. However, as the total number of network structures increases exponentially with the increase in the network's depth, it is impossible to evaluate all the possible candidates to discover the optimal set of components. To overcome this cumbersome process and retain the flexibility of the model, we propose a novel approach by using the GA to find the optimal topology of the CNN model automatically.

CNN is generally composed of two parts. One is the feature extraction part, which contains the convolutional and pooling layers and the other is the trainable fully connected part for the prediction or classification. In this study, we specifically focus on generating the optimal topology in the feature extraction part. The convolutional and pooling layers of the CNN detect the patterns and extract the key features of the given input as the kernels move over the original data. Thus, determining the appropriate design for these layers can result in considerable performance improvements. Optimum subset of such hyper-parameters may vary with the dynamics of the input signal, so these parameters are best determined with a systematic approach. In the case of the size of kernel, if the kernel size is very large, the detailed characteristics of the input data cannot be taken into consideration, and a very small kernel may cause confusion by learning too much information. Moreover, the number of kernels for each convolutional layer can affect the process of feature learning. Every kernel produces different feature maps and acts as a feature detector with a different point of view. As the number of kernels increases, the perspective of analyzing input data varies. However, rather than simply increasing the number of kernels, it is necessary to find an optimal value that can learn features from the input data well and reduce the computational complexity. In most studies that use CNN, the network structure is chosen based on the empirical performance rather the theoretical justifications.

In this study, we employ GA to investigate the optimal structure of a multi-channel CNN model to improve the performance of stock market prediction. Every architectural factor can affect the performance of the network; therefore, a simultaneous adjustment of all of the parameters has to be conducted for the finding of the CNN's optimal structure. The search space associated with the deep neural network designing problem is very large and complex in nature. We employed GA, because its robustness has been theoretically and empirically demonstrated as well as its ability to efficiently solve the large combinational problems. GA has several advantages over other conventional search algorithms in the context of optimization of network design. First, GA is different from other search methods since it deals with population of coded solutions, possessing a global search function and fitting a large-scale parallel process. These characteristics reduce the chance of GA converging toward a local minimum compared to other optimization techniques as it simultaneously considers many points in the search space. In contrast to the single solution-based algorithms such as tabu search and simulated annealing, the GA has excellent global search ability since it is a population-based algorithm [5, 7, 63]. Second, GA detects a set of solutions of a network component which have a minimum cost. These components may correspond to quite different designs that can be then compared in terms of other important but nonquantifiable objectives [10]. Lastly, GA only uses a fitness function, while more conventional search methods need existence and continuity of derivatives or other auxiliary information [64].

The search process of GA consists of the following steps:

Step 1 The search process of GA begins by creating an initial set of possible solutions. The most important task in using the GA is to express the potential solution of the problem to be solved by the chromosomes and to establish the measurement standard for the performance of each

chromosome. As the expression of the potential solution affects all the genetic operations, each chromosome should appropriately reflect the characteristics of the target problem.

As mentioned above, in this study, we optimize the number of kernels, the kernel size, and the pooling window size applied to the operation of the convolutional and pooling layer through GA; therefore, the size of the components of each layer is encoded in binary strings. As shown in Fig. 8, candidate solutions constitute the population of GA. Each chromosome contains the potential value of number of kernels for each convolutional layer, size of kernels for each channel, and pooling window size. That is, various combinations of the candidate structures of the multi-channel CNN compose the population of the solutions.

Step 2 After the population initialization step, the performance of each chromosome is measured by a predefined fitness function. In this study, the fitness function consists of the total classification accuracy of the stock market movement (up/down trends). According to the fitness score, the chromosomes that achieve higher classification accuracy will be reproduced more often than those with lower accuracy.

Step 3 Once the fitness scores for the whole population have been calculated, the genetic operators of selection, crossover and mutation are applied in order to produce the new population of the next generation. The genetic operators provide new information into the population. Through these operations, GA tends to converge on optimal or near-optimal solutions. Finally, the ultimate subset of hyper-parameters for multi-channel CNN is obtained based on the GA search. We illustrate the simplified architecture of the proposed multi-channel CNN model in Fig. 9 for easier presentation.

5 Experimental design

5.1 Data and variables

In this study, 17-year data of the daily KOSPI from January 4, 2000, to December 31, 2016, are used as the experimental data, and the dataset is collected from Bloomberg. The raw data includes the low price, high price, opening price, closing price, and trading volume of 4203 trading days in total. The first 80% of the data is used as the training set, 20% of training data is chosen as validation set, and the remaining 20% is used as the holdout set. The summary statistics of the data components is presented in Table 2.

