Original Research Article

Enhancing Agricultural Commodity Price Forecasting Using Generative Models: A Deep Learning Approach

ABSTRACT

Predicting stock market prices remains one of the most critical and challenging tasks in finance. Technical analysis, a key methodology in investment theory, involves forecasting price movements by analyzing historical market data. Recently, deep learning has achieved significant success across various domains due to its powerful data processing capabilities. It is increasingly applied in financial areas such as stock market prediction, portfolio optimization, financial information processing, and trade execution strategies. This study explores a novel architecture that integrates a Generative Adversarial Network (GAN) with a Convolutional Neural Network (CNN) as the discriminator and Gated Recurrent Units (GRU) as the generator. This architecture generates distributions of daily stock data through an adversarial learning system to forecast closing stock prices. An empirical study was conducted using daily data from Ruchi Soya Industries Limited, which provided a broad range of trading days for analysis. The experimental results demonstrated that this novel GAN architecture offers promising performance in predicting closing prices, outperforming other deep learning models such as GRU, LSTM and Bi-LSTM.

Keywords:*Bidirectional LSTM (Bi-LSTM), Convolutional Neural Network (CNN), Discriminator, Gated Recurrent Unit (GRU), Generative models, Generator, Long Short-Term Memory (LSTM)*

1. INTRODUCTION

Predicting stock prices is notoriously challenging due to the inherent complexity and chaotic dynamics of financial markets, compounded by numerous non-decidable, nonstationary stochastic variables involved (Marszalek and Burczynski, 2014). Researchers from various fields have long studied historical financial time series patterns, proposing a variety of methods to forecast stock prices. Achieving accurate predictions typically requires the careful selection of input variables, the development of predictive models grounded in professional financial expertise, and the application of sophisticated statistical methods for arbitrage analysis. These complexities often make it difficult for those outside the financial domain to effectively utilize these methods for stock price prediction (Huang et al., 2014; Wang et al., 2017; Chong et al., 2017).

Recent studies have utilized machine learning (ML) and deep learning (DL) models to predict agricultural prices across various vegetable crops, as examined by Avinash et al. (2023a, 2023b), Nayak et al. (2024a, 2024b), Vinay et al. (2024), Baishya et al. (2023), and Singh et al. (2023). Generative Adversarial Networks (GANs), introduced by Goodfellow et al. (2014), have also shown promise in generating image patches from random noise using two simultaneously trained networks. Specifically, in a GAN, the discriminative network (D) learns to distinguish between real and generated data, while the generative network (G) aims to fool D by generating high-quality data. Although GANs have been successfully applied across various fields, such as image inpainting, semantic segmentation, and video prediction, their application in stock forecasting is relatively recent, as demonstrated by Iizuka et al. (2017). To date, there has been no attempt to apply GANs to agricultural market prediction.

This study leverages basic technical index data as input variables, which can be easily obtained from trading software, enabling individuals outside the financial sector to predict stock prices using the proposed GAN model. This approach addresses challenges faced by deep generative models, particularly their difficulty in approximating many intractable probabilistic computations arising from maximum likelihood estimation and related strategies, and the challenge of leveraging piecewise linear units in a generative context. Within the adversarial framework, the generative model competes

against an adversary: a discriminative model that learns to distinguish between model-generated samples and real data. This dynamic is analogous to counterfeiters attempting to produce undetectable fake currency while law enforcement strives to identify it. The competition between these models drives both to improve continuously until the generated data is indistinguishable from the real data.The remainder of this paper is structured as follows: Section 2 provides a review of the literature on algorithms used for financial market prediction. Section 3 formulates the problem and details the General Adversarial Network (GAN) framework. The experiments section presents the experimental analysis using the proposed model, along with a comparison of the results with those from classical prediction models. Finally, Sections 4 and 5 discuss the results obtained from real datasets and present the study's conclusions and references.

