*International Journal of Basic and Applied Sciences*, 14 (2) (2025) 68-77

**International Journal of Basic and Applied Sciences**

*Website: www.sciencepubco.com/index.php/IJBAS*
*https://doi.org/10.14419/7y9sax33*
***Research paper***



# Fine-tuning and Comparative Analysis of CNN Pre-trained Model for Stock Market Trend Prediction

**Jitendra Kumar Chauhan [1] *, Tanveer Ahmed [1], Amit Sinha [2]**

[1] *Computer Science Department, Bennett University, Plot Nos 8, 11, TechZone 2, Greater Noida, 201310, Uttar Pradesh, India*
[2] *Information Technology Department, ABES EC Ghaziabad, NH24, Ghaziabad, 201009, Uttar Pradesh, India*
**Corresponding author E-mail: E19SOE817@bennett.edu.in*

*Received: May 6, 2025, Accepted: May 22, 2025, Published: June 10, 2025*

**Abstract**

This manuscript examines the utilization of pre-trained Convolutional Neural Networks (CNNs), namely VGG16, ResNet50, InceptionV3, MobileNetV2, and Xception, for the prediction of stock market trends. The framework introduced herein hinges upon transforming financial time series into a 2D image. This transformation is done via the application of the Grammian Angular Field (GAF) and the Markov Transition Field (MTF). This image transformation allows for CNN compatibility, thereby optimizing the models for financial forecasting. The prediction of the trend is done in two ways. Initially, unmodified pre-trained CNNs are deployed for prediction. Subsequently, fine-tuning via the obtained financial images is done, and the results are recorded. The aim here is to augment the predictive performance substantially. The viability of the framework is assessed in the Indian Financial sector. Specifically, by focusing on twenty stocks from the NSE index: Nifty, we aim to quantify the performance of the proposed work. By experimenting on this dataset, we try to quantify how effectively pre-trained CNNs improve the competency of economic time series prediction. Moreover, the analysis presented in this article sheds further insights into future research and current procedures at the Intersection of Economics and Artificial Intelligence.

***Keywords***: *Fine Tuning; Stock Trend Classification, Trading; Convolutional Neural Network; GAF; MTF [42].*

## 1. Introduction

The art and science of stock price forecasting have been the focus of intense interest for both individuals and institutions participating in the financial markets for decades. The quest for more accurate forecasting methods has driven extensive research across multiple disciplines, including finance, economics, computer science, and statistics [1]. The desire for improved forecasting techniques extends beyond the realm of academia, as the implications of accurate predictions can translate directly into financial gain or loss. Accurate stock price forecasts hold the potential to guide investment decisions, enable timely market entry and exit, inform macroeconomic strategies, and even influence international business relationships. In essence, the ability to accurately predict stock prices can profoundly impact individual investors and the broader financial ecosystem. The stock markets operate as a complex and dynamically changing system influenced by a mass of variables that collectively form an intricate web of interactions. While the efficient market hypothesis suggests that stock prices already reflect all available information [2], the reality is something else. In this article, we aim to address this problem and try to apply deep learning-based methods to predict the trend in the financial time series.

The stock market is characterized by many factors, including volatility, trade decisions carried out emotionally, laws changing, and global occurrences like pandemics or clashes between countries, which all combine to form an intricate mix of dynamics [3]. Traditional economic concepts, while important, sometimes prove incompatible with this complexity. Models such as the Capital Asset Pricing Model (CAPM) generally account for market risk but overlook other sources of influence, including investor conduct and macroeconomic variability. While beneficial for pricing options, the Black-Scholes Model has trouble handling non-linearity and sudden changes in the market [4]. Classical time series models such as ARIMA (Autoregressive Integrated Moving Average) apply to economic forecasting, although they must assume that the data is stationary, which is not often the case in economic time series [5]. Even refined econometric models such as GARCH (Generalized Autoregressive Conditional Heteroskedasticity) find it hard to cope with drastic modifications in the information, like extreme values and structural changes [6]. Although these models are helpful, they may not completely represent the intricacies of financial markets, especially in unstable and uncertain periods.

The introduction of machine learning, intense learning, has ushered in a novel age of predictive pricing. Convolutional Neural Networks (CNNs), specifically, have shown astoundingly efficient outcomes across a broad variety of objectives, including image recognition, natural language processing, and even some initial steps toward economics [7]. However, applying these models to economic data presents issues, as it commonly requires considerable labeled data and computing capacity. This is where the notion of pre-trained models is significant. Models such as VGG16, ResNet50, and InceptionV3 have been trained on immense datasets including ImageNet, and have

**Copyright © Jitendra Kumar Chauhan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.**

been indicated to be immensely efficient in dissimilar vision responsibilities [8 - 10]. The fundamental idea of pre-trained models is that they embody knowledge from past studies, offering a strong foundation for analyzing renewed, analogous data.
The work presented in this article seeks to pave a new path by utilizing pre-trained models for forecasting stock prices. The article tries to classify the trend in a financial time series into the Buy, Sell, and Hold categories. To do this, the Markov transformation field (MTF) and the Grammian angular field (GAF) ([11] are used to transform raw financial data into images. The images are then fed to the CNN in two different ways. First, the images are fed into pre-trained models. For this purpose, we have chosen VGG16, ResNet50, InceptionV3, MobileNetV2, and Xception [42] as the models to experiment with. Second, we fine-tune each of the pre-trained models on the image data we create. To create the image data, we also use a novel image creation algorithm tailored for the financial time series. To test the feasibility of the proposed work, we experimented on twenty different stocks of the Indian Financial sector. To the best of our knowledge, these results are the first wherein we try to quantify the performance of the models on such a large scale. The results show that the work presented in this article shows good performance. It should be noted here that the goal of this article is to show the applicability of pre-trained CNNs in forecasting financial trends.

