International Journal of Innovative Science and Modern Engineering (IJISME)
ISSN: 2319-6386 (Online), Volume-12 Issue-2, February 2024

Robustness Evaluation of LSTM-based Deep Learning Models for Bitcoin Price Prediction in the Presence of Random Disturbances

Vijaya Kanaparthi

***Abstract:** As Deep Learning (DL) continues to be widely adopted, the growing field of study on the robustness of DL approaches in finance is gaining steam. This paper investigates the robustness of a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) intended for daily closing price predictions of Bitcoin (BTC). The research entails reproducing and adjusting an LSTM design from previous research, with an emphasis on evaluating the robustness of the network. The network is trained using data that has been disturbed by Gaussian noise to assess robustness, and the effect on predictions made outside of the sample is examined. To examine the impact of adding Gaussian noise layers and noisy dense layers on training accuracy and out-of-sample predictions, further robustness tests are conducted. The results show that the LSTM network has remarkable robustness to random disturbances in the data. Nevertheless, the Root Mean Square Error (RMSE) of the prediction increases with the addition of Gaussian noise and noisy dense layers. When random noise is present in the training data, the Autoregressive Integrated Moving Average (ARIMA) model is more vulnerable to it than the LSTM, according to the robustness of the two models. These findings highlight how robustness DL techniques are overall when compared to more conventional linear methods. However, because these models are black-box, the study highlights the significance of comprehensive testing. Although the robustness of the LSTM is impressive, it is important to understand that each network may behave differently depending on the circumstances.*

***Index Terms:** Autoregressive Integrated Moving Average, Bitcoin, Deep Learning, Gaussian Noise, Long Short-Term Memory, Recurrent Neural Network, Robustness, Root Mean Square Error*

I. INTRODUCTION

Artificial Intelligence (AI) systems are the cornerstone of many applications in today's dynamic world, with Machine Learning (ML) serving as the foundation for many of them. According to [1][27][28][29], ML is the capacity of systems to learn from particular training data, allowing the automation of the creation of analytical models to address related tasks. Deep Learning (DL) is a well-known subfield of ML that uses Artificial Neural Networks (ANNs) to solve complicated problems. As AI, ML, and DL are frequently used interchangeably, it is important to understand the differences between them for clarity.

Manuscript received on 03 January 2024 | Revised Manuscript received on 25 January 2023 | Manuscript Accepted on 15 February 2024 | Manuscript published on 28 February 2024.
**Correspondence Author(s)*
Vijaya Kanaparthi*, Senior Software Engineering, Microsoft, Northlake, Texas, USA. E-mail: datalivesite@gmail.com, ORCID ID: 0000-0003-2054-3101

AI is the development of systems intended to mimic or outperform human behaviour and judgement, particularly in complex scenarios. Deep Neural Networks (DNNs), a subtype of DL, on the other hand, differ from simple one-layer neural networks in that they include numerous hidden layers and sophisticated infrastructures inside these levels. This differentiation is crucial when we investigate the robustness of DL networks.

Specifically, we study the financial prediction of a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) architecture, using the closing prices of Bitcoin (BTC) as the target variable. Sequence models, an important element of DL, entail creating a sequence of outputs based on a sequence of inputs. This paradigm is used in several fields, including computer vision, natural language processing, and self-driving automobiles. Even with their widespread use, end users frequently lack an understanding of these DL systems' underlying workings. A devoted group of software engineers and data scientists is responsible for guaranteeing the precision, robustness, and prevention of undesirable results in real-world situations where outside influences and data fluctuations may affect the behaviour of the model. DL applications in finance require the same careful consideration for random perturbations in both the underlying data and model behaviour. The robustness of the model and the calibre of the data utilised in its construction are critical factors in determining the outcome of significant capital choices that depend on the outputs of DL models. Therefore, this paper aims to evaluate the robustness of a certain type of DL network (an RNN with LSTM architecture) in financial prediction using BTC closing prices as the case study. Based on the previous five-day closing prices and traded volume, multiple RNNs with different infrastructures will be built and given the duty of forecasting the closing prices of BTC for the upcoming five days. The original LSTM model, which was motivated by [2], is modified to meet the goals of this investigation. Data covering 1247 days, from January 1, 2018, to May 31, 2021, is used to train the models. Several models are created via testing, which eventually results in creating a model that can produce believable predictions. To satisfy the main objective of evaluating its robustness, this final model is subjected to a rigorous examination. The study examines the model's susceptibility to minute perturbations or adversarial assaults in the training data, which may be undetectable to humans, to assess the robustness of the suggested LSTM network. The research also looks at how model perturbations affect forecast accuracy.

Adversarial assaults are based on the idea of adversarial examples in neural networks and entail small-scale modifications to training data to negatively affect the predictions that a neural network makes. In this work, different noisy layers are added to the network and training data is mixed with Gaussian noise to produce adversarial samples. We provide a detailed analysis and assessment of the impacts of these perturbations on the training period and post-training prediction accuracy on data that is not in the sample. Furthermore, the precision and robustness of the RNN created in this investigation are contrasted with those of a conventional Autoregressive Integrated Moving Average (ARIMA) model tasked with forecasting closing BTC values. This thorough investigation clarifies the usefulness and dependability of DL models, providing insightful information for use in the unstable field of BTC price prediction.

This paper is as follows; related works are shown in the following section. Section III provides an overview of the data and preprocessing. The selected model and implementation are explained in Section IV. Section V delves into the robustness evaluation debate. Section VI presents the findings from the experiments. The discussion part in Section VII offers our perspectives on the results, and Section VIII wraps up the paper with some conclusions and ideas for future research.