As input variables significantly affect the performance of the model, careful consideration should be given to the selection. In this study, seven technical indicators are selected as the input variables. Fund managers and traders capture important market signals through technical indicators, and many studies related to stock market prediction have proven the empirical effectiveness of these technical indicators. For the empirical experiment, we referred to the earlier literature [26, 28, 65]. The selected technical indicators and formulas are shown in Table 3.

5.2 Model development

In this study, we employ a multi-channel CNN model consisting of seven channels of input variables, two filter layers, one pooling layer, and two fully connected layers. The inputs of the proposed model are multiple 1D subsequences which are separated from the multivariate time-series. The feature learning is conducted on individual univariate time-series and we concatenate fully connected layers at the end of feature learning to perform the classification. The output variable is the movement direction of the stock index of the following day, and the output values

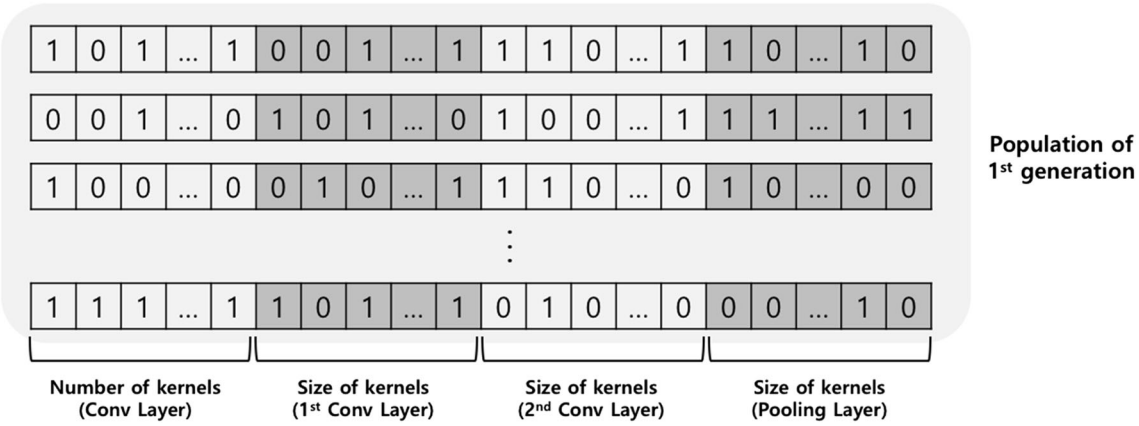

Fig. 8 Example of a chromosome structure used in this study

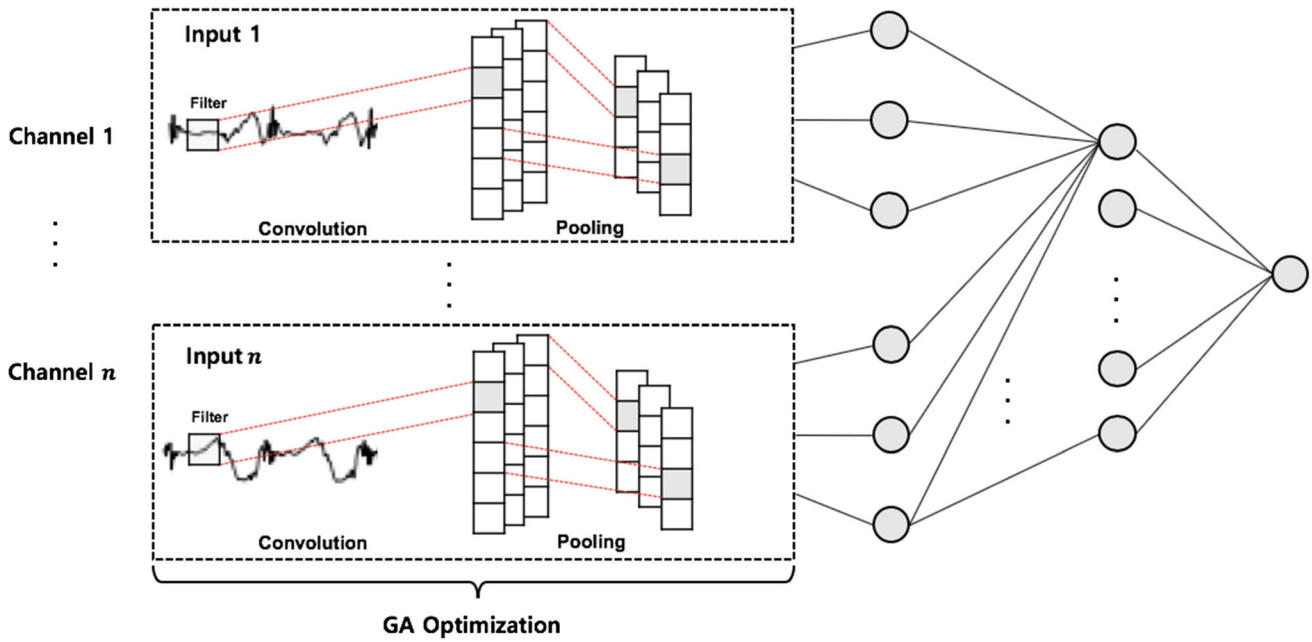

Fig. 9 The basic structure of proposed model

Table 2 The summary statistics of input data

	Opening	High	Low	Closing	Volume
Mean	1431.28	1439.90	1420.12	1430.54	420,925,246
Median	1581.60	1588.73	1566.83	1579.21	379,664,992
Max	2225.95	2231.47	2202.92	2228.96	2,379,293,952
Min	466.57	472.31	463.54	468.76	136,328,992
SD	541.09	541.45	539.59	540.63	192,818,209

are represented in the range of 0 and 1. As the range of the values of raw data varies widely, the input data are linearly scaled to the value between 0 and 1, acquiring the normalized multi-dimensional time-series. This data scaling process increases the efficiency of training procedure of neural network model. The linearly scaled value of x' can be expressed as follows:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

where $\min(x)$ and $\max(x)$ are, respectively, the minimum and the maximum values of the sample x.

In contrast to image classification, we exploit the 1D kernels to extract the local temporal information and the size and the number of the kernels are searched by the GA. In the fully connected layers, we use 200 and 40 hidden nodes, respectively. We implement a rectified linear unit (ReLU) as the activation function of all the layers except the output layer. ReLU is a nonlinear function that outputs x if the input value x is positive and 0 otherwise. The ReLU function was proposed to solve the gradient vanishing problem because traditional sigmoidal functions make the back-propagated errors converge to zero with the increase in the number of layers of the neural network. The ReLU function with input x can be expressed as follows:

Table 3 The selected technical indicators and formulas

Technical indicator	Formula
Momentum	$C_t - C_{t-n}$
Stochastic K %	$\frac{C_t - \mathrm{LL}_{t-n}}{\mathrm{HH}_{t-n} - \mathrm{LL}_{t-n}} \times 100$
Relative strength index (RSI)	$100 - \frac{100}{1 + \left(\sum_{t=0}^{n-1} \mathrm{Up}_{t-1}/n\right) / \left(\sum_{t=0}^{n-1} \mathrm{Dw}_{t-1}/n\right)}$
Moving average convergence divergence (MACD)	$\mathrm{EMA}_{12} - \mathrm{EMA}_{26}$
LW %R	$\frac{H_n - C_t}{H_n - L_n} \times 100$
A/D oscillator	$\frac{H_t - C_{t-1}}{H_t - L_t} \times 100$
Commodity channel index (CCI)	$\frac{M - m}{d \times 0.015} \times 100$

$$f(x) = \max(0, x) \tag{2}$$

To perform the binary classification, the output layer adopts the logistic sigmoid as the activation function. The sigmoid function can be defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Moreover, the adaptive moment estimation (Adam) optimizer is used for weight parameter learning, whose effectiveness is proven by applications to various deep learning models. Adam is known as an efficient weight optimizer, particularly for models with nonstationary and noisy data [66].

To perform the genetic search, we organize a population of chromosomes with the encoded bits of the candidate subset of the parameters of CNN. The encoded parameters and the corresponding chromosome values used in this study are described below.

- Number of kernels in each convolutional layer: 1–63
- Kernel size of individual channels in the first convolutional layer: 1–31
- Kernel size of individual channels in the second convolutional layer: 1–31
- Window size of the pooling layer: 1–31

GA has many controlling parameters to adjust in addition to determining the fitness functions and the problem representation. There has been much debate regarding the optimal controlling parameters that should be specified for the experiment. For this experiment, we set the controlling parameters by referring to previous studies that applied GA to financial problems [44, 60]. In this study, the crossover rate is set to 0.7, and the mutation rate is set to 0.25. The detailed information of the GA parameters is provided in Table 4.

Python version 3.6.1, which is a scientific programming language, and the deep learning framework Keras version 2.0.4 are used to build the proposed optimized CNN model.

Table 4 List of parameters used for the GA

Parameter	Value
Population size	100
Number of generations	10
Crossover rate	0.7
Mutation rate	0.25
Chromosome representation	Binary strings
Fitness function	Hit ratio

6 Result and analysis

This study proposes an integrated deep CNN model to predict KOSPI index movement by deriving the optimal network structure using GA. Table 5 shows the optimal CNN hyper-parameters investigated by GA.