2. MATERIAL AND METHODS

2.1 GENESIS OF GAN

Related work falls into two categories: econometric and soft computing models. Econometric models like AR, MA, ARMA, and ARIMA (Brockwell & Devis, 2013) forecast by treating new signals as noisy combinations of recent signals but rely on assumptions like i.i.d. noise, which GARCH models address by predicting conditional variances.Soft computing models, inspired by AI, include ANN (Kara et al., 2011), FL (Hassan et al., 2009), SVM (Huang et al., 2005), and PSO (Majhi et al., 2008). Deep neural networks (Rather et al., 2015; Chong et al., 2017) effectively predict high-frequency financial time series, while Chen et al. (2017) use a double-layer network for stock return dependencies.These methods often need expert constraints, unlike the proposed model, which directly uses trading software data. Algorithms like RNN, LSTM, and GRU are widely used in time-series forecasting (Gao et al., 2022).GANs, developed by Goodfellow et al. (2014) for image generation, have been adapted for sequential data, improving stock price prediction through the adversarial generator-discriminator relationship. Some of these studies are summarized in Table 1 below.

Table 1: Review of Time series data and modelling tasks using GAN

Applications	Task	Values in task	Model	Evaluation methods
RCGAN (Esteban *et al.,* 2017)	Generation	Medical data	GAN and RNN	TSTR and TRTS
Grid-GAN (Zhang*et al.,* 2018)	Generation	Smart grid data	CGAN and CNN	TSTR and TRTS
EEG-GAN (Hartmann *et al.,* 2018)	Generation	EEG brain signals	WGAN and CNN	IS, FID, and ED
StockGAN (Zhou *et al.,* 2018)	Generation	Stock data	GAN, CNN and LSTM	RMSRE, DPA
GRU-GAN (Luo *et al.,* 2018)	Imputation	Medical records, meteorologic data	GAN and GRU	Imputation accuracy
ForGAN (Koochali*et al.,* 2019)	Generation	Synthetic series and internet traffic	CGAN and LSTM	KL divergence
TimeGAN (Yoon *et al.,* 2019)	Generation	Sines, stocks, energy and events data	GAN	Diversity, fidelity(*e.g.*, RMSRE, DPA)
E2GAN (Luo *et al.,* 2019)	Imputation	Medical records, meteorologic data	GAN and GRU	Imputation accuracy
SimGAN (Golany*et al.,* 2020)	Generation	Heart rate ECG signals	GAN	Prediction accuracy
Ad-Attack (Dang-Nhu *et al.,* 2020)	Generation	Stock prices and electricity data	GAN	Domain metrics (*e.g.*, attack success rate, returned of perturbed portfolio)

2.2 THEORETICAL BACKGROUND

The conventional deep learning approaches for time series forecasting such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU)and Long Short Term Memory (LSTM) were compared with the GAN architectures in time series modelling for various applications are discussed subsequently.

2.2.1Convolutional Neural Network (CNN)

CNN is a type of deep, feed-forward neural network commonly used to analyse visual imagery. A typical CNN model is composed of an input layer, an output layer, and some hidden layers. Compared to the traditional multilayer perceptron (MLP), CNNs can develop internal representations of two-dimensional images, allowing CNNs to be used more generally on other types of data with spatial correlations. Though CNNs are not specifically developed for non-image data, it has been widely used in time series sequential data forecasting problem by LeCunn *et al.* (2015).

2.2.2Recurrent Neural Network (RNN)

It is a type of neural network where the previous outputs are fed as the input to the current step. The advantage of RNN is the hidden state (internal memory) that captures information calculated so far in a sequence.

Though the RNNs work effectively in many application domains, they may suffer from a problem called vanishing gradients Li *et al*. (2018) To cope with this problem, two variants of RNN have been developed: Long Short-Term Memory (LSTM) by Hochreiter and Schmidhuber (1997) and Gated Recurrent Units (GRU) networks by Kyunghym *et al.* (2014). LSTM is capable of learning long-term dependencies with a special memory unit. An LSTM cell has three gates (forget gate, input gate, and output gate) to regulate the information flow. Compared with standard LSTM models, GRU has fewer parameters, which combines the input gate and the forget gate into an “update gate” and merges the cell state and hidden state**.** RNN, LSTM, and GRU are widely used to learn the temporal correlations of time series and Spatio-Temporal (ST) data.