## 2. Literature review

The financial world has always pursued more accurate forecasting models, from traditional methods like basic technical analysis to more data-driven methods using machine learning techniques. Time series models like ARIMA support financial forecasting but face setbacks due to their design [12]. In recent years, decision trees, support vector machines, and their machine learning algorithms resemble various types of neural networks [13], but these methods often require extensive feature engineering and can be sensitive to economic noise. Previously, traditional convolutional neural networks (CNNs), originally developed for image-processing tasks, especially in image-processing applications [14], widely accepted models such as VGG16, ResNet50, and INCEPTION have been trained on large datasets such as ImageNet to develop new performance standards. Transfer learning, the process of transferring insights from one domain to another, has enabled more efficient and robust computational applications [8 - 10].
Knowledge transfer has revolutionized our utilization of Convolutional Neural Networks across a wide array of domains, such as economics. Modifying pre-trained models through fine-tuning supplies a practical resolution to the challenge of acquiring correctly specific and computationally efficient models. This approach has been successfully applied to medical imaging [15], natural language processing (Devlin et al., 2018) [16], and economic applications [17]. Innovations that transform time series records into picturesque forms, such as Grammian Angular Fields (GAF) and Markov Transition Fields (MTF), have enabled Convolutional Neural Networks to properly identify investment statistics (Wang and Oates, 2015) [18]. This alteration is indispensable for the effective utilization of CNNs in financial time-series data, as convolution layers can recognize key traits within stock price trends. Although research on perfecting pre-trained CNNs for economic purposes is gaining momentum, it is still in its early stages. Previous efforts have mainly focused on resource selection and risk perception, exhibiting the efficiency of transfer learning [19]. Nonetheless, the rare nuances of the financial markets, comprising nonlinearity, high unpredictability, and noise, present exceptional obstacles that necessitate imaginative solutions. One such method is the idea of integration, which proposes an alternative plan to combat these issues [20]. Judging the outcomes of micro-adjusted models in economics is more than just accuracy. Metrics like precision, recall, F1-score, and computational effectiveness are imperative for building the robustness and applicability of the model in practical economic scenarios [21].
The integration of pre-trained CNNs and financial forecasting symbolizes an encouraging frontier in predictive modeling [22]. By exploiting the data-shift capabilities of pre-programmed models and utilizing exceptional data transformation techniques, the peculiarities posed by financial statistics can be addressed. The concept of transfer learning, a cornerstone of contemporary machine learning, can redefine the layout of financial forecasting [23]. The thought of using knowledge gleaned from one domain to improve performance in another mirrors the belief in productive knowledge transfer. In the financial realm, pre-trained CNNs, initially created for image-related tasks, can be revamped to extract valuable revelations from sophisticated financial data [24]. This strategy not only restricts the requirement for intense feature engineering but also accelerates model creation. Fine-tuning, a technique often applied in transfer learning, allows us to adjust pre-trained models to domain-specific objectives [25]. In the scope of financial forecasting, fine-tuning involves making minor modifications to the pre-trained CNN's weights to harmonize it with the intricacies of financial data [26]. This endeavor maintains the treasured knowledge embedded in the pre-trained model while personalizing it to the nuances of financial markets.
One of the pivotal issues in utilizing CNNs for financial data is the inherent temporal nature of time series data. Financial time series data, normally including past stock prices, is fundamentally consecutive. To make this information compatible with CNNs, progressive data transformation techniques have appeared. Grammian Angular Fields (GAF) and Markov Transition Fields (MTF) are two such approaches that have gained approval. These methods efficiently convert serial financial data into image-like frameworks, permitting convolutional layers to draw out applicable features [27] [28].
Despite the encouraging advances in using pre-trained CNNs and data transformation procedures for fiscal forecasting, multiple impediments persist. Financial markets are characterized by their intricacy and exposure to a wide selection of factors, such as geopolitical events, investor sentiment, and economic policy transformations. These components can generate commotion and nonlinearity, which may present probs for prognostic models [29]. Financial markets regularly display unfavorable behavior, in which little shifts in input variables can bring about substantive variances in output. Though pre-trained CNNs are prevalent instruments for trait extraction, handling unpredictable trends is yet a challenge [30]. Researchers in the realm of fiscal forecasting are evaluating the assimilation of nonlinear activation capacities and specialized constructions to catch and model nonlinear relationships aptly. Volatility is a distinctive quality of financial markets. Unanticipated upheavals, regularly triggered by abrupt affairs, can confront the sturdiness of prognostic models. Scientists are examining procedures to advance model stability under high volatility, including developing ensemble models and embracing volatility-specific features. Financial markets generate extensive amounts of data, which incorporates not only stock prices but also financial flags, reports, attitudes, and social media murmurings. Accurately joining and processing this assorted information is a main difficulty. Advanced methodologies in natural language processing (NLP) and emotion examination are being applied to pick out valuable knowledge from unstructured textual data sources, enriching the load features for predictive models [31].
Fine-tuning Convolutional Neural Networks (CNNs) is a pivotal technique in the field of deep learning, allowing the adaptation of pre-trained models to domain-specific tasks [22]. This approach has revolutionized various domains, including computer vision, natural language processing, and more. Fine-tuning mitigates the need for training deep networks from scratch, making it a resource-efficient choice for model development. Below, we explore the key facets and trends in fine-tuning CNNs, accompanied by a selection of 10 references that contribute to this dynamic field. Transfer Learning: Fine-tuning is a manifestation of transfer learning, enabling knowledge transfer from one domain to another [32]. Architecture Selection: CNN architectures like VGG, ResNet, and Inception have been employed for

fine-tuning, with architecture choice driven by the specific task [14]. Data Augmentation: Techniques such as data augmentation, including rotation, flipping, and cropping, enhance the diversity of training data (Shorten [33]. Regularization: To prevent overfitting during fine-tuning, regularization methods like dropout and weight decay are applied [34]. Domain Adaptation: Fine-tuning is instrumental in domain adaptation tasks, where models trained in one domain are adjusted for a different domain [35]. Hyperparameter Tuning: Fine-tuning demands effective hyperparameter tuning, often achieved through grid search or random search [36]. Applications: Fine-tuning is widely used in tasks like image classification, object detection, sentiment analysis, and more [37]. Deep Learning Frameworks: Frameworks such as TensorFlow, PyTorch, and Keras offer tools to facilitate fine-tuning [38]. Resource Efficiency: Fine-tuning streamlines model development, reducing computational costs [39]. Challenges: Challenges in fine-tuning encompass architecture selection, handling domain shifts, and addressing data scarcity [40].

[41] Demonstrated the performance of a hybrid CNN-LSTM model in financial and maritime market trend prediction. Their experiment validated that such models have very high prediction accuracy and excellent generalization performance on various datasets, rendering them an appropriate benchmark to evaluate deep-learning-based financial prediction systems. This indicates the performance of hybrid models as alternative or complementary options to traditional CNN-based image transformation methods like GAF and MTF used in this research.

In conclusion, fine-tuning CNNs has evolved into a fundamental technique, enabling efficient model adaptation and transfer of knowledge across domains. As deep learning continues to advance, fine-tuning will remain a key element in building state-of-the-art machine learning systems. The amalgamation of pre-trained CNNs and data transformation processes with financial forecasting stands to extend the veracity and robustness of prognostic models. Transference learning grants the possibility to leverage expertise from one area to another, cutting back on the need for descriptive feature engineering. Adjusting pre-trained models allows us to modify them to the distinct features of financial data, reconciling generalization and domain loyalty. Data transmogrification surgeries enable the correct utilization of CNNS for financial time series data, paving the path for updated feature extraction. Despite difficulties remaining, including coping with nonlinearity, dealing with extreme volatility, and tackling heterogeneous data resources, sustained research attempts are continuing to aid in the field's development.