When it comes to stock market prediction, each technical indicator has a different effective time window for the prediction, an aspect considered by only a few studies. However, the proposed model could, first, reflect the effective temporal properties of each input variable for stock market prediction through the individual GA optimization of each channel, and second, enable the finding of the best time window for each technical indicator.

To verify the effectiveness of the proposed model, we compare its performance to standard ANN and CNN models without optimization. We employed accuracy as the performance measure of the classification results. The accuracy evaluates the ability of a classifier to provide a level of accurate diagnosis. The formula of accuracy is as follows.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}(\%) \tag{4}$$

where TP = true positives, TN = true negatives, FP = false positives and FN = false negatives. Table 6 compares the proposed model against the benchmark models (Fig. 10).

The ANN model is organized with one hidden layer containing 10 hidden neurons. The number of hidden neurons is determined by repeated experiments with values between 2 and 12 and selected based on the performance of the validation set. Experimental ANN prediction accuracy is 60.67% and 58.62% for training and holdout data, respectively, which is lower than other CNN models. As presented in Fig. 11, ANN model could not get high performance during training process due to ANN limitations such as its inability to reflect the temporal properties of stock market data when applied on its own.

For ease of comparison, the basic CNN model has the same structure as the proposed model except the optimization structure. The comparative basic CNN model employs 1×5 kernels in each convolutional layer, which is commonly exploited in current time-series classification study using CNN [4]. Prediction accuracy for the standard CNN model is 71.69% and 70.16% for the training and holdout set, respectively, which is significantly higher than the basic ANN model. We believe that the ability to monitor the temporal properties and flexibility of CNN can provide strong support in stock market prediction problem dealing with complicated and rich information in financial time-series data. The training and validation accuracy per epoch in standard CNN model is depicted in Fig. 12.

Table 5 The optimized values of multi-channel CNN

Layer	Optimized parameter	Optimal value
Convolutional layer		
First convolutional layer	# of kernel	56
	Size of kernel	
	Channel 1 (momentum)	4
	Channel 2 (stochastic K%)	22
	Channel 3 (RSI)	27
	Channel 4 (MACD)	6
	Channel 5 (LW %R)	17
	Channel 6 (A/D oscillator)	10
	Channel 7 (CCI)	23
Second convolutional layer	# of kernel	63
	Size of kernel	
	Channel 1 (momentum)	3
	Channel 2 (stochastic K%)	2
	Channel 3 (RSI)	12
	Channel 4 (MACD)	3
	Channel 5 (LW %R)	6
	Channel 6 (A/D oscillator)	1
	Channel 7 (CCI)	4
Pooling layer	Pooling size	2

Table 6 Summary of experimental result

	ANN	CNN	GA-CNN
Training	60.67	71.69	75.95
Validation	58.40	69.94	74.69
Holdout	58.62	70.16	73.74

Finally, prediction accuracy for our proposed model presented 75.95% and 73.74% for the training and holdout data. Moreover, as shown in Fig. 13, the integrated GA-CNN approach converged faster and got higher performance than other comparative models. These results may be caused by the fact that the globally investigated CNN architecture increases the computational efficiency of the model. Thus, we can conclude that the simultaneous