2.2.3.1 Gated recurrent unit (GRU)

Gated recurrent unit (GRU) is a kind of RNN that uses gating mechanisms to control the flow of information between cells in the neural network derived from LSTM and was introduced in 2014 by Kyunghyun Cho *et al.* (1997). GRU is composed of two gates, an update gate and a reset gate. These gates are used to filter out what information should remain and what should be disposed of. Different from traditional RNN, GRUs solve the vanishing and exploding gradient problems. Unlike LSTM, GRU has fewer parameters than LSTM due to the lack of one gate. Another difference is that GRUs also lack the cell state from LSTM so that GRU can only store both long and short-term memory in the hidden state. Recently, GRUs have been shown to perform better than LSTM on certain smaller and less frequent datasets. Fig. 1 shows the internal architecture of a GRU unit cell.

2.2.3.2 Long-Short term memory (LSTM)

Long short-term memory (LSTM) is a specific recurrent neural network (RNN) architecture. It was proposed (Hochreiter *et al*., 1997). Unlike a traditional feed-forward neural network, it includes feedback connections. Furthermore, it can be utilized on single-point data and the sequence of data as well. The essential components of LSTM are an input gate, an output gate and a forget gate, and the LSTM network was developed to resolve the vanishing gradient problem while training the traditional RNNs. LSTM is a cell memory unit that means that LSTM can remove or add information to the cell state. LSTM has overcome the vanishing gradients and the exploding gradients problem that appeared in RNN through the units' specific internal structure built in the model. Nowadays, LSTM has been known as a powerful method capable of processing, classifying, and making predictions based on time series data.

There are three gate controls: input gate (i_t), output gate (o_t), and forget gate (f_t) in LSTM cell. The structure of the LSTM unit is shown in Fig.2

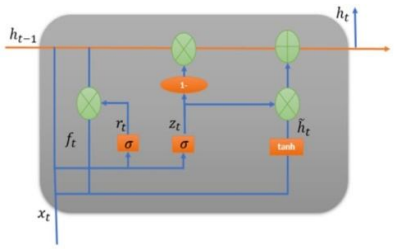

Fig. 1: Architecture of the GRU

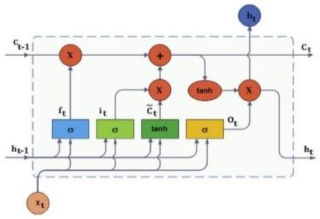

Fig. 2: Architecture of the LSTM

2.2.3.3 Bidirectional LSTM (Bi-LSTM)

The bidirectional LSTM (Bi-LSTM) model is shown in Fig. 3 which includes a forward LSTM layer and a backward LSTM layer. $\vec{h}_t$and $\overleftarrow{h}_t$is used to denote the LSTM hidden vector of the forward LSTM layer and backward LSTM layer at time t, respectively. As shown in Fig. 3$\vec{h}_t$ and $\overleftarrow{h}_t$are independent of each other and are only related to their respective LSTM layers. The corresponding output of Bi-LSTM (y_t) is obtained by the weighted connection of these two hidden layers. The process can be described as:

$$\vec{h}_t = LSTM\left(x_t, \vec{h}_{t-1}\right)$$

$\overleftarrow{h}_t$=LSTM $\left(x_t, \overleftarrow{h}_{t+1}\right)$

$$y_t = \delta\left(W_{\vec{h}_y}\vec{h}_t + W_{\overleftarrow{h}_t}\overleftarrow{h}_t + b_y\right)$$

where LSTM(.), represents LSTM network, $W_{\vec{h}_y}$ and $W_{\overleftarrow{h}_t}$) y represent the weight of the forward and backward LSTM layer at time t, respectively. by denotes the bias of the output layer, δ(.) represents the activation function.

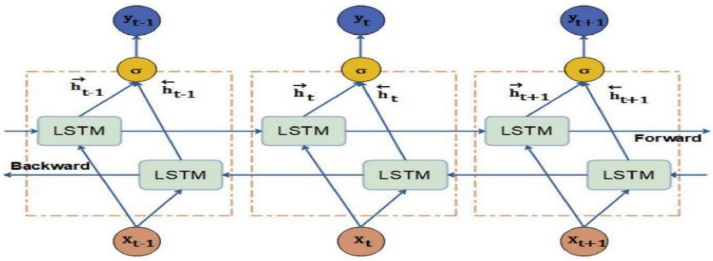

Fig. 3: Architecture of the Bi-LSTM

2.3GENERATIVE ADVERSARIAL NETWORKS (GANs)

2.3.1 Basic Idea/Principles of GANs:

GAN is a new framework which trains two models like a zero-sum game by Goodfellow *et al.* (2014). In the adversarial process, the generator can be seen as a cheater/fool to generate the similar data as the real data, while the discriminator plays the role of judge to distinguish the real data and generated data. They can reach an ideal point that the discriminator is unable to differentiate the two types of data. At this point, the generator can capture the data distributions from this game. Based on this principle, we propose our GAN architecture for the prediction of stock closing price.