II. RELATED WORKS

DL has emerged as a potent instrument in a number of fields, such as finance [3], science [4]–[7], and technology [8]–[10]. A key element of DL, neural networks have found effective applications in a wide range of fields, including pattern recognition [11], autonomous vehicle navigation [12], forecasting soil erosion [13], organic chemistry reaction prediction [14], and stochastic control in finance [15][25][26], [16]. The advent of innovative solutions, such as DeepMind's AlphaFold [], which predicts protein structures using neural networks, is evidence of the revolutionary potential of DL in addressing challenging issues. Compared to standard models, neural networks—especially deep ones—offer benefits in capturing non-linear interactions within training data, which improves in-sample fit. The accessibility of processing power, backpropagation techniques, and programming libraries like TensorFlow and PyTorch has enabled non-expert programmers to apply DL methods quickly. Neural networks are powerful tools, but they are not without restrictions. One key issue is their computational cost, as demonstrated by initiatives like as AlphaFold [17], whose model training needed massive amounts of data and supercomputer capacity. Furthermore, interpretability issues arise from DL models' "black-box" nature. Due to the large number of factors involved, it can be challenging to understand the underlying mechanics and behaviours of these models, which makes interpretation more complex than with typical linear approaches such as ARIMA models. One of the fundamental ideas in finance is the Efficient-Market Hypothesis (EMH) [2], which holds that asset prices accurately represent all available information. Research on the effectiveness of BTC markets has produced mixed findings; some claim that the markets are inefficient [18], while others claim that they have become more effective over time [19]. This discrepancy drives the experiments in this work, which build a model to forecast cryptocurrency values based on past prices and traded volumes and rigorously examine it. For financial applications, RNNs, which are optimised for sequential data such as time series, seem to be a good option. An RNN type called a LSTM network is particularly well-suited for modelling and predicting time series data in financial environments because it can handle problems such as disappearing or bursting gradients. DNNs are widely used and successful, but there isn't much research on how robust they are. Adversarial assaults are a type of testing that was first described by [20]. They test a network's vulnerabilities by introducing small disturbances. These attacks reveal neural network vulnerabilities that are hard for humans to find, which is why robustness metrics like Propagated output Quantified Robustness for RNNs (POPQORN) were developed. Even yet, there are still problems to be solved in measuring and enhancing neural networks' robustness in many scenarios. Thus, DL has advanced significantly across a range of domains, with neural networks exhibiting astounding powers. Applications include financial forecasting and scientific problem-solving. Nonetheless, difficulties like interpretability problems and processing costs highlight the necessity of further study to improve the robustness of DL models. This study sheds insight on the promise and limits of DL in the context of financial forecasting, as the financial world struggles with the efficiency of cryptocurrency markets. It does this by developing and analysing a model specifically designed to anticipate cryptocurrency values. The study of adversarial attacks and robustness in DL sequence models used in financial applications has received less attention than that of adversarial assaults and robustness in image classification and natural language processing. This is mostly because this field of study is new. Furthermore, practitioners who can create precise, cutting-edge DNNs for financial time series prediction that are highly robustness and impervious to adversarial attacks would likely be inclined to protect their proprietary models, realising the potentially profitable value of doing so. [21] are attempting to look into adversarial assaults in the context of high-frequency trading. Initially, attempts are made to estimate stock values 10 seconds into the future based on size-weighted average prices, employing order book data from the prior minute. They provide several potential techniques for crafting hostile assaults. An assault technique that "works on a large number of past training snippets with the aim that it translates to unseen testing bits" is the most practical course of action. The findings show that their model is continuously fooled by tiny perturbations that are minor in comparison to the order book. Furthermore, they note that adversarial patterns designed to fool one model frequently work well to fool others—a phenomenon that was previously discussed in a different context.

III. DATA

A. Data Description

The research employs cryptocurrency data, specifically daily closing prices of BTC in US Dollars (USD) and volume data in BTC obtained from the Binance exchange. The dataset is 1247 days long, starting on January 1, 2018, and ending on May 31, 2021. Because it is the most traded cryptocurrency and because Binance is the largest cryptocurrency exchange globally based on trading volume, BTC was chosen for this study. The information was obtained from *www.cryptodatadownload.com*, a website that offers large-scale cryptocurrency data from well-known exchanges at no cost for private or educational use. Fig. 1 below shows plots of the volume transacted in BTC throughout the dataset, as well as the closing values of BTC in USD. As previously noted, the study's concentration on cryptocurrencies is a result of possible market inefficiencies. As a result, the neural networks created for this study could be able to find long-term trends in the data and somewhat accurately forecast future pricing. The decision to focus on cryptocurrencies was made because of their quick rise in the finance industry, which has drawn many investors who were previously dubious about this new asset class. Furthermore, it is well known that cryptocurrency markets are extremely volatile. This has been demonstrated by past events, like as a 12% decline in price after an unexpected tweet from a powerful billionaire. It is expected that DL techniques, despite the data's intrinsic volatility, would be able to identify pricing trends and may even beat conventional linear models.

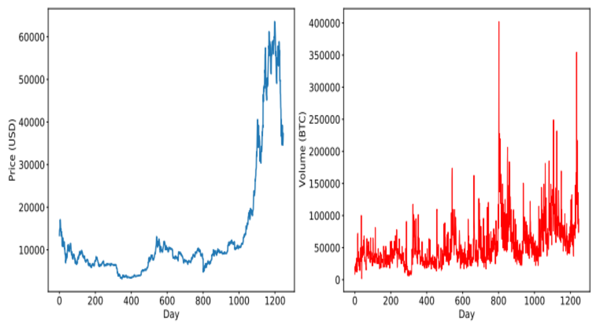

Fig. 1: Left: BTC Close Prices (USD). Right: Volume Traded (BTC)

B. Dataset Preprocessing and Windowing

The data will first be divided into three separate pieces. As of January 1, 2018, the first 90% of the data will be used as the training dataset. 10% will be further divided: two-thirds will be used as the validation dataset, and the remaining three-thirds will be used as the test dataset. The MinMax scaler will then be used to scale the data between zero and one. The data in all three blocks will be windowed into arrays of successive five-day inputs after scaling. Five-day closing prices and volumes will be utilised in this windowing method to forecast the asset's following five-day closing pricing. Next, these forecasts—referred to as the outputs—will be contrasted with the actual five-day closing prices. Fig. 2 shows a graphic depiction of the data windowing throughout the first 10 days of the training dataset. The closing price and volume traded at time *t* are represented by each orange block in this depiction, while the closing price at time *t* is represented by each blue block. The validation and test data sets will undergo the same windowing process. The network will be fitted and trained using the training data. As not directly used in the model fitting process, the validation set will make it possible to evaluate the model's generalisation skills while it is being trained. Lastly, nothing will be seen or affected by the test dataset, also known as unseen data, while the model is being trained. It will be applied to creating predictions based on the trained model's parameters as evaluating the model's results in predicting out-of-sample data.

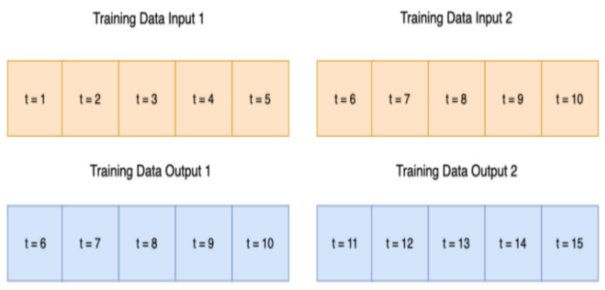

Fig. 2. Data Windowing Process for Training Data

IV. MODEL IMPLEMENTATION

A. Model Inspiration

The models utilised in this research are based on models developed by [2]. The authors' work involved the use of LSTM networks for regression to forecast changes in the NIFTY 50 stock price on the National Stock Exchange of India (NSE). The authors developed four different DL models: an RNN with one LSTM layer with 200 nodes, and three more dense layers with 100, 5, and 5 nodes, respectively. Our work replicates this basic architecture but adds certain adjustments, such as adding more LSTM and thick layers and using regularisation approaches. Concerning the models put forward in the previous study, these modifications seek to investigate possible improvements in prediction skills. In line with [2], we windowed the data, forecasting the next five-day pricing based on the five-day data that came before it. Nonetheless, experimenting with other input and prediction window widths is still a possibility. [2]'s study has a big impact on this work, and it fits in perfectly with the main goal of looking at how robustness a DL sequence model is in a financial setting. Furthermore, the amazing findings given by [2] emphasise the potency of DL with LSTM architecture compared to alternative ML approaches used to the same stock prediction problem. By using Root Mean Squared Error (RMSE) to assess prediction accuracy, the authors showed that LSTM-based models performed better than SVM regressions, random forests, and boosted regressions. Crucially, the authors left open the possibility of this study delving deeper into the robustness of models with a comparable purpose and architecture—in our instance, BTC price prediction—by not delving too far into the robustness of the models they created.