Based on these literature gaps, this research proposes a CNN pre-trained model for stock market trend prediction.

## 3. Proposed methodology

This section expands upon the details of the proposed approach. The overall methodology has been divided into four main parts. They are as follows [42]:

1) The first subsection explains the model architecture in detail.
2) In the second subsection, explain the CNN architecture in detail.
3) The third section discusses labeling the stock's closing price.
4) The last section describes the process of image generation.

### 3.1. Model architecture

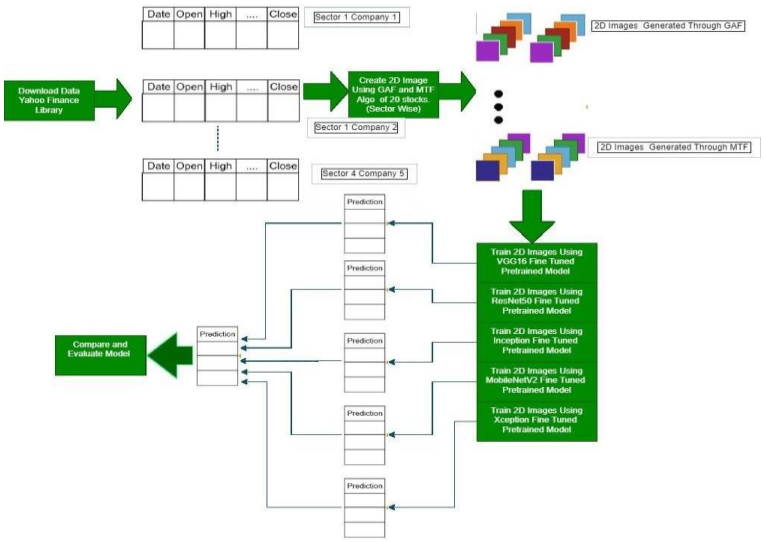



**Fig. 1:** Proposed Framework for Stock Trend Classification Using A Fine-Tuned Pre-Trained CNN Model.

### 3.2. Labeling of stock's close price

Fig. 2: Stock Label Algorithm

Algorithm 1 Stock Price Labeling Algorithm

The Provided Stock Price Labeling Algorithm is designed to analyze a time series of stock prices and assign labels of "Buy," "Sell," or "Hold" to different data points. It takes two key inputs: the time series data of stock prices (TS) and a user-defined window size parameter (WindowSize), which defaults to 15 if not specified. The algorithm begins by initializing an empty array called LabeledData to store the resulting labels. It then enters a loop that iterates through the time series data. Within each iteration, the algorithm creates a rolling window of data with a size equal to WindowSize. It calculates the midpoint, maximum price (MaxPrice), and minimum price (MinPrice) within this window.

Require:

Time series data: TS (an array of stock prices) Window size: Window Size (default: 15)

Ensure:
Labeled stock price data: Labeled Data (an array of labels)
1: function LABELSTOCKPRICES(TS, Window Size)
2: Initialize an empty array Labeled Data
3: for i from 0 to N − Window Size do
4: Window ← TS[i : i + Window Size]
5: Midpoint ← Window[Window Size//2]
6: Max Price ← max(Window)
7: Min Price ← min(Window)
8: if Midpoint = Max Price then
9: Append ”Sell” to Labeled Data 10: else if Midpoint = Min Price then 11: Append ”Buy” to Labeled Data 12: else
13: Append ”Hold” to Labeled Data
14: end if
15: end for
16: return Labeled Data
17: end function

### 3.3. Image generation

In this article, we focused our efforts on forecasting price trends. To do this, we converted stock prices into pictures. In this case, we used Grammian angular fields (GAF) and Markov transition fields (MTF). Both methods put time-collection data into picks in a way that gives a time cost based on the total pattern embedded in the series and make those patterns more visible to algorithms and detection devices.

#### 3.3.1. Grammian angular field (GAF)

Algorithm 2: Convert Time Series Data to GAF Image
Require:
Time series data: TS (an array of numerical values) Number of time points: N
Image size: Image Size
Ensure:
GAF image: GAF Image (a 2D array)
1: function CONVERTTOGAF(TS, N, Image Size)
2: Min Value ← min(TS)
3: Max Value ← max(TS)
4: Normalized TS ← (TS − Min Value)/(Max V alue − Min V alue)
5: Create an empty 2D array GAF Image of size (Image Size, Image Size)
6: Time Points ← evenly spaced points between 0 and N-1 with Image Size intervals
7: for i from 0 to Image Size − 1 do
8: for j from 0 to Image Size − 1 do
9: Cosine Value ← cos(Normalized TS[i] − Normalized TS[j]) 10: Sine Value ← sin(Normalized TS[i] − Normalized TS[j]) 11: GAF Image[i][j] ← [Cosine V alue, Sine Value]
12: end for
13: end for
14: return GAF Image
15: end function

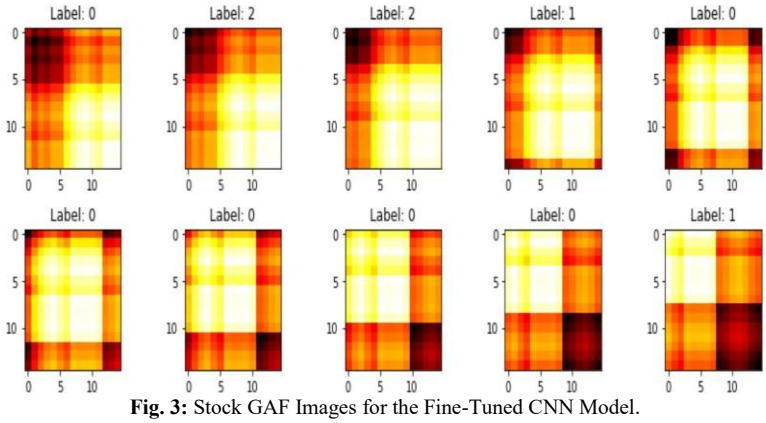


**Fig. 3:** Stock GAF Images for the Fine-Tuned CNN Model.

#### 3.3.2. Markov transition field (MTF)

Algorithm 3 Convert Time Series Data to MTF Image
Require:
Time series data: TS (an array of numerical values) Number of time points: N
Image size: Image Size
Ensure:
MTF image: MTF Image (a 2D array)

```
function CONVERTTOMTF(TS, N, Image_Size)
    Min_Value ← min(TS)
    Max_Value ← max(TS)
    Normalized_TS ← (TS − Min_Value)/(Max_Value − Min_Value)
    Create an empty 2D array MTF_Image of size (Image_Size, Image_Size)
    Divide the range [0, 1] into Image_Size bins
    for i from 0 to N − 1 do
        for j from 0 to N − 1 do
            Bin_i ← find the bin index for Normalized_TS[i]
            Bin_j ← find the bin index for Normalized_TS[j]
            MTF_Image[Bin_i][Bin_j] ← MTF_Image[Bin_i][Bin_j] + 1
        end for
    end for
    return MTF_Image
end function
```