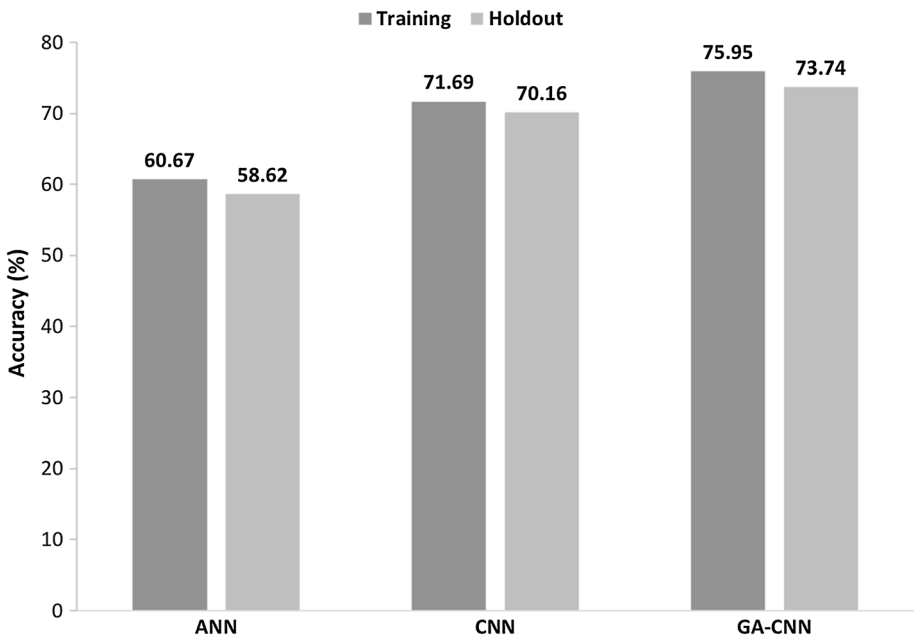

Fig. 10 Classification result for training and holdout set

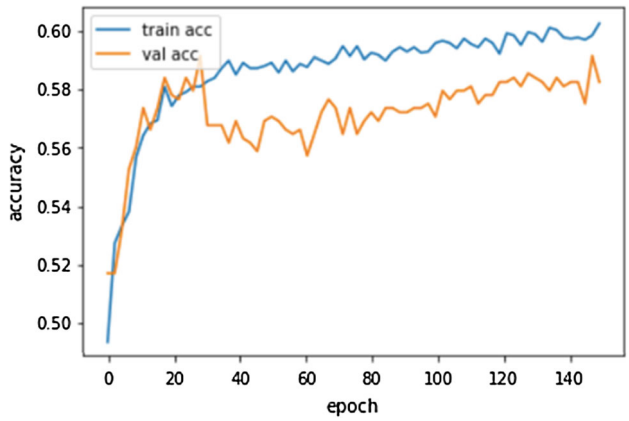

Fig. 11 Training and validation accuracy per epoch in ANN model

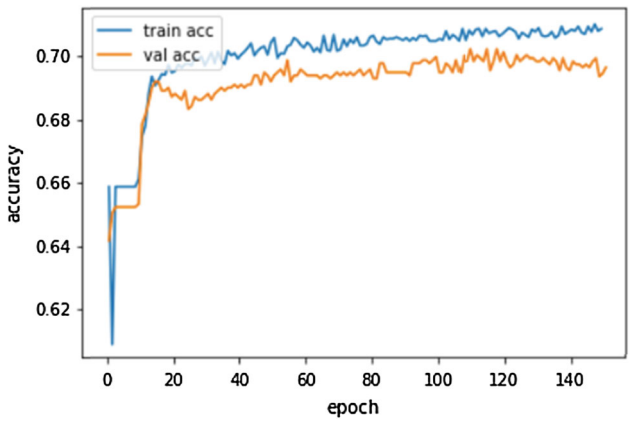

Fig. 12 Training and validation accuracy per epoch in standard CNN model

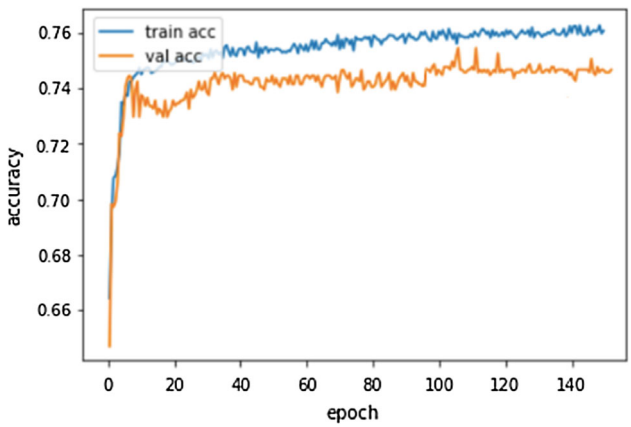

Fig. 13 Training and validation accuracy per epoch in GA-optimized CNN model

optimization of architectural factors of CNN using GA can be the most effective method for the stock market prediction.

We performed McNemar test to examine the proposed model's performance statistical significance compared to the benchmark models, as shown in Table 7. McNemar test is a representative nonparametric statistical technique that uses the Chi-squared distribution to find if statistically significant proportion changes have occurred on a dichotomous trait at two time points on the same population.

Table 7 McNemar test result for the comparison of the difference between models

	CNN	GA-CNN
ANN	0.000***	0.000***
CNN		0.005***

***Significant at 1% (ρ-value)

The predictive performance of proposed model shows statistically significant difference at the 1% significance level for both the ANN and standard CNN model, i.e., the proposed model performs significantly better than comparative models.

7 Conclusions

Stock market prediction has been widely studied in various fields, such as economics, mathematics, and statistics due to its practical importance. However, current statistical techniques and econometric models are limited to reflect nonlinear characteristics of financial time-series data. This study predicts KOSPI fluctuations using CNN, a popular deep learning technique that can learn complex correlation between data. However, various hyper-parameters within the CNN must be adjusted by the practitioner. Kernel size and number, and the pooling size are particularly important parameters because they directly affect the feature extraction process, which is the most important process during a CNN training. Previous studies have mostly used trial-and-error-based approaches with iterative experiments to determine network topology. Therefore, we propose a methodology to optimize CNN structure using GA, an evolutionary search algorithm, in an attempt to further enhance stock market prediction effectiveness.