B. Model Building

During the model-building process, numerous frameworks were experimented with, involving the testing of different layer types, amounts of layers, and neurons in each layer over a range of networks. The previously mentioned model metrics will be used to compare the training, validation, and out-of-sample accuracy of various models, along with a few significant milestones from the model-building process. The first stage was to replicate [2] LSTM1 model. Our five-day windowed training data, which includes daily volume traded and close prices, makes up the input layer of this model. As such, the input layer's form is identified as (5, 2). The input data is then processed by an LSTM layer with 200 neurons after that. The LSTM1 layer then directs its output to two dense layers: one with 100 neurons and the other with five neurons. The output layer is a dense layer with five neurons that generates predictions for the next five days based on the windowed data that has been supplied. The ReLU activation function is used in every layer. The model is trained for 500 epochs with the ADAM optimizer at a learning rate of 0.00001, with the selected loss function being the Mean Squared Error (MSE). MSE and RMSE for this model are shown in Fig. 3. Fig. 3 shows a significant difference between the RMSE and loss values for the training and validation sets. This mismatch shows that while the model performs well in fitting the training data, it struggles to generalise successfully to the validation dataset. As a result, efforts were made to improve this baseline model's performance especially for our situation. While creating a state-of-the-art model for BTC prediction is not the main goal of this research, the goal is to construct a model with adequate prediction skills so that the robustness of a neural network that produces reasonable forecasts may be examined.

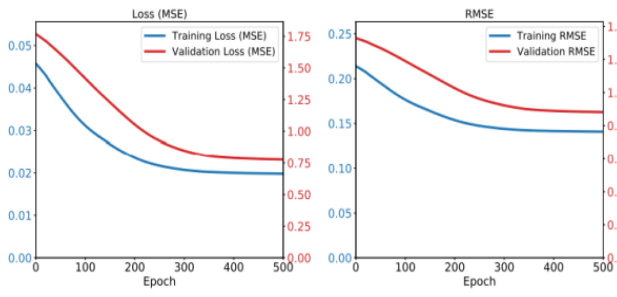

Fig. 3: LSTM1 MSE and RMSE for Training and Validation Datasets

Initially, the Leaky ReLU function was substituted for each activation function in the previous model in an effort to improve model performance. In this adaption, the α rate that was assigned to 0.2 by default in the Keras package was utilised. From now on, this altered network will be called LSTM2. It's significant to remember that the general architecture from the prior model was not altered. The training and validation sets' RMSE improved somewhat as a result of this modification. Fig. 4 shows plots of the MSE and RMSE for LSTM2. The Leaky ReLU activation function has been kept as the activation function for because to the improved performance shown at each layer.

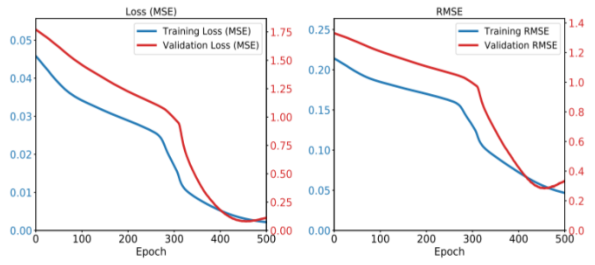

Fig. 4: LSTM2 MSE and RMSE for Training and Validation Datasets

While LSTM2 and the third created model of interest, or LSTM3, are identical, the third model included a second LSTM layer with 200 neurons following the first LSTM layer. By adding another layer, the network is made deeper, which make it possible to identify more subtle trends in the data. However, it is crucial to realise that this also prolong the training period and may lead to overfitting. Fig. 5 provides plots showing the model's training and validation losses as well as the RMSE. The observed improvement in the accuracy metrics with the addition of the extra LSTM layer led to the retention of this architectural modification in subsequent models.

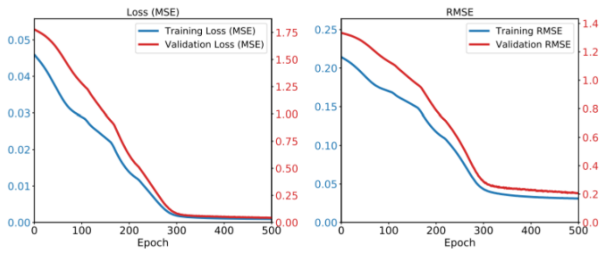

Fig. 5: LSTM3 MSE and RMSE for Training and Validation Datasets

The next significant step forward in the model-building process was adding a second dense layer to the network and investigating differences in the number of nodes in each layer. Then, LSTM4 is presented, with two successive LSTM layers that are identical to the previous one, but with four succeeding dense layers that have five, twenty, sixty, and one node each. When compared to LSTM3, LSTM4 performed better, as seen by the statistic that showed it reduced the RMSE for the validation set and just slightly increased it for the training set. The MSE and RMSE for this model are shown in Fig. 6, which shows significant variations from the results for LSTM1.

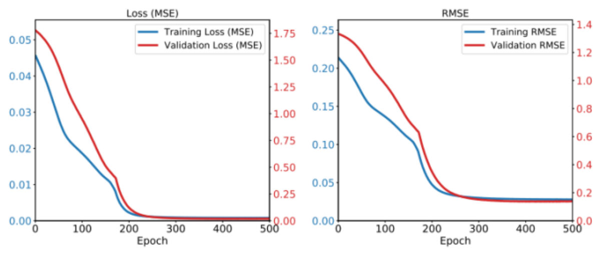

Fig. 6: LSTM4 MSE and RMSE for Training and Validation Datasets

Experimentation was also conducted by introducing dropout regularization to LSTM4 between each of the dense layers; however, this did not yield additional improvements in reducing the loss or RMSE, nor did it contribute to mitigating potential overfitting. In instances where the dropout rate was relatively high, it proved detrimental to the model's metrics. Plots illustrating the MSE and RMSE for LSTM5, featuring a dropout regularization rate of 0.3, are presented in Fig. 7. Oscillatory effects were induced in the performance metrics by the dropout regularization, leading to a failure of convergence in both training and validation loss. Consequently, the decision has been made not to proceed with the addition of dropout regularization to the network. Instead, LSTM4 will be retained as the final model for conducting the primary objective of this work, which involves investigating the robustness of a DL sequence model.