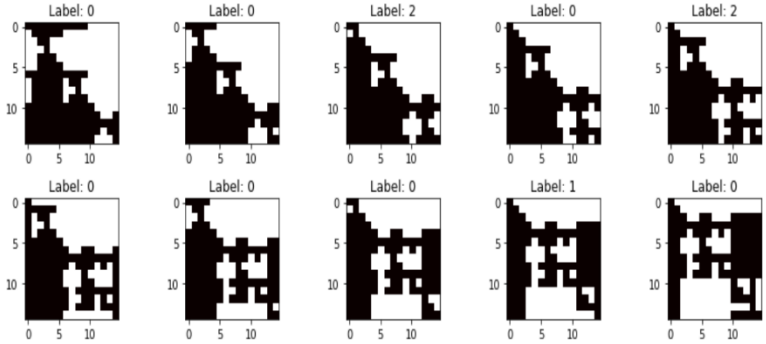



**Fig. 4:** Stock MTF Images for the Fine-Tuned CNN Model.

While this study employs Gramian Angular Fields (GAF) and Markov Transition Fields (MTF) in transforming financial time series into image data to be utilized in CNN-based classification, other orthogonal transformation techniques can be applied to learn more and enhance model performance. Recurrence Plots (RPs) are a robust nonlinear tool that represents recurring states in a time series and hence can be utilized to identify intricate temporal patterns, regime changes, or chaotic dynamics common in finance data. Continuous Wavelet Transform (CWT) is a powerful tool that analyzes time series data by decomposing it into localized frequency components across various scales.

In contrast to Fourier-based methods, CWT can identify transient events as well as non-stationary patterns and is therefore a suitable tool for identifying sudden price increases or decreases. Visibility Graphs is the third tool, which transforms time-series data into complex networks by connecting points under a geometric visibility constraint. This method retains the temporal structure and uncovers concealed topological structures, which can be transformed into graph-based images to be used as input for CNN. Combining these complementary techniques with GAF and MTF may provide richer feature representations and stronger models for stock trend prediction.

## 4. Results

In this section, we expand upon the results of the proposed method.

### 4.1. Dataset and evaluation criterion

The dataset for this research includes the daily stock rates for the following Indian stocks over the last five years: Experimentation with Different Pretrained Models.

#### 4.1.1. GAF images results with different fine-tuned pretrained models

**Table 1:** Results from BAJAJ AUTO Stocks Images through GAF

| Image Method | Finetuned Pretrained Model | Label | Precision | Recall | F1-Score | Overall Accuracy |
|---|---|---|---|---|---|---|
| GAF | VGG16 | Hold | 0.59 | 0.37 | 0.46 | 0.64 |
| | | Buy | 0.61 | 0.85 | 0.71 | – |
| | | Sell | 0.8 | 0.53 | 0.61 | – |
| | ResNet50 | Hold | 0.5 | 0.02 | 0.03 | 0.56 |
| | | Buy | 0.6 | 0.83 | 0.7 | – |
| | | Sell | 0.47 | 0.62 | 0.54 | – |
| | InceptionV3 | Hold | 0.49 | 0.58 | 0.53 | 0.59 |
| | | Buy | 0.73 | 0.59 | 0.65 | – |
| | | Sell | 0.51 | 0.6 | 0.55 | – |
| | MobileNetV2 | Hold | 0.45 | 0.21 | 0.29 | 0.5 |
| | | Buy | 0.5 | 0.92 | 0.65 | – |
| | | Sell | 0 | 0 | 0 | – |
| | Xception | Hold | 0.47 | 0.66 | 0.55 | 0.57 |
| | | Buy | 0.74 | 0.54 | 0.62 | – |
| | | Sell | 0.49 | 0.53 | 0.51 | – |

A few key patterns emerged in analyzing the performance of various pre-trained models on stock image [42] data processed through Grammian Angular Field (GAF). InceptionV3 consistently demonstrated balanced performance across all categories—Bajaj Auto, ONGC, and the Average of 20 Stocks, making it a versatile choice for different types of stock data. On the other hand, VGG16 stood out for its proficiency in classifying 'Buy' labels but faltered when it came to 'Hold' and 'Sell' categories. MobileNetV2 showcased an intriguing duality; it led to overall accuracy for Ongc stocks but had noticeable shortcomings, particularly in identifying 'Sell' labels across all datasets. ResNet50 struggled to make its mark, underperforming in most scenarios and notably lagging in classifying 'Hold' labels in the Bajaj Auto dataset. Finally, Xception turned out to be the best; it posted moderate to good scores across all metrics but never took the lead in any area. Therefore, the choice of a model could depend significantly on what one prioritizes: overall accuracy, balanced classification, or specialization in a specific stock label like 'Hold,' 'Buy,' or 'Sell.'

**Table 2:** Results from ONGC Stocks Images through GAF

| Image Method | Finetuned Pretrained Model | Label | Precision | Recall | F1-Score | Overall Accuracy |
|---|---|---|---|---|---|---|
| GAF | VGG16 | Hold | 0.44 | 0.29 | 0.35 | 0.6 |
| | | Buy | 0.62 | 0.83 | 0.71 | – |
| | | Sell | 0.66 | 0.46 | 0.54 | – |
| | ResNet50 | Hold | 0.43 | 0.39 | 0.41 | 0.5 |
| | | Buy | 0.73 | 0.56 | 0.63 | – |
| | | Sell | 0.33 | 0.51 | 0.4 | – |
| | InceptionV3 | Hold | 0.53 | 0.37 | 0.44 | 0.57 |
| | | Buy | 0.65 | 0.71 | 0.68 | – |
| | | Sell | 0.44 | 0.51 | 0.48 | – |
| | MobileNetV2 | Hold | 0.58 | 0.59 | 0.58 | 0.61 |
| | | Buy | 0.64 | 0.87 | 0.74 | – |
| | | Sell | 0.53 | 0.17 | 0.26 | – |
| | Xception | Hold | 0.5 | 0.57 | 0.53 | 0.57 |
| | | Buy | 0.69 | 0.59 | 0.64 | – |
| | | Sell | 0.46 | 0.51 | 0.48 | – |