2.3.2 The Architecture of GAN

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptron. To learn the generator's distribution p_g over data x, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as G (z; θ_g), where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron D (x; θ_d) that outputs a single scalar. D(x) represents the probability that x came

from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize log (1− D(G(z))). In other words, D and G play the following two-player minimax game with value function V (G, D):

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_{z(z)}}[\log(1 - D(G(z)))]$$

In this case adversarial nets, essentially showing that the training criterion allows one to recover the data generating distribution as G and D are given enough capacity depicted in Fig.4. In practice, there must be implementation of the game using an iterative, numerical approach. Optimizing D to completion in the inner loop of training is computationally prohibitive, and on finite datasets would result in overfitting. Instead, we alternate between k steps of optimizing D and one step of optimizing G. This results in D being maintained near its optimal solution, so long as G changes slowly enough. The procedure is formally presented in Algorithm 1 and the loss function for both the discriminator and generator as given below.

$$L_D = -E_{x\sim P_{data}(x)}[\log D(x)] - E_{Z\sim P_{z(Z)}}[\log(1 - D(G(z)))] \text{ and } L_G = E_{Z\sim P_{z(Z)}}[\log(1 - D(G(z)))]$$

In practice, above equation may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, log (1 − D(G(z))) saturates. Rather than training G to minimize log (1 − D(G(z))) we can train G to maximize log D(G(z)). This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning. The generator G implicitly defines a probability distribution p_g as the distribution of the samples G(z) obtained when z ~p_z. Therefore, we would like Algorithm 1 to converge to a good estimator of p_{data}, if given enough capacity and training time and finally this minimax game has a global optimum for p_g = p_{data}. Finally, the generator and the discriminator are trained iteratively, and the two continue to play games, which has achieved the goal of the predicted data generated by the generator as close to the actual data as possible.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used k = 1, the least expensive option, in our experiments.

for number of training iterations do
for k steps do
- Sample minibatch of m noise samples $\{z_{(1)}, \ldots, z_{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x_{(1)}, \ldots, x_{(m)}\}$ from data generating distribution $p_{data}(x)$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$$

end for
- Sample minibatch of m noise samples $\{z_{(1)}, \ldots, z_{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^{(i)})))]$$

end for
The gradient-based updates can use any standard gradient-based learning rule.

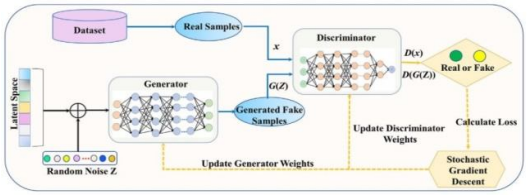

Fig. 4: Architecture of the GAN (Huang et al. 2022)

3.. RESULTS AND DISCUSSION
EMPIRICAL STUDY

3.1 Data description

In this study, the daily prices of soya oil (in INR) were used as the experimental dataset. The price series data, spanning from July 30, 2010, to June 30, 2020, were sourced from the Yahoo Finance stock market trading website (https://finance.yahoo.com). The descriptive statistics of the soya stock price series used in this study are summarized in Table 2, and a time plot illustrating the data is shown in Figure 5. The time plot confirms the non-stationarity and nonlinearity of the series, which are further validated through statistical tests as presented in Table 2. The soya stock prices range from Rs. 4,306 to Rs. 14,105, with a standard deviation of Rs. 3,580.47, indicating significant variability within the dataset. Additionally, the data exhibit a positive skewness of 0.8619 and a platykurtic value of 2.6052, suggesting a non-normal distribution, which is confirmed by the Jarque-Bera test results in Table 2.

The dataset comprises 2,498 observations, divided into training (80%) and testing (20%) sets. The training dataset includes 1,998 observations, while the testing dataset contains 500 observations, which were used for post-sample predictions.

