International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

# Comparative Analysis of TCN & DeepTCN Models for Indonesian Stock Price Prediction

**Felix[1], Evandiaz Fedora[1] and Alexander