**Table 3:** Results from an Average of 20 Stocks Images through GAF [42]

| Image Method | Finetuned Pretrained Model | Label | Precision | Recall | F1-Score | Overall Accuracy |
|---|---|---|---|---|---|---|
| GAF | VGG16 | Hold | 0.38 | 0.32 | 0.32 | 0.52 |
| | | Buy | 0.54 | 0.73 | 0.61 | – |
| | | Sell | 0.4 | 0.33 | 0.34 | – |
| | ResNet50 | Hold | 0.36 | 0.39 | 0.33 | 0.45 |
| | | Buy | 0.6 | 0.52 | 0.53 | – |
| | | Sell | 0.41 | 0.36 | 0.31 | – |
| | InceptionV3 | Hold | 0.47 | 0.51 | 0.48 | 0.56 |
| | | Buy | 0.66 | 0.65 | 0.65 | – |
| | | Sell | 0.5 | 0.47 | 0.48 | – |
| | MobileNetV2 | Hold | 0.5 | 0.27 | 0.29 | 0.52 |
| | | Buy | 0.57 | 0.76 | 0.64 | – |
| | | Sell | 0.4 | 0.26 | 0.28 | – |
| | Xception | Hold | 0.5 | 0.4 | 0.42 | 0.55 |
| | | Buy | 0.65 | 0.65 | 0.65 | – |
| | | Sell | 0.5 | 0.5 | 0.5 | – |

### 4.1.2. MTF images results with different fine-tuned pretrained models

**Table 4:** Results from BRITTANIA Stocks Images through MTF [42]

| Image Method | Finetuned Pretrained Model | Label | Precision | Recall | F1-Score | Overall Accuracy |
|---|---|---|---|---|---|---|
| MTF | VGG16 | Hold | 0.79 | 0.56 | 0.65 | 0.72 |
| | | Buy | 0.74 | 0.79 | 0.76 | – |
| | | Sell | 0.64 | 0.78 | 0.7 | – |
| | ResNet50 | Hold | 0.65 | 0.52 | 0.58 | 0.65 |
| | | Buy | 0.66 | 0.7 | 0.67 | – |
| | | Sell | 0.64 | 0.71 | 0.67 | – |
| | InceptionV3 | Hold | 0.77 | 0.59 | 0.67 | 0.7 |
| | | Buy | 0.71 | 0.86 | 0.78 | – |
| | | Sell | 0.62 | 0.54 | 0.58 | – |
| | MobileNetV2 | Hold | 0.67 | 0.44 | 0.53 | 0.65 |
| | | Buy | 0.65 | 0.85 | 0.74 | – |
| | | Sell | 0.64 | 0.55 | 0.59 | – |
| | Xception | Hold | 0.71 | 0.56 | 0.63 | 0.64 |
| | | Buy | 0.69 | 0.67 | 0.68 | – |
| | | Sell | 0.54 | 0.69 | 0.61 | – |

Analyzing the performance of various pre-trained models on stock image data processed through the Markov Transition Field (MTF) for different stocks [42] such as Brittania, Infy, and an average of 20 stocks provides us with interesting insights. VGG16 shines in the Brittania dataset with notable precision in the 'Buy' and 'Sell' labels, scoring an overall accuracy of 0.72, but seems less consistent in INFY and the average dataset. ResNet50 generally trails, especially in the Brittania dataset, with a lackluster overall accuracy of 0.65. It particularly struggles with 'Hold' labels across the board.

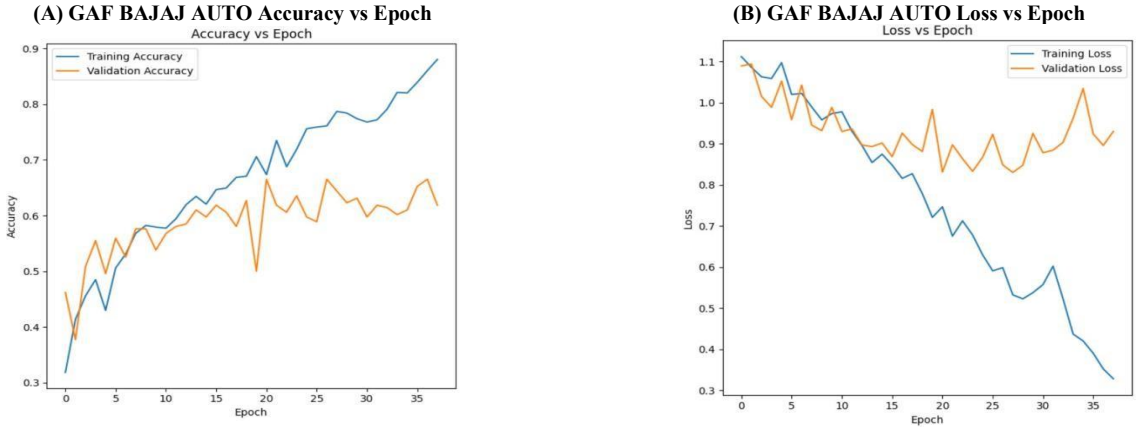



**Fig. 5:** Results from GAF Bajaj Auto.

**Table 5:** Results from INFY Stocks Images through MTF [42]

| Image Method | Finetuned Pretrained Model | Label | Precision | Recall | F1-Score | Overall Accuracy |
|---|---|---|---|---|---|---|
| MTF | VGG16 | Hold | 0.74 | 0.48 | 0.58 | 0.68 |
| | | Buy | 0.74 | 0.74 | 0.74 | – |
| | | Sell | 0.58 | 0.78 | 0.67 | – |
| | ResNet50 | Hold | 0.57 | 0.52 | 0.54 | 0.6 |
| | | Buy | 0.72 | 0.53 | 0.61 | – |
| | | Sell | 0.51 | 0.8 | 0.62 | – |
| | InceptionV3 | Hold | 0.65 | 0.67 | 0.66 | 0.71 |
| | | Buy | 0.76 | 0.76 | 0.76 | – |
| | | Sell | 0.67 | 0.65 | 0.66 | – |
| | MobileNetV2 | Hold | 0.67 | 0.29 | 0.41 | 0.65 |
| | | Buy | 0.61 | 0.9 | 0.73 | – |
| | | Sell | 0.81 | 0.54 | 0.65 | – |
| | Xception | Hold | 0.68 | 0.48 | 0.56 | 0.72 |
| | | Buy | 0.7 | 0.86 | 0.77 | – |
| | | Sell | 0.8 | 0.7 | 0.74 | – |

InceptionV3 stands out for its balanced performance, particularly in the INFY dataset, where it achieves an overall accuracy of 0.71. MobileNetV2 is a mixed bag; for example, it struggles with 'Hold' labels in the BRITTANIA dataset but excels in 'Buy' labels for INFY. Xception remains a reliable all-rounder, posting decent metrics across all labels and datasets, but never leading the pack. Its performance is notably consistent in the INFY dataset, excelling in both 'Buy' and 'Sell' labels.
In the average of 20 stocks dataset, InceptionV3 and Xception both post moderate to good scores in all metrics, but again, neither takes a decisive lead. MobileNetV2 notably falls short in this average dataset, especially when identifying 'Hold' and 'Sell' labels. Overall, choosing a model would be a matter of prioritizing specific performance metrics such as precision, recall, F1-Score, or overall accuracy, as each model has its strengths and weaknesses across these datasets.