The objective of this study is to develop a GAN model for forecasting agricultural commodity prices and compare its performance with other forecasting models, including LSTM, Bi-LSTM, and GRU. The software was executed on a system with the following specifications: Intel Core i5-10500 CPU, 2.50 GHz, 8 GB RAM, and Intel(R) UHD Graphics 10 GB.

Table 2:Descriptive statistics of Soya stock price series (in INR)

Descriptive statistics	Price (in INR)
Minimum	17.00
Mean	4306.00
Maximum	14105.00
Standard deviation	3580.47
CV (%)	83.15
Skewness	0.88
Kurtosis	2.60
Jarque-Bera	339.95**

3.2 Test for Stationarity

A stationary series is characterized by a mean and variance that remain constant over time. To determine whether the soya price series is stationary, the Augmented Dickey-Fuller (ADF), Phillip-Perron (PP), and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests were applied. The ADF test's null hypothesis posits that the time series contains a unit root, indicating non-stationarity. In this case study, the soya price series failed to reject the null hypothesis. Similarly, the PP test, which assumes the time series is integrated of order 1, also confirmed non-stationarity, as shown in Table 3.

Table 3: ADF, PP and KPSS test results of daily price series of soya stock

Price series	ADF test		PP test		KPSS test	
	Statistic	p-value	Statistic	value	Statistic	p-value
Soya stock price series	-2.2800	0.2260	-3.42	0.28	9.05	0.01

3.3 Brock-Dechert- Scheinkman (BDS) test for nonlinearity

The BDS test is employed to assess nonlinearity in the time series. The null hypothesis asserts that the series is independent and identically distributed. The test results, presented in Table 4, indicate that the p-values calculated at points ranging from 0.50 to 2.00 confirm the nonlinearity of the soya stock market price data, particularly for embedding dimensions (number of lags) 2 and 3.

Table 4: BDS test for non-linearity

Epsilon for close points	Embedding dimensions		p- value
	2	3	
0.5 σ	208.08	344.99	<0.0001
1.0 σ	154.16	186.64	<0.0001
1.5 σ	146.38	160.76	<0.0001
2.0 σ	131.97	132.33	<0.0001

3.4 Data pre-processing and normalization

The data series did not contain any missing values, so no imputation was required. However, normalization was necessary to ensure effective fitting and extrapolation of neural network models. The normalization technique involved rescaling the data values between 0 and 1, using the following formula: $X_t' = (X_t - X_{min}) / (X_{max} - X_{min})$where X_{min}, X_{max} and X_t are the minimum, maximum and observation at time t, respectively and $X_t'$is the rescaled value. represent the minimum, maximum, and observation at time t. In Python, the MinMaxScaler function from the Scikit-learn package was used for this purpose.

3.5 Implementation of forecasting models

Given the confirmed non-stationarity and nonlinearity of the time series, various forecasting models were developed and fitted using Python's TensorFlow and Keras libraries. The GRU model, tuned using a grid search approach, was configured with 128 input neurons, 64 hidden neurons in the first layer, and 32 neurons in the second hidden layer. Utilizing the Adam optimizer, this model, with a batch size of 128 and 50 epochs, achieved an RMSE of 741.63. Similarly, the LSTM model, also optimized through grid search, yielded a test data RMSE of 449.72, performing well on test data but less effectively overall compared to GRU. The Bi-LSTM model further improved on these results, recording an RMSE of 388.04, outperforming both the GRU and LSTM on test data.