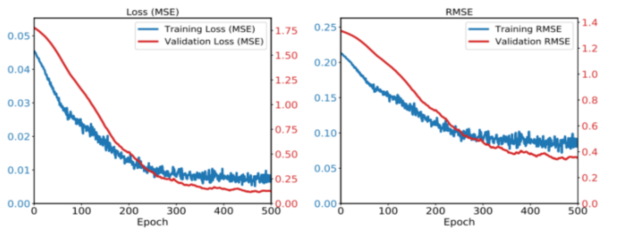

Fig. 7: LSTM5 MSE and RMSE for Training and Validation Datasets

C. Model Performance

LSTM4 is the model chosen to go closer to the main goal of this work. It performed well on the training and validation datasets, exhibiting little loss and RMSE. In summary, the suggested RNN for robustness analysis is set up with four dense layers—100, 60, and 20 neurons each—and two LSTM layers—each with 200 neurons—before reaching the final output layer, which has five neurons as previously mentioned. Each layer's activation function is the Leaky ReLU, with α set at 0.2. Upon comparing the plots of MSE and RMSE for this final model in Figure 6 to those for LSTM1 in Figure 3, it is evident that LSTM4 exhibits significantly lower loss and RMSE. While there exists the possibility of further refinement through continuous tuning of model hyperparameters, adjusting the number of layers within the network, and experimenting with different learning rates and optimizers to enhance accuracy, this is not the central focus of this work. While there are 40 days' worth of prices in the unseen out-of-sample dataset, it would be unreasonable to make forecasts for all 40 days. This is because the validation dataset's 80 days, which coincide with a time of notable bull run in the BTC market generally, were not used to train our model. When such a critical time is removed from the training data, our neural network can miss certain underlying trends. To assess out-of-sample performance, we thus limited our unseen dataset to the first 15 prices and volumes. In a real-world situation, when fresh data is collected, say at the end of the day or after five days, and the overall architecture consistently produces forecasts that beat the market, the model would be continuously modified and retrained. For the sake of this investigation, we do not make this ongoing modification to see how robustness tests affect training, validation, out-of-sample loss, and RMSE. The RMSE that results from predicting prices with the unseen dataset is 0.1179. To acquire the real and anticipated values, the scaling procedure must be reversed because these numbers are scaled. Fig. 8 provides plots showing the genuine prices and projections of the model. Considering that the model's sole inputs were closing prices and volume traded, its forecast performance is impressive. Moreover, the unseen dataset has the most current prices from the complete dataset, but the validation dataset, which had more recent prices, was not used to train the model.

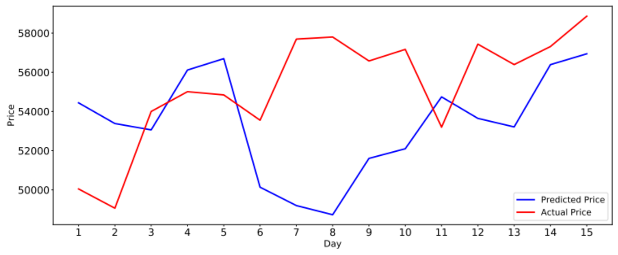

Fig. 8: Predicted and Actual BTC Prices (USD) for LSTM4

V. ROBUSTNESS EVALUATION

The first technique used to evaluate the model's robustness is to generate Gaussian noise and then add it to the training set of data. The zero mean and variable standard deviation define the Gaussian noise. At the beginning, the standard deviation is initially kept very small to "disguise" the additional noise and preserve the unnoticeable nature of the robustness tests. Furthermore, there is flexibility in how many noise perturbations are produced. By experimenting with a high number of perturbations from the same distribution as compared to a limited number of perturbations in the training data, this variability aims to investigate whether there is any impact on the model's training performance. Every disturbance is dispersed at random among the blocks and added at random to one of the five block closing prices. It's crucial to remember that the disturbances can only have an impact on one closing price each five-day block. This perturbation needs to be performed to both the input and output vectors in order to preserve data consistency. For an illustration of noise injection in a single five-day block, see Fig. 9, where x_i stands for the closing price at time i and represents the Gaussian noise. The data that is disturbed is indicated in red. This technique was picked mostly for its simplicity, as introducing noise to the training data is a trivial operation, especially considering that the training data is scaled between 0 and 1 owing to the nature of the MinMax scaler. Secondly, this sort of noise injection connects nicely with our prior discussion of adversarial assaults, when minor perturbations are injected to the data with the purpose of limiting model performance. Nonetheless, it is crucial to maintain the realism of this noise injection since the perturbations are meant to be negligible enough for people or other detection methods to miss them. The literature recognises this type of data augmentation as a common regularisation technique and a way to evaluate the robustness of the model.

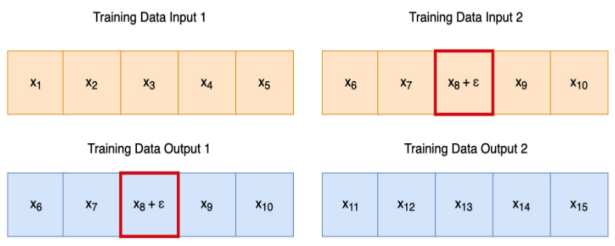

Fig. 9: Noise Injection Example

[22] misclassified a time series because of undetectable noise perturbing it in a time series classification task. Using the same DL infrastructure as the final model, we will fit and train a model using the adjusted noisy training data, and we will explore the effects of different noise intensities and frequencies on the model. The model's performance will then be assessed on our unseen out-of-sample dataset and contrasted with the original data setting used to train LSTM4. There are several techniques to introduce noise into the neural network itself. The inclusion of Gaussian noise layers between neural network layers and the substitution of noisy dense layers for dense layers in the DL model will be the two main techniques discussed. This is a little less practical way to introduce an adversarial perturbation since it would require a quantitative researcher to build the code to add these layers to their network, but testing the robustness of a DL model using this experiment is still very important. This experiment adds to the evaluation of the potential "black box" issue with certain models. The incorporation of synthetic noise into the model's internal structure might reveal the model's susceptibility to additional unanticipated inputs or mistakes made during the model-building phase. To put the performance and robustness of the RNN created in this work into context, a comparison will be made against that of traditional linear methods. An ARIMA model designed for predicting closing prices, like our neural network, will be created. The performance of this ARIMA model in predicting prices for the unseen data will be investigated. Additionally, the ARIMA model's parameters and the statistical significance of these parameters will be explored when noise is added to the data used to fit the model, like the noise injection technique for LSTM4. Despite ARIMA models being somewhat obsolete for modelling asset prices and having been replaced by more complicated ML methods, this comparison is crucial for contextualizing the robustness or potential vulnerability of the LSTM networks created in this work.

VI. RESULTS

A. Data Noise Injection Results

Initially, Gaussian noise with a standard deviation of 0.01 to 25 days was added to the training data in order to verify robustness utilising the noise injection approach. Considering that the training set contained prices for more than 1000 days, this was a rather modest number of days, but it was still a useful baseline. The original data is shown in red in Fig. 10, and the noisy data is shown in blue. It is seen that the two lines are almost identical. Because of this, humans are unlikely to notice this augmentation of the training data. Fig. 11 shows plots of the MSE and RMSE of the LSTM4 trained on this noisy data set. There was a very minor reduction in the training data's RMSE. At 0.1179, the RMSE of the forecasts for the hidden data did not change. The noise injection experiment was then repeated, adding 100 Gaussian noise samples to the training set this time. Analyzing a plot of the data in Fig. 12 revealed that, once again, the disturbance was mostly undetected. As seen in Fig. 13, the supplemented data again had no effect on the training and validation loss and RMSE. At 0.1182, the RMSE for the out-of-sample data stayed mostly unaltered.