### 4.1.3. Comparison with and without fine-tuning

In analyzing the accuracy of various deep learning architectures for a given task, it is evident that finetuning generally provides a performance boost, although the magnitude of this improvement varies by model. VGG16, for example, shows a noticeable uptick in accuracy from 0.61 to 0.65 when finetuned. Conversely, ResNet50's accuracy slightly declines after finetuning, going from 0.59 to 0.57, suggesting that its initial architecture may already be well-suited for the task. Among all models, InceptionV3 benefits the most from finetuning, boasting a substantial increase in accuracy from 0.55 to 0.67. MobileNetV2 maintains a consistent performance level before and after finetuning, both at 0.58 accuracy, implying that finetuning might not be necessary for this specific model. Lastly,

**Table 6:** Results from an Average of 20 Stock Images through MTF [42]

| Image Method | Finetuned Pretrained Model | Label | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|---|---|
| MTF | VGG16 | Hold | 0.64 | 0.54 | 0.58 | 0.65 |
| | | Buy | 0.69 | 0.73 | 0.7 | – |
| | | Sell | 0.62 | 0.61 | 0.61 | – |
| | ResNet50 | Hold | 0.5 | 0.64 | 0.54 | 0.57 |
| | | Buy | 0.69 | 0.52 | 0.58 | – |
| | | Sell | 0.57 | 0.6 | 0.57 | – |
| | InceptionV3 | Hold | 0.63 | 0.64 | 0.63 | 0.67 |
| | | Buy | 0.71 | 0.73 | 0.71 | – |
| | | Sell | 0.66 | 0.6 | 0.63 | – |
| | MobileNetV2 | Hold | 0.56 | 0.46 | 0.44 | 0.58 |
| | | Buy | 0.63 | 0.73 | 0.66 | – |
| | | Sell | 0.67 | 0.44 | 0.5 | – |
| | Xception | Hold | 0.61 | 0.54 | 0.57 | 0.64 |
| | | Buy | 0.68 | 0.71 | 0.7 | – |
| | | Sell | 0.64 | 0.62 | 0.62 | – |

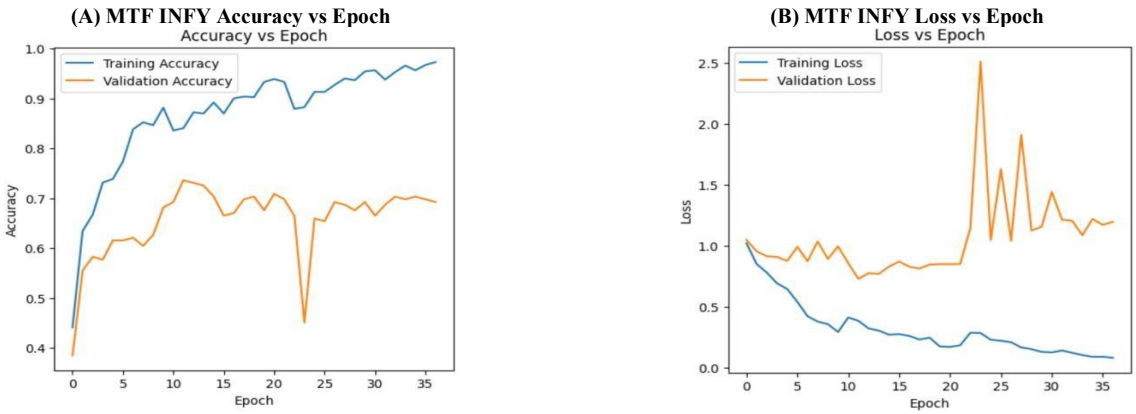



**Fig. 6:** Results from MTF INFY.

Exception sees a moderate accuracy boost from 0.57 to 0.64 when finetuned. These findings underscore the importance of choosing the right architecture and considering the merits of finetuning based on the specific needs and constraints of a given application."

**Table 7:** Model Comparison with and without Finetuning in MTF images [42]

| | | Without Finetuning | | | | | With Finetuning | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Image type | Model | Label | P | R | F1 | Accuracy | Precision | Recall | F1-Score | Accuracy |
| | | Hold | 0.57 | 0.47 | 0.49 | 0.61 | 0.64 | 0.54 | 0.58 | 0.65 |
| MTF | VGG16 | Buy | 0.64 | 0.73 | 0.68 | – | 0.69 | 0.73 | 0.7 | – |
| | | Sell | 0.62 | 0.52 | 0.55 | – | 0.62 | 0.61 | 0.61 | – |
| | | Hold | 0.64 | 0.36 | 0.42 | 0.59 | 0.5 | 0.64 | 0.54 | 0.57 |
| MTF | ResNet50 | Buy | 0.6 | 0.8 | 0.68 | – | 0.69 | 0.52 | 0.58 | – |
| | | Sell | 0.63 | 0.42 | 0.47 | – | 0.57 | 0.6 | 0.57 | – |
| | | Hold | 0.55 | 0.49 | 0.5 | 0.55 | 0.63 | 0.64 | 0.63 | 0.67 |
| MTF | InceptionV3 | Buy | 0.61 | 0.66 | 0.62 | – | 0.71 | 0.73 | 0.71 | – |
| | | Sell | 0.5 | 0.43 | 0.44 | – | 0.66 | 0.6 | 0.63 | – |
| | | Hold | 0.56 | 0.41 | 0.46 | 0.58 | 0.56 | 0.46 | 0.44 | 0.58 |
| MTF | MobileNetV2 | Buy | 0.65 | 0.76 | 0.66 | – | 0.63 | 0.73 | 0.66 | – |
| | | Sell | 0.63 | 0.41 | 0.46 | – | 0.67 | 0.44 | 0.5 | – |
| | | Hold | 0.56 | 0.42 | 0.46 | 0.57 | 0.61 | 0.54 | 0.57 | 0.64 |
| MTF | Xception | Buy | 0.61 | 0.73 | 0.66 | – | 0.68 | 0.71 | 0.7 | – |
| | | Sell | 0.57 | 0.44 | 0.47 | – | 0.64 | 0.62 | 0.62 | – |