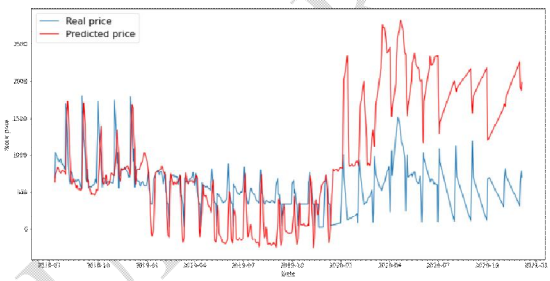

Fig. 5 (a): GRU model on test data

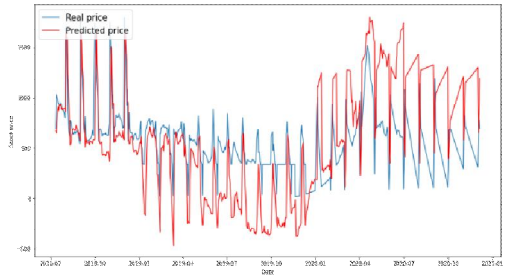

Fig. 5 (b): LSTM model on test data

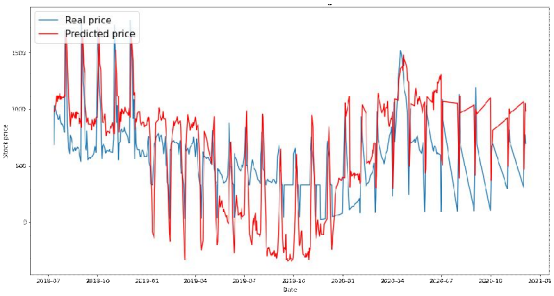

Fig. 5 (c): Bi-LSTM model on test data

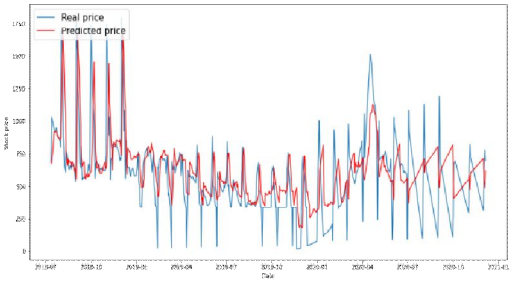

Fig. 5 (d): GAN model on test data

Table 5: Optimized results obtained by different models for the Soya price series data

Models	RMSE
GRU	741.63
LSTM	449.72
Bi-LSTM	388.04
GAN	**251.60**

Building on the strengths of these models, the GAN model was designed to incorporate both GRU's and Bi-LSTM's capabilities. The GAN's generator employed three layers of GRU with 1024, 512, and 256 neurons, while the discriminator, a Convolutional Neural Network (CNN), was tasked with distinguishing between real and generated data. The discriminator included three 1D convolution layers with 32, 64, and 128 neurons, followed by three dense layers with 220, 220, and 1 neuron. The Leaky ReLU activation function was used throughout, except in the output layer, which utilized the Sigmoid function. The GAN model demonstrated superior performance, achieving an RMSE of 251.60, outperforming traditional models by minimizing both forecast error and direction prediction loss. Bayesian Optimization was employed to fine-tune key hyperparameters, further enhancing the model's accuracy. This architecture, termed GAN-FD (GAN for minimizing forecast error loss and direction prediction loss), effectively combines these losses to produce accurate predictions.

4. CONCLUSIONS

This paper introduces a novel GAN model designed to generate predicted soya stock price series data through the generator component of the GAN network. This approach marks a pioneering effort in applying GANs to time series forecasting within the agricultural sector. The model effectively incorporates the uncertainties of soya stock prices by constructing a generator that leverages a new type of deep neural network, accounting for noise sequences in the latent space. Through adversarial training, the model achieves superior prediction accuracy compared to traditional methods, demonstrating its effectiveness in agricultural stock price forecasting. However, challenges remain in tuning hyperparameters, particularly within GAN models that include RNNs, which can lead to instability if not optimally adjusted. Future research should focus on developing advanced hyperparameter optimization techniques, such as reinforcement learning, to enhance the predictive performance of GAN models in this context.

REFERENCES

Araujo, R.D. A., Oliveira, A. L. and Meira, S. (2015). A hybrid model for high-frequency stock market forecasting. *Expert Systems with Applications*, **42(8)**, 4081-4096.

Avinash, G., Ramasubramanian, V., Ray, M., Paul, R. K., Dahiya, S. Iquebal, M. A., Godara, S. and Manjunath, B., (2024b). Price Forecasting of TOP (Tomato, Onion and Potato) Commodities using Hidden Markov based Deep Learning Approach. *Statistics and Applications*, **22**(2), 1-28.

Avinash, G., Ramasubramanian, V., Ray, M., Paul, R. K., Godara, S., Nayak, G. H., Kumar, R. R., Manjunath, B., Dahiya, S. and Iquebal, M. A. (2024a). Hidden Markov guided Deep Learning models for forecasting highly volatile agricultural commodity prices. *Applied Soft Computing*, *158*, 111557.