Each CNN channel is composed of a sequence of technical indicators. We use a neural network model with two convolutional layers and one pooling layer. Optimized kernels are applied to the input data and optimal kernel size was simultaneously adopted for each layer. We compare the performance of the proposed model with standard CNN and ANN models and confirm superior predictive ability. The comparative ANN model is organized with one hidden layer containing 10 hidden neurons. ANN's classification accuracy at hold out data is 58.62% which the lowest performance. The standard (not optimized by GA) CNN model reached an accuracy of 70.16%, exhibiting a much better than the ANN model. Lastly, the GA-optimized

CNN model achieved a 73.74% accuracy at hold out data proving the effectiveness of proposed approach.

Moreover, we conducted the McNemar statistical tests to further explore the results. The test results can be interpreted as empirical proof that the difference in accuracy performance between the benchmark models and the proposed model is significant in a statistical manner. As we can see from the experimental results, optimizing the CNN structure using GA verifies the possibility of solving the problem effectively by flexible CNN application to noisy and nonlinear stock market data.

This study suggests an optimization method for deep learning techniques that could be effectively applied to stock market prediction, but there remains a margin of improvement. Firstly, GA has many controlling parameters that can have a great influence on the model's performance. However, this study used a fixed set of parameters selected based on past research due to the limitations of computing resources. In fact, GA-CNN model iterates the learning process of CNN whenever genetic evolution happens, so it requires considerable computing power. Thus, future research should extend the range of controlling parameters of GA to improve the prediction results. Second, CNN models include various parameters that can affect its predictive performance. Although this study optimized convolutional and pooling layers, further research can target other CNN model factors, such as learning parameters and parameters in fully connected layers.

Acknowledgements This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press, Cambridge
2. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Proceedings of the 26th conference on neural information processing systems (NIPS), pp 1097–1105
3. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1746–1751
4. Zheng Y, Liu Q, Chen E, Ge Y, Zhao JL (2016) Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. Front Comput Sci 10(1):96–112. https://doi.org/10.1007/11704-015-4478-2
5. Abd-Elazim SM, Ali ES (2016) Load frequency controller design via BAT algorithm for nonlinear interconnected power system. Int J Electr Power Energy Syst 77:166–177. https://doi.org/10.1016/j.ijepes.2015.11.029
6. Abd-Elazim SM, Ali ES (2016) Imperialist competitive algorithm for optimal STATCOM design in a multimachine power system. Int J Electr Power Energy Syst 76:136–146. https://doi.org/10.1016/j.ijepes.2015.09.004
7. Abd-Elazim SM, Ali ES (2018) Load frequency controller design of a two-area system composing of PV grid and thermal generator via firefly algorithm. Neural Comput Appl 30(2):607–616. https://doi.org/10.1007/s00521-016-2668-y
8. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539
9. Amin AE (2013) A novel classification model for cotton yarn quality based on trained neural network using genetic algorithm. Knowl Based Syst 39:124–132. https://doi.org/10.1016/j.knosys.2012.10.008
10. Donate JP, Li X, Sánchez GG, de Miguel AS (2013) Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. Neural Comput Appl 22(1):11–20. https://doi.org/10.1007/s00521-011-0741-0
11. Azadeh A, Mianaei HS, Asadzadeh SM, Saberi M, Sheikhalishahi M (2015) A flexible ANN-GA-multivariate algorithm for assessment and optimization of machinery productivity in complex production units. J Manuf Syst 35:46–75. https://doi.org/10.1016/j.jmsy.2014.11.007
12. Braun MA, Seijo S, Echanobe J, Shukla PK, del Campo I, Garcia-Sedano J, Schmeck H (2016) A neuro-genetic approach for modeling and optimizing a complex cogeneration process. Appl Soft Comput 48:347–358. https://doi.org/10.1016/j.asoc.2016.07.026
13. Armaghani DJ, Hasanipanah M, Mahdiyar A, Majid MZA, Amnieh HB, Tahir MM (2018) Airblast prediction through a hybrid genetic algorithm-ANN model. Neural Comput Appl 29(9):619–629. https://doi.org/10.1007/s00521-016-2598-8
14. Zhao M, Ren J, Ji L, Fu C, Li J, Zhou M (2012) Parameter selection of support vector machines and genetic algorithm based on change area search. Neural Comput Appl 21(1):1–8. https://doi.org/10.1007/s00521-011-0603-9
15. Ahmad I, Hussain M, Alghamdi A, Alelaiwi A (2014) Enhancing SVM performance in intrusion detection using optimal feature subset selection based on genetic principal components. Neural Comput Appl 24(7–8):1671–1682. https://doi.org/10.1007/s00521-013-1370-6
16. Raman MG, Somu N, Kirthivasan K, Liscano R, Sriram VS (2017) An efficient intrusion detection system based on hypergraph-Genetic algorithm for parameter optimization and feature selection in support vector machine. Knowl Based Syst 134:1–12. https://doi.org/10.1016/j.knosys.2017.07.005
17. Tao Z, Huiling L, Wenwen W, Xia Y (2019) GA-SVM based feature selection and parameter optimization in hospitalization expense modeling. Appl Soft Comput 75:323–332. https://doi.org/10.1016/j.asoc.2018.11.001
18. Shin KS, Han I (1999) Case-based reasoning supported by genetic algorithms for corporate bond rating. Expert Syst Appl 16(2):85–95. https://doi.org/10.1016/S0957-4174(98)00063-3
19. Ahn H, Kim KJ (2009) Bankruptcy prediction modeling with hybrid case-based reasoning and genetic algorithms approach. Appl Soft Comput 9(2):599–607. https://doi.org/10.1016/j.asoc.2008.08.002
20. Abu-Mostafa YS, Atiya AF (1996) Introduction to financial forecasting. Appl Intell 6(3):205–213. https://doi.org/10.1007/BF00126626
21. Fama EF (1970) Efficient capital markets: a review of theory and empirical work. J Finance 25(2):383–417. https://doi.org/10.2307/2325486

22. Huang W, Nakamori Y, Wang SY (2005) Forecasting stock market movement direction with support vector machine. Comput Oper Res 32(10):2513–2522. https://doi.org/10.1016/j.cor.2004.03.016
23. De Faria EL, Albuquerque MP, Gonzalez JL, Cavalcante JTP, Albuquerque MP (2009) Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods. Expert Syst Appl 36(10):12506–12509. https://doi.org/10.1016/j.eswa.2009.04.032
24. Babu CN, Reddy BE (2015) Prediction of selected Indian stock using a partitioning–interpolation based ARIMA–GARCH model. Appl Comput Inform 11(2):130–143. https://doi.org/10.1016/j.aci.2014.09.002
25. Cavalcante RC, Brasileiro RC, Souza VL, Nobrega JP, Oliveira AL (2016) Computational intelligence and financial markets: a survey and future directions. Expert Syst Appl 55:194–211. https://doi.org/10.1016/j.eswa.2016.02.006
26. Kim KJ, Han I (2000) Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert Syst Appl 19(2):125–132. https://doi.org/10.1016/s0957-4174(00)00027-0
27. Armano G, Marchesi M, Murru A (2005) A hybrid genetic-neural architecture for stock indexes forecasting. Inf Sci 170(1):3–33. https://doi.org/10.1016/j.ins.2003.03.023
28. Kara Y, Boyacioglu MA, Baykan ÖK (2011) Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul Stock Exchange. Expert Syst Appl 38(5):5311–5319. https://doi.org/10.1016/j.eswa.2010.10.027
29. Chong E, Han C, Park FC (2017) Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies. Expert Syst Appl 83:187–205. https://doi.org/10.1016/j.eswa.2017.04.030
30. Box GE, Jenkins GM (1976) Time series analysis: forecasting and control, revised edn. Holden-Day, Oakland
31. Engle RF (1982) Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica 50(4):987–1007. https://doi.org/10.2307/1912773
32. Schwaiger WS (1995) A note on GARCH predictable variances and stock market efficiency. J Bank Finance 19(5):949–953. https://doi.org/10.1016/0378-4266(94)00081-d
33. Wang JJ, Wang JZ, Zhang ZG, Guo SP (2012) Stock index forecasting based on a hybrid model. Omega 40(6):758–766. https://doi.org/10.1016/j.omega.2011.07.008
34. Wei LY, Chen TL, Ho TH (2011) A hybrid model based on adaptive-network-based fuzzy inference system to forecast Taiwan stock market. Expert Syst Appl 38(11):13625–13631. https://doi.org/10.1016/j.eswa.2011.04.127
35. Atsalakis GS, Valavanis KP (2009) Surveying stock market forecasting techniques–part II: soft computing methods. Expert Syst Appl 36(3):5932–5941. https://doi.org/10.1016/j.eswa.2008.07.006
36. Fernandez-Rodrıguez F, Gonzalez-Martel C, Sosvilla-Rivero S (2000) On the profitability of technical trading rules based on artificial neural networks: evidence from the Madrid stock market. Econ Lett 69(1):89–94. https://doi.org/10.1016/s0165-1765(00)00270-6
37. Tay FE, Cao L (2001) Application of support vector machines in financial time series forecasting. Omega 29(4):309–317. https://doi.org/10.1016/s0305-0483(01)00026-3
38. Lee MC (2009) Using support vector machine with a hybrid feature selection method to the stock trend prediction. Expert Syst Appl 36(8):10896–10904. https://doi.org/10.1016/j.eswa.2009.02.038
39. Adebiyi AA, Adewumi AO, Ayo CK (2014) Comparison of ARIMA and artificial neural networks models for stock price prediction. J Appl Math 2014:1–7. https://doi.org/10.1155/2014/614342
40. Saad EW, Prokhorov DV, Wunsch DC (1998) Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. IEEE Trans Neural Netw 9(6):1456–1470. https://doi.org/10.1109/72.728395
41. Yu H, Chen R, Zhang G (2014) A SVM stock selection model within PCA. Procedia Comput Sci 31:406–412. https://doi.org/10.1016/j.procs.2014.05.284
42. Kim KJ, Lee WB (2004) Stock market prediction using artificial neural networks with optimal feature transformation. Neural Comput Appl 13(3):255–260. https://doi.org/10.1007/s00521-004-0428-x
43. Lu CJ (2013) Hybridizing nonlinear independent component analysis and support vector regression with particle swarm optimization for stock index forecasting. Neural Comput Appl 23(7–8):2417–2427. https://doi.org/10.1007/s00521-012-1198-5
44. Kim MJ, Min SH, Han I (2006) An evolutionary approach to the combination of multiple classifiers to predict a stock price index. Expert Syst Appl 31(2):241–247. https://doi.org/10.1016/j.eswa.2005.09.020
45. Heaton JB, Polson NG, Witte JH (2017) Deep learning for finance: deep portfolios. Appl Stoch Models Bus Ind 33(1):19–21. https://doi.org/10.1002/asmb.2230
46. Fischer T, Krauss C (2018) Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res 270(2):654–669. https://doi.org/10.1016/j.ejor.2017.11.054
47. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791
48. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
49. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
50. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: Proceedings of the European conference on computer vision. Springer, pp 818–833
51. He F, Zhou J, Feng Z, Liu G, Yang Y (2019) A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with Bayesian optimization algorithm. Appl Energy 237:103–116. https://doi.org/10.1016/j.apenergy.2019.01.055
52. Bengio Y (2000) Gradient-based optimization of hyperparameters. Neural Comput 12(8):1889–1900. https://doi.org/10.1162/089976600300015187
53. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(1):281–305
54. Jaddi NS, Abdullah S, Hamdan AR (2016) A solution representation of genetic algorithm for neural network weights and structure. Inf Process Lett 116(1):22–25. https://doi.org/10.1016/j.ipl.2015.08.001
55. Tian D, Deng J, Vinod G, Santhosh TV, Tawfik H (2018) A constraint-based genetic algorithm for optimizing neural network architectures for detection of loss of coolant accidents of nuclear power plants. Neurocomputing 322:102–119. https://doi.org/10.1016/j.neucom.2018.09.014
56. Ciancio C, Ambrogio G, Gagliardi F, Musmanno R (2015) Heuristic techniques to optimize neural network architecture in manufacturing applications. Neural Comput Appl 27(7):2001–2015. https://doi.org/10.1007/s00521-015-1994-9
57. LeCun Y, Bengio Y (1995) Convolutional networks for images, speech, and time series. In: Arbib MA (ed) The handbook of brain theory and neural networks. MIT Press, Cambridge

58. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26. https://doi.org/10.1016/j.neucom.2016.12.038
59. Holland J (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
60. Kim HJ, Shin KS (2007) A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets. Appl Soft Comput 7(2):569–576. https://doi.org/10.1016/j.asoc.2006.03.004
61. Pal SK, Wang PP (1996) Genetic algorithms for pattern recognition. CRC Press, Boca Raton
62. Boureau YL, Ponce J, LeCun Y (2010) A theoretical analysis of feature pooling in visual recognition. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 111–118
63. Ali ES, Elazim SA (2018) Mine blast algorithm for environmental economic load dispatch with valve loading effect. Neural Comput Appl 30(1):261–270. https://doi.org/10.1007/s00521-016-2650-8
64. Simpson AR, Dandy GC, Murphy LJ (1994) Genetic algorithms compared to other techniques for pipe optimization. J Water Resour Plan Manag 120(4):423–443. https://doi.org/10.1061/(ASCE)0733-9496(1994)120:4(423)
65. Vanstone B, Finnie G (2009) An empirical methodology for developing stock market trading systems using artificial neural networks. Expert Syst Appl 36(3):6668–6680. https://doi.org/10.1016/j.eswa.2008.08.019
66. Kingma DP, Ba JL (2014) Adam: a method for stochastic optimization. In: Proceedings of the 3rd international conference on learning representation (ICLR)