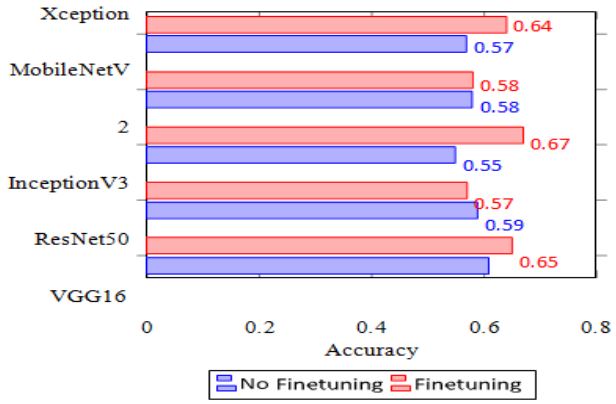



**Fig. 7:** Accuracy Comparison for Different Models Model Accuracy with and Without Finetuning.

**Table 8:** Model Comparison with and without Finetuning in GAF Image

| Image Type | Model | Label | Accur P | racy wi R | without Fi F1 | tuning Accuracy | Accu P | racy w R | With Fin F1 | e Tuning Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Hold | 0.45 | 0.33 | 0.36 | 0.54 | 0.38 | 0.32 | 0.32 | 0.52 |
| GAF | VGG16 | Buy | 0.61 | 0.78 | 0.67 | – | 0.54 | 0.73 | 0.61 | – |
| | | Sell | 0.51 | 0.3 | 0.33 | – | 0.4 | 0.33 | 0.34 | – |
| | | Hold | 0.44 | 0.31 | 0.3 | 0.49 | 0.36 | 0.39 | 0.33 | 0.45 |
| GAF | ResNet50 | Buy | 0.58 | 0.69 | 0.61 | – | 0.6 | 0.52 | 0.53 | – |
| | | Sell | 0.35 | 0.3 | 0.29 | – | 0.41 | 0.36 | 0.31 | – |
| | | Hold | 0.43 | 0.81 | 0.3 | 0.5 | 0.47 | 0.51 | 0.48 | 0.56 |
| GAF | InceptionV3 | Buy | 0.57 | 0.75 | 0.63 | – | 0.66 | 0.65 | 0.65 | – |
| | | Sell | 0.47 | 0.3 | 0.32 | – | 0.5 | 0.47 | 0.48 | – |
| | | Hold | 0.47 | 0.24 | 0.28 | 0.52 | 0.5 | 0.27 | 0.29 | 0.52 |
| GAF | MobileNetV2 | Buy | 0.58 | 0.81 | 0.66 | – | 0.57 | 0.76 | 0.64 | – |
| | | Sell | 0.48 | 0.29 | 0.33 | – | 0.4 | 0.26 | 0.28 | – |
| | | Hold | 0.43 | 0.32 | 0.34 | 0.52 | 0.5 | 0.4 | 0.42 | 0.55 |
| GAF | Xception | Buy | 0.58 | 0.76 | 0.65 | – | 0.65 | 0.65 | 0.65 | – |
| | | Sell | 0.48 | 0.29 | 0.34 | – | 0.5 | 0.5 | 0.5 | – |

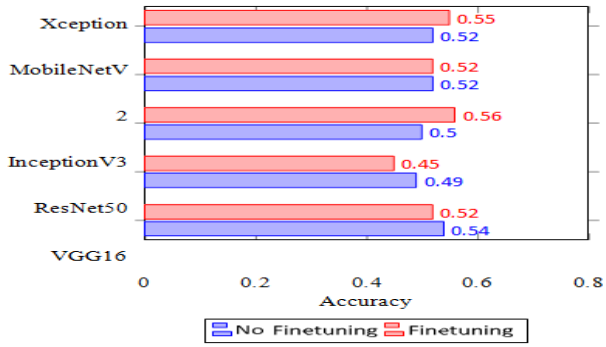



**Fig. 8:** Accuracy Comparison for Different Models Model Accuracy with and without Finetuning.

The figures presented in Figs 7 and 8 capture a nuanced comparison of overall model accuracy, both with and without the application of fine-tuning. Remarkably, the effect of fine-tuning varies quite distinctly across different combinations of image-creation methods and pre-trained models. For instance, InceptionV3 experiences a noticeable improvement in its performance metrics, as its accuracy ascends from an initial 0.5 to 0.56 after fine-tuning. Similarly, the Xception model's accuracy receives a modest but positive bump, rising from 0.52 to 0.55 post-fine-tuning.

Conversely, the GAF-VGG16 and ResNet50 models see a slight but palpable dip in their performance after fine-tuning, with their accuracies descending from 0.54 to 0.52 and from 0.49 to 0.45, respectively. Intriguingly, the MobileNetV2 model remains unaffected by the fine-tuning process, maintaining its accuracy at 0.52.

The data underscores that fine-tuning is not a one-size-fits-all solution for enhancing model performance. While certain model configurations stand to gain from the fine-tuning process, others might experience an adverse impact. As such, the decision to implement fine-tuning must be judiciously considered, factoring in the specific intricacies and performance benchmarks of each unique model configuration.

The way that certain CNN architectures handle patterns makes them more effective for financial data. InceptionV3, for instance, detects both short-term and long-term trends in stock photos by simultaneously using various filter sizes. This is helpful since patterns of varying lengths are frequently found in financial data. Another powerful model that can learn intricate features with less processing power is Xception, which works well with big financial datasets. ResNet50, on the other hand, is useful for deep networks since it retains crucial information across multiple layers, but it might not work as well for more local patterns. Accurate predictions can be made by selecting the CNN model that best fits the characteristics of financial data.

## 5. Conclusion

This paper presents an application of pre-trained CNNs for predicting the trends in a financial time series. Specifically, VGG16, ResNet50, InceptionV3, MobileNetV2, and Xception were used to predict the trend in a financial time series. The numerical time series was converted into an Image using the Grammian Angular Field and Markov Transition Field. The pre-trained models were also fine-tuned, and their performance was optimized even further. The operational feasibility of these adapted CNN models is rigorously assessed in the Indian Financial sector. This assessment is specifically targeted at twenty diverse stocks from the NSE index: Nifty. We showed that in our experiments, the model InceptionV3 performed better compared to other models. Although the margin of improvement was small, nevertheless, InceptionV3 emerged as a consistently strong performance model. Specifically, the prediction accuracy came out to be 67% on twenty different stocks. The model's prediction accuracy of 67% on 20 stocks was better than chance. Despite these promising studies like [43] shows that such accuracy can only generate profits when combined with a tested trading strategy. Real-world gains are also influenced by market conditions and transaction costs.