Baishya, M., Avinash, G., Sharma, K., Veershetty and Nayak, G. H. (2023). Navigating soybean price volatility: A deep learning perspective. *International Journal of Statistics and Applied Mathematics*; SP-8(5): 980-984

Che, Z., Cheng, Y., Zhai, S., Sun, Z. and Liu, Y. (2017, November). Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In *2017 IEEE International Conference on Data Mining (ICDM)*, 787-792.

Chen, Z. and Jiang, C. (2018). Building occupancy modeling using generative adversarial network. *Energy and Buildings*, **174**, 372-379.

Ding, X., Zhang, Y., Liu, T. and Duan, J. (2015, June). Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*.

Gao, N., Xue, H., Shao, W., Zhao, S., Qin, K.K., Prabowo, A. and Salim, F. D. (2022). Generative adversarial networks for spatio-temporal data: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, **13(2)**, 1-25.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S. and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, **27**.

Huang, W., Nakamori, Y. and Wang, S.Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers and operations research*, **32(10)**, 2513-2522.

Huang, X., Li, Q., Tai, Y., Chen, Z., Liu, J., Shi, J. and Liu, W. (2022). Time series forecasting for hourly photovoltaic power using conditional generative adversarial network and Bi-LSTM. *Energy*, **246**, 123403.

Jaiswal, R., Jha, G.K., Kumar, R.R. and Choudhary, K. (2021). Deep long short-term memory-based model for agricultural price forecasting. *Neural Computing and Applications*, 1-16.

Jinsung, Y. and Daniel, J. Schaar Mihaela van der. 2019. In *Time-series generative adversarial networks. In International Conference on Advances in Neural Information Processing Systems* (5508-5518).

Kara, Y., Boyacioglu, M. A. and Baykan, Ö.K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert systems with Applications*, **38(5)**, 5311-5319.

Li, S., Li, W., Cook, C., Zhu, C. and Gao, Y. (2018). Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (5457-5466).

Liu, Y., Yu, R., Zheng, S., Zhan, E. and Yue, Y. (2019). NAOMI: Non-autoregressive multiresolution sequence imputation. *Advances in neural information processing systems*, 32.

Luo, Y., Cai, X., Zhang, Y. and Xu, J. (2018). Multivariate time series imputation with generative adversarial networks. *Advances in neural information processing systems*, **31**.

Majhi, R., Panda, G., Sahoo, G., Panda, A. and Choubey, A. (2008, June). Prediction of S&P 500 and DJIA stock indices using particle swarm optimization technique. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* (1276-1282). IEEE.

Mogren, O. (2016). C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:*1611.09904. (https://arxiv.org/abs/1611.09904 was available for access on May 31, 2022)

Nayak, G. H., Alam, M. W., Singh, K. N., Avinash, G., Kumar, R. R., Ray, M., & Deb, C. K. (2024b). Exogenous variable driven deep learning models for improved price forecasting of TOP crops in India. *Scientific Reports*, *14*(1), 17203.

Nayak, G. H., Alam, W., Singh, K. N., Avinash, G., Ray, M., & Kumar, R. R. (2024a). Modelling monthly rainfall of India through transformer-based deep learning architecture. *Modeling Earth Systems and Environment*, 1-18.

Sheta, A. F., Ahmed, S.E.M. and Faris, H. (2015). A comparison between regression, artificial neural networks and support vector machines for predicting stock market index. *Soft Computing*, **7(8)**, 2.

Singh, K. N., Sharma, K., Avinash, G., Kumar, R. R., Ray, M., Ramasubramanian, V., Lama, A. and Lal, S. B. (2023). LSTM based Stacked Autoencoder Approach for Time Series Forecasting. *J. Indian Society of Agricultural Statistics*, *77*, 71-78.

Vinay, H.T., Pavitra, V., MS, J., Avinash, G., and GH, H. Nayak. (2024). A Comparative Analysis of Time Series Models for Onion Price Forecasting: Insights for Agricultural Economics. *Journal of Experimental Agriculture International*, *46*(5), 146-154.

Zhang, C., Kuppannagari, S.R., Kannan, R. and Prasanna, V.K. (2018). Generative adversarial network for synthetic time series data generation in smart grids. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (1-6). IEEE.

Zhang, K., Zhong, G., Dong, J., Wang, S. and Wang, Y. (2019). Stock market prediction based on generative adversarial network. *Procedia computer science*, **147**, 400-406.