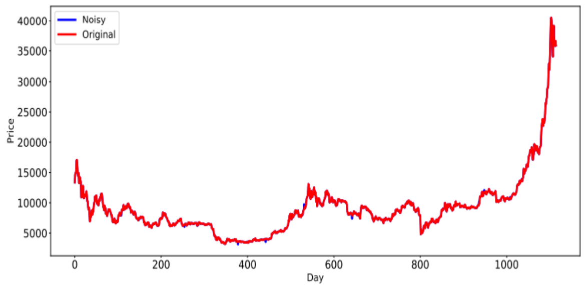

Fig. 10: Noisy and Original Training Data with 25 Noisy Data Points

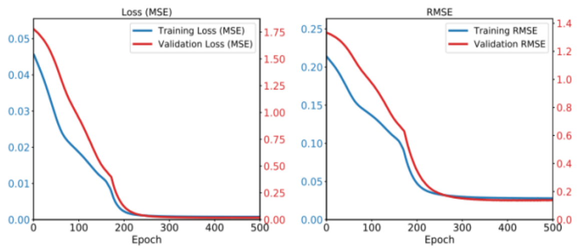

Fig. 11: MSE and RMSE for Training Data with 25 Noisy Days and Validation Data

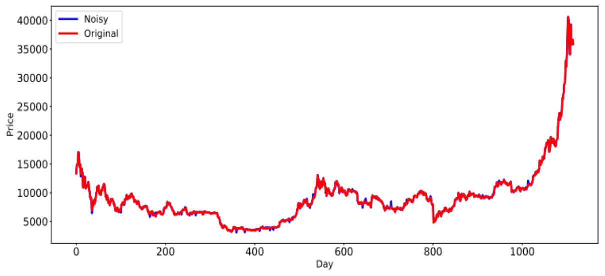

Fig. 12: Noisy and Original Training Data with 100 Noisy Data Points

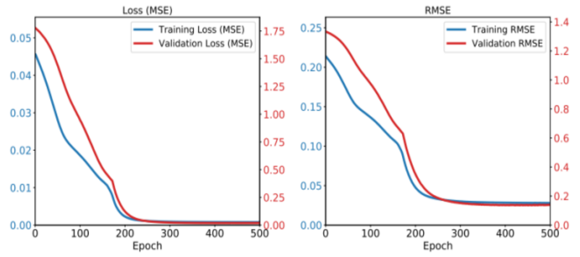

Fig. 13: MSE and RMSE for Training Data with 100 Noisy Days and Validation Data

After the two previous noise injection experiments showed no significant decline in the model's prediction power, an extreme example was tested by introducing 250 noise samples at random throughout the training data and raising the noise standard deviation to 0.1. Fig. 14 illustrates how this augmentation has an impact. Upon examining this figure, a researcher would undoubtedly detect some inaccuracy made during the data processing phase. Still, the procedure went forward, and this enhanced data was used to train the model. Surprisingly, despite the significant noise supplied to the training data, the MSE and RMSE of the noisy data points seemed to have little to no influence, as seen in Fig. 15. In addition, the out-of-sample RMSE was 0.1169, which is comparable to earlier tests using original and noisy training data. In contrast to the first two trials, there was a little rise in the training RMSE.

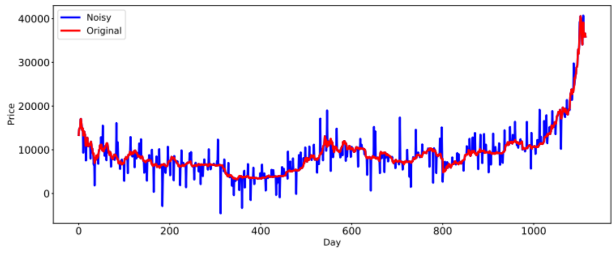

Fig. 14: Noisy and Original Training Data with 250 Noisy Data Points

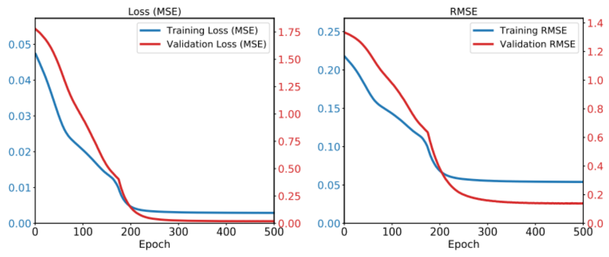

Fig. 15: MSE and RMSE for Training Data with 100 Noisy Days and Validation Data

B. Model Noise Injection Results

a. Gaussian Noise Layers

The outcomes of applying Gaussian noise layers to the network are shown in order to evaluate the various approaches for introducing noise into the network itself. The suggested final model was first modified by adding three Gaussian noise layers with a standard deviation of 0.02. Following the second LSTM layer, one was added, and then one following each of the subsequent two dense layers. Plots of the MSE and RMSE during the model training phase can be found in Fig. 16. With respect to the unseen dataset, the RMSE of the predictions was virtually stable at 0.1166.

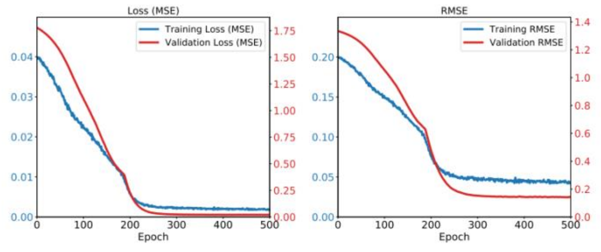

Fig. 16: MSE and RMSE for LSTM4 with Three Gaussian Noise Layers

Additional Gaussian noise layer was added, with the first three in the same places as previously in the model, and another Gaussian noise layer before the last dense layer, without having a significant impact on the MSE and RMSE from the prior experiment. Additionally, the standard deviation was raised to 0.1 in every layer to look at the possible impact of raising the perturbation. It is evident from Fig. 17 that the accuracy of the network for the training and validation data was affected by both the addition of an additional noisy layer and raising the standard deviation of the Gaussian noise. This jitter effect indicates that the loss cannot converge, proving detrimental to the predictions for the unseen dataset, which had an RMSE of 0.4019 compared to an RMSE of 0.1179 for LSTM4. Fig. 18 displays a visualisation of the predictions generated by this perturbed network. It is evident that their forecasts are noticeably poorer than LSTM4's.

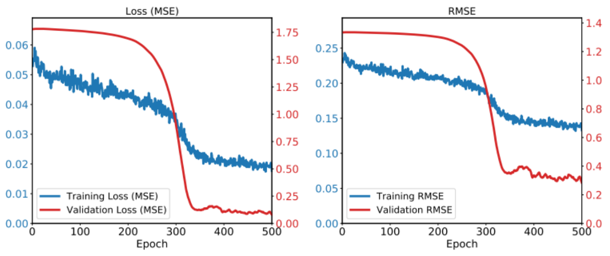

Fig. 17: MSE and RMSE for LSTM4 with Four Gaussian Noise Layers

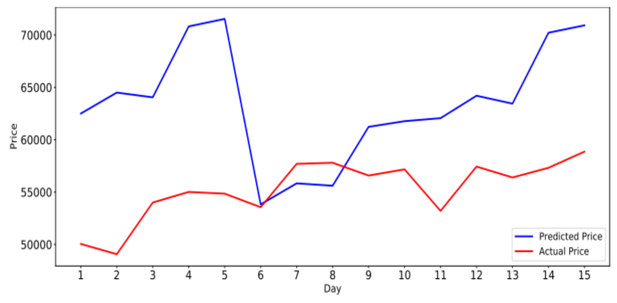

Fig. 18: Predicted and Actual Prices for LSTM4 with Four Gaussian Noise Layers

b. Noise Dense Layers

The following set of data shows what happens when noisy dense layers are added to some of the RNN's dense layers. Initially, a noisy dense layer with a standard deviation of 0.1 was added in place of the dense layer with 100 neurons. The noisy layer was substituted for this layer, and the MSE and RMSE graphs in Fig. 19 show that this change had an instant impact on the metrics for validation data, causing some oscillatory effects. With this model, the RMSE of the predictions for the unseen data was 0.1567, which was somewhat less accurate than our final model's accuracy. Following the observation that one noisy dense layer had a negative influence on the model metrics during training, an analysis was carried out to determine the unfavourable effect of replacing all dense layers—with the exception of the final output layer.

The 100, 60, and 20 neuron dense layers were swapped out with noisy dense layers with standard deviations of 0.1 and the corresponding number of neurons in each layer. As can be seen in Fig. 20, the results from the preceding experiment were amplified as predicted. This also had an impact on the 0.1829 RMSE of the predictions made using the unseen dataset. It is noticeable that this performance is poorer than the RMSE for LSTM4 but better than the findings from the last trial with Gaussian noise layers.

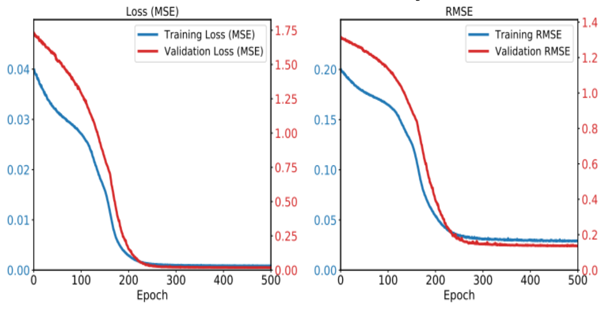

Fig. 19: MSE and RMSE for Training Data with One Noisy Dense Layer

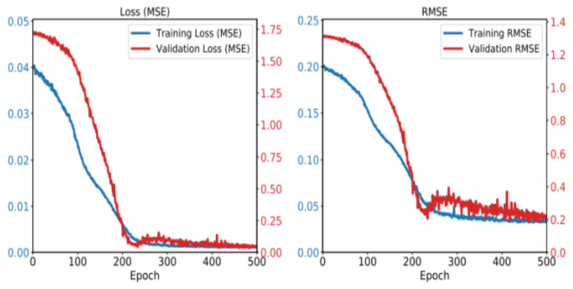

Fig. 20: MSE and RMSE for Training Data with Three Noisy Dense Layers

C. Comparison with ARIMA

The out-of-sample predictions for the same 15 days, generated by the ARIMA (2, 1, 2) model are presented in Fig. 21. It is evident that predictions made by this model are inferior when compared to our LSTM4 network in terms of accuracy in out-of-sample data. The same projection was executed, but the ARIMA (2, 1, 2) model was updated after making a 5-day prediction. This involved fitting the ARIMA (2, 1, 2) model again, incorporating the actual values from the unseen dataset into the training data. The predictions for this updated model are presented in Fig. 22. Even though this significantly altered fitting method is an improvement over the prior model, it still has a very low five-day prediction power. It basically establishes that the price of BTC will stay practically unchanged until the model is updated throughout the refitting process with the real price at the beginning of the following five-day timeframe. Additionally, Gaussian noise—which is comparable to the noise provided for training the RNN—was added to the training data in order to fit the ARIMA model. The additional noise was applied to 15% of the closing prices over a period of 1200 days, with a standard deviation of 0.1. A plot illustrating the loud and original pricing is given in Fig. 23. Once more, there is not much of a difference between this noise and the original.

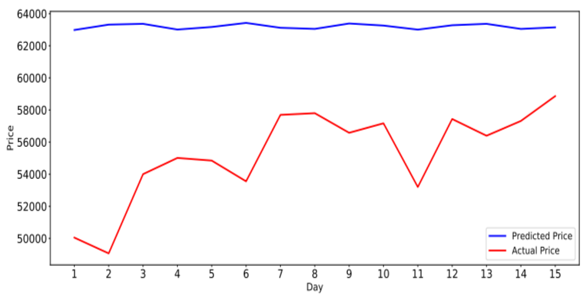

Fig. 21: ARIMA (2, 1, 2) 15-Day Projections

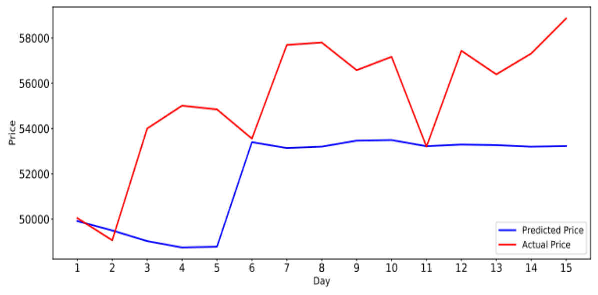

Fig. 22: ARIMA (2, 1, 2) 15-Day Projections with Model Updating

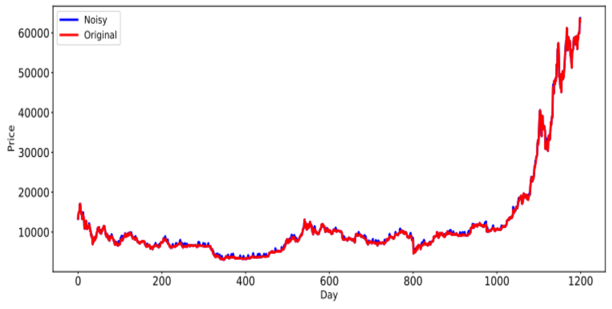

Fig. 23: ARIMA (2, 1, 2) Noisy Training Data

VII. DISCUSSION

It can be seen from the RMSE and the plot of our final model's predictions that the model offers reasonable forecasts for each of the three five-day window periods in the out-of-sample dataset. In a little critical note, there are times in the 15-day forecasts where the expected prices are around 7,000 less than the actual price. It would be quite difficult to predict daily BTC values using only past closing prices and trade volumes with a level of accuracy that would beat the market. Examining the MSE and RMSE for the training data, it is accepted that there are faint indications of overfitting; the RMSE is around 0.04. Adding dropout regularisationin an attempt to fix this didn't help the RMSE converge. This may be resolved in future studies by adjusting the learning rate, adding more input variables, and investigating alternative regularisation techniques. It was shown that using noisy data to train the model did not significantly alter the RMSE for the out-of-sample predictions. It is not unexpected, nevertheless, that the training loss and RMSE marginally increased when trained on the noisiest data, with $\sigma = 0.1$.

Practitioners who may experience noise in their data from randomness or imprecise measuring techniques may find these results encouraging. They could still be certain that the noise won't materially impair their ability to foresee. This accords with the findings of [23], where sequence-to-sequence models are regarded to be resilient to adversarial assaults in the data. The RMSE for the out-of-sample predictions in the first case, where three Gaussian noise layers with a standard deviation of 0.02 were added to the network, was marginally higher than the RMSE for the original model. While there is a tiny jitter pattern in the training RMSE plot shown in Figure 16, there is not a significant increase. The random generation of the Gaussian noise for every training session is probably the cause of this jitter effect. When the standard deviation was raised to 0.1 and an additional Gaussian noise layer was added to the network, this impact was amplified. The model's inability to converge was evident from the increasing training and validation MSE and RMSE, which may be attributed to the Gaussian noise layers. This impact extended to the predictions made outside of the sample, where the RMSE was 0.4019, far higher than the LSTM4 estimate. The training and validation loss were first somewhat impacted by the addition of one noisy dense layer, with the out-of-sample RMSE rising to 0.1576 from the LSTM4 baseline of 0.1179. The impacts on the MSE and RMSE increased with the replacement of two more thick layers. Surprisingly, the noisy thick layers kept the RMSE from converging, which is why the validation RMSE suffered the most. Even though the RMSE increased to 0.1829, the effect on out-of-sample accuracy was not as significant as it was with the Gaussian noise layers. The number of noisy layers and the noise standard deviation both had a significant role in the model's disruption. These findings could suggest that an RNN is more vulnerable to disturbances transferred across network layers than they are to disturbances contained inside a layer. Furthermore, it was discovered that the ARIMA model outperformed the LSTM models in out-of-sample prediction and was very sensitive to disturbances in the data. This is consistent with [24] claim that RNNs are significantly more effective than conventional linear techniques.

VIII. CONCLUSION AND FUTURE WORKS

It was possible to develop an RNN with LSTM architecture that could reasonably forecast BTC values. Increasing the depth and adding LSTM layers to the network increased prediction accuracy for validation dataset and the out-of-sample dataset. The DL sequence model was resilient to random perturbations in the data, even when the magnitude and quantity of perturbations in the training data were unreasonably enormous, according to tests conducted to assess the model's resilience. However, the addition of noisy dense layers and Gaussian noise layers, especially several disruptive layers, weakened the resilience of the model. The most notable decline in model accuracy was found in Gaussian noise layers, particularly when the noise's standard deviation was raised to 0.1. These model disruptions highlight the need for prudence while building deeper networks, as demonstrated by models such as Alpha Fold, which consists of many linked networks. For out-of-sample prediction, the RNN performed better than a linear ARIMA model and demonstrated resistance to data disturbances, which resulted in significant changes to the AR and MA parameters of the ARIMA model. So, the LSTM showed a respectable level of resilience; only intrusive, sometimes unreasonably large perturbations were necessary to get it to perform less well. Even though it was successful, this work has flaws and deserves more debate. It may be argued that a model trained just on volume traded and closing prices would not provide hidden excess alpha over time. However, the goal was to concentrate on the main goals of this work rather than to create a cutting-edge model that might compete with quantitative hedge funds. Despite having just six layers and over 500,000 parameters, LSTM4's model complexity was maintained manageable in order to achieve the key objectives. The EMH may impose constraints on these models, and subsequent iterations may experiment with various time scales and prediction windows in order to evaluate their robustness at minute, second, or even lower time scales. Nonetheless, before evaluating the robustness of these models, it would be crucial to take into account how market dynamics could change over incredibly short time periods. Given their projected future use, it will remain imperative to investigate the robustness of financial DL models.

DECALARION STATEMENT

Funding	No, I did not receive.
Conflicts of Interest	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material/ Data Access Statement	Not relevant.
Authors Contributions	I am only the sole author of the article

REFERENCES

1. D. Muchlinski, "Machine learning and deep learning," in Elgar Encyclopedia of Technology and Politics, vol. 31, no. 3, Springer Science and Business Media Deutschland GmbH, 2022, pp. 114–118. doi: 10.4337/9781800374263.machine.learning.deep. https://doi.org/10.4337/9781800374263.machine.learning.deep
2. S. Mehtab, J. Sen, and A. Dutta, "Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models," in Communications in Computer and Information Science, 2021, vol. 1366, pp. 88–106. doi: 10.1007/978-981-16-0419-5 8. https://doi.org/10.1007/978-981-16-0419-5
3. H. Habib, G. S. Kashyap, N. Tabassum, and T. Nafis, "Stock Price Prediction Using Artificial Intelligence Based on LSTM– Deep Learning Model," in Artificial Intelligence & Blockchain in Cyber Physical Systems: Technologies & Applications, CRC Press, 2023, pp. 93–99. doi: 10.1201/9781003190301-6. https://doi.org/10.1201/9781003190301-6
4. S. Wazir, G. S. Kashyap, and P. Saxena, "MLOps: A Review," Aug. 2023, Accessed: Sep. 16, 2023. [Online]. Available: https://arxiv.org/abs/2308.10908v1
5. M. Kanojia, P. Kamani, G. S. Kashyap, S. Naz, S. Wazir, and A. Chauhan, "Alternative Agriculture Land-Use Transformation Pathways by Partial-Equilibrium Agricultural Sector Model: A Mathematical Approach," Aug. 2023, Accessed: Sep. 16, 2023. [Online]. Available: https://arxiv.org/abs/2308.11632v1

6. N. Marwah, V. K. Singh, G. S. Kashyap, and S. Wazir, "An analysis of the robustness of UAV agriculture field coverage using multi-agent reinforcement learning," International Journal of Information Technology (Singapore), vol. 15, no. 4, pp. 2317–2327, May 2023, doi: 10.1007/s41870-023-01264-0.
7. G. S. Kashyap, K. Malik, S. Wazir, and R. Khan, "Using Machine Learning to Quantify the Multimedia Risk Due to Fuzzing," Multimedia Tools and Applications, vol. 81, no. 25, pp. 36685–36698, Oct. 2022, doi: 10.1007/s11042-021-11558-9.
8. G. S. Kashyap, A. E. I. Brownlee, O. C. Phukan, K. Malik, and S. Wazir, "Roulette-Wheel Selection-Based PSO Algorithm for Solving the Vehicle Routing Problem with Time Windows," Jun. 2023, Accessed: Jul. 04, 2023. [Online]. Available: https://arxiv.org/abs/2306.02308v1
9. G. S. Kashyap, D. Mahajan, O. C. Phukan, A. Kumar, A. E. I. Brownlee, and J. Gao, "From Simulations to Reality: Enhancing Multi-Robot Exploration for Urban Search and Rescue," Nov. 2023, Accessed: Dec. 03, 2023. [Online]. Available: https://arxiv.org/abs/2311.16958v1
10. S. Wazir, G. S. Kashyap, K. Malik, and A. E. I. Brownlee, "Predicting the Infection Level of COVID-19 Virus Using Normal Distribution-Based Approximation Model and PSO," Springer, Cham, 2023, pp. 75–91. doi: 10.1007/978-3-031-33183-1_5. https://doi.org/10.1007/978-3-031-33183-1_5
11. S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in Proceedings of 2014 Science and Information Conference, SAI 2014, Oct. 2014, pp. 372–378. doi: 10.1109/SAI.2014.6918213. https://doi.org/10.1109/SAI.2014.6918213
12. J. Radcliffe, J. Cox, and D. M. Bulanon, "Machine vision for orchard navigation," Computers in Industry, vol. 98, pp. 165–171, Jun. 2018, doi: 10.1016/j.compind.2018.03.008. https://doi.org/10.1016/j.compind.2018.03.008
13. M. Rekha Sundari, G. Siva Rama Krishna, V. Sai Naveen, and G. Bharathi, "Crop Recommendation System Using K-Nearest Neighbors Algorithm," in Lecture Notes in Networks and Systems, 2021, vol. 177 LNNS, pp. 581–589. doi: 10.1007/978-981-33-4501-0_54. https://doi.org/10.1007/978-981-33-4501-0_54
14. J. L. Araújo, C. Morais, and J. C. Paiva, "Poetry and alkali metals: Building bridges to the study of atomic radius and ionization energy," Chemistry Education Research and Practice, vol. 16, no. 4, pp. 893–900, Oct. 2015, doi: 10.1039/c5rp00115c.
15. J. Wang, T. Sun, B. Liu, Y. Cao, and D. Wang, "Financial Markets Prediction with Deep Learning," in Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, Jan. 2019, pp. 97–104. doi: 10.1109/ICMLA.2018.00022. https://doi.org/10.1109/ICMLA.2018.00022
16. T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," PLoS ONE, vol. 14, no. 2, p. e0212320, Feb. 2019, doi: 10.1371/journal.pone.0212320. https://doi.org/10.1371/journal.pone.0212320
17. L. Espeholt et al., "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures," in 35th International Conference on Machine Learning, ICML 2018, Jul. 2018, vol. 4, pp. 2263–2284. Accessed: Dec. 03, 2023. [Online]. Available: https://proceedings.mlr.press/v80/espeholt18a.html
18. S. Agarwal and N. B. Muppalaneni, "Portfolio optimization in stocks using mean–variance optimization and the efficient frontier," International Journal of Information Technology (Singapore), vol. 14, no. 6, pp. 2917–2926, Oct. 2022, doi: 10.1007/s41870-022-01052-2. https://doi.org/10.1007/s41870-022-01052-2
19. O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," Applied Soft Computing Journal, vol. 90, 2020, doi: 10.1016/j.asoc.2020.106181. https://doi.org/10.1016/j.asoc.2020.106181
20. C. Szegedy et al., "Intriguing properties of neural networks," in 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings, Dec. 2014. Accessed: Jan. 18, 2024. [Online]. Available: https://arxiv.org/abs/1312.6199v4
21. M. Goldblum, A. Schwarzschild, A. Patel, and T. Goldstein, "Adversarial attacks on machine learning systems for high-frequency trading," in ICAIF 2021 - 2nd ACM International Conference on AI in Finance, Nov. 2021. doi: 10.1145/3490354.3494367. https://doi.org/10.1145/3490354.3494367
22. H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller, "Adversarial Attacks on Deep Neural Networks for Time Series Classification," in Proceedings of the International Joint Conference on Neural Networks, Jul. 2019, vol. 2019-July. doi: 10.1109/IJCNN.2019.8851936. https://doi.org/10.1109/IJCNN.2019.8851936
23. M. Cheng, J. Yi, P. Y. Chen, H. Zhang, and C. J. Hsieh, "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," in AAAI 2020 - 34th AAAI Conference on Artificial Intelligence, Apr. 2020, vol. 34, no. 04, pp. 3601–3608. doi: 10.1609/aaai.v34i04.5767. https://doi.org/10.1609/aaai.v34i04.5767
24. G. Petneházi, "Recurrent Neural Networks for Time Series Forecasting," Jan. 2019, Accessed: Jan. 18, 2024. [Online]. Available: https://arxiv.org/abs/1901.00069v1
25. Mukherjee, P., Palan, P., & Bonde, M. V. (2021). Using Machine Learning and Artificial Intelligence Principles to Implement a Wealth Management System. In International Journal of Soft Computing and Engineering (Vol. 10, Issue 5, pp. 26–31). https://doi.org/10.35940/ijsce.f3500.0510521
26. Stock Price Prediction. (2019). In International Journal of Innovative Technology and Exploring Engineering (Vol. 9, Issue 2S, pp. 425–429). https://doi.org/10.35940/ijitee.b1042.1292s19
27. Md. Rakib, M. K., Himu, H. D., Fahim, Md. O. F., Zaman, Ms. Z., & Palak, MD. J. U. R. (2023). Automatic Recognition of Medicinal Plants: Based on Multispectral and Texture Features using Hidden Deep Learning Model. In Indian Journal of Computer Graphics and Multimedia (Vol. 3, Issue 1, pp. 1–7). https://doi.org/10.54105/ijcgm.d4089.023123
28. Rajeev, H., & Chakkaravarty, Dr. M. (2023). Prediction of Cybercrime using the Avinashak Algorithm. In Indian Journal of Artificial Intelligence and Neural Networking (Vol. 4, Issue 1, pp. 5–10). https://doi.org/10.54105/ijainn.a1078.124123
29. Jindam, S., Mannem, J. K., Nenavath, M., & Munigala, V. (2023). Heritage Identification of Monuments using Deep Learning Techniques. In Indian Journal of Image Processing and Recognition (Vol. 3, Issue 4, pp. 1–7). https://doi.org/10.54105/ijipr.d1022.063423

AUTHORS PROFILE

Vijaya Kanaparthi, Education Qualifications: Bachelor of commerce, ICWAI. Currently Working for Microsoft, Accomplished finance and accounting leadership professional with a passion for leveraging data analytics and AI technology solutions to solve business problems and elevate organizational capacity for growth, operational excellence, and profitability. Experienced in leading and collaborating with international and cross-functional teams to implement innovative solutions, perform complex functional analysis, build models and propose strategic changes to bolster growth. Served as trusted advisor to large corporations across oil & gas, information technology and manufacturing to navigate turnarounds, streamline processes via automation, and minimize transaction errors and revenue loss. Did extensive research in finance, Machine learning & Artificial Intelligence areas. Published research papers on Finance & Artificial Intelligence areas. Been a Judge for the 2023 Globee International awards. Currently a member of the world economic forum.