This figure shows that in the long run, such models would result in considerable profit following the Law of Large Numbers. The analysis presented in the article further augments the existing knowledge base by offering substantial insights for future research endeavors and contemporary methodologies operating at the confluence of Economics and Artificial Intelligence. Furthermore, this article tried to present a new guideline for improving the performance of stock trend prediction using advanced CNN-based models.

## 6. Declaration of Competing Interest

Data sharing does not apply to this article as no datasets were generated or analyzed during the current study.

## References

[1] Fama, E.F.: Efficient capital markets: A review of theory and empirical work. Journal of Finance 25, 383–417 (1970) https://doi.org/10.2307/2325486.
[2] Malkiel, B.G.: A Random Walk Down Wall Street (1973).
[3] Shefrin, H.: Behavioral Corporate Finance: Decisions that Create Value. McGraw-Hill/Irwin, (2005).
[4] Black, F., Scholes, M.: The pricing of options and corporate liabilities. Journal of Political Economy 81, 637–654 (1973) https://doi.org/10.1086/260062.
[5] Abhyankar, A., Copeland, L., Wong, W.: Nonlinear dynamics in real-time equity market indices: Evidence from the United Kingdom. The Economic Journal 107, 864–880 (1997) https://doi.org/10.2307/2235155.
[6] Hartman, P., Hlinka, P.: Structural breaks in financial time series. Finance Research Letters 24, 66–73 (2018).
[7] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521, 436–444 (2015) https://doi.org/10.1038/nature14539.
[8] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014)
[9] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778 (2015) https://doi.org/10.1109/CVPR.2016.90.

[10] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2818–2826 (2016) https://doi.org/10.1109/CVPR.2016.308.
[11] Wang, Z., Oates, T.: Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI) (2015).
[12] Box, G.E., Jenkins, G.M.: Time Series Analysis: Forecasting and Control. Holden-Day, (1976).
[13] Atsalakis, G.S., Valavanis, K.P.: Surveying stock market forecasting techniques—part ii: Soft computing methods. Expert Systems with Applications 36(3), 5932–5941 (2009) https://doi.org/10.1016/j.eswa.2008.07.006.
[14] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012).
[15] Shin, H.-C., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., Summers, R.M.: Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. IEEE Transactions on medical imaging 35(5), 1285–1298 (2016) https://doi.org/10.1109/TMI.2016.2528162.
[16] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
[17] Dixon, M., Klabjan, D., Jin Hoon, B.: Sequence classification for credit scoring using deep learning. arXiv preprint arXiv:1607.02470 (2016).
[18] Wang, Z., Oates, T.: Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. AAAI Workshop: Time Series, 40–46 (2015).
[19] Sezer, O.B., Ozbayoglu, M.: Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. Applied Soft Computing 70, 525–538 (2018) https://doi.org/10.1016/j.asoc.2018.04.024
[20] Bao, W., Yue, J., Rao, Y.: A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLOS ONE 12(7), 0180944 (2017) https://doi.org/10.1371/journal.pone.0180944.
[21] Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. Proceedings of the 24th International Conference on Artificial Intelligence, 2327–2333 (2015).
[22] Zhang, Y., Zhao, J., Leung, H.F.: Stock price prediction with big data: A deep learning approach. IEEE Transactions on Cybernetics 49(11), 3958–3970 (2019).
[23] Fischer, T., Krauss, C.: Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research 270(2), 654–669 (2018).
[24] Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), pp. 2327–2333 (2015).
[25] Guo, X., Sharma, A., Rissanen, J.: Stock price prediction with a hybrid model using multiple sources. Expert Systems with Applications 95, 43–54 (2018).
[26] Kwon, Y.K., Moon, B.R.: Stock price prediction using deep learning and sentiment analysis. Expert Systems with Applications 129, 150–168 (2019).
[27] Kim, Y.D., Choi, J., Kim, Y.J., Kim, C.H.: Stock price prediction using cnn-lstm neural net- works. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN), pp. 483–490 (2016).
[28] Akita, R., Yoshihara, A., Matsubara, T.: Stock price prediction with deep temporal cnn-lstm networks. Expert Systems with Applications 141, 112953 (2020).
[29] Zhang, Y., Zhang, Y., Liu, Y., Dai, Q.: Deep learning for financial market prediction. In: Pro- ceedings of the European Conference on Computer Vision (ECCV) Workshops, pp. 391–407 (2017).
[30] Lahmiri, S., Bekiros, S., Al-Yahyaee, K.H.: Stock market forecasting: A systematic literature review. Expert Systems with Applications 129, 418–450 (2019).
[31] Yoon, J.H., Swales, G.D., Kim, Y.S.: Predicting stock prices using convolutional neural networks. In: Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI), pp. 1102–1107 (2018).
[32] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010) https://doi.org/10.1109/TKDE.2009.191
[33] Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of Big Data 6(1), 60 (2019) https://doi.org/10.1186/s40537-019-0197-0.
[34] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014).
[35] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Lempitsky, V.: Domain-adversarial training of neural networks. Journal of Machine Learning Research 17(1), 2096–2030 (2016).
[36] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13(1), 281–305 (2012)
[37] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 770–778 (2017) https://doi.org/10.1109/CVPR.2016.90.
[38] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Zheng, X.: Tensorflow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 265–283 (2016).
[39] Yam, S.S., Ang Jr, M.H., Ng, S.K., Tan, K.: Efficient fine-tuning of deep convolutional neural networks with transferred fisher scores for object recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 475–484 (2017).
[40] Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: An astounding baseline for recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 806–813 (2014) https://doi.org/10.1109/CVPRW.2014.131.
[41] Han, Z., Zhu, X., & Su, Z. (2024). Forecasting Maritime and Financial Market Trends: Leveraging CNN-LSTM Models for Sustainable Shipping and China's Financial Market Integration. *Sustainability*, *16*(22), 9853. https://doi.org/10.3390/su16229853
[42] Chauhan, J. K., Ahmed, T., & Sinha, A. (2023, December). Comparative Analysis of CNN Pre-trained Model for Stock Market Trend Prediction. In *International Conference on Recent Trends in Image Processing and Pattern Recognition* (pp. 110-129). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-53082-1_10.
[43] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, *270*(2), 654-669. https://doi.org/10.1016/j.ejor.2017.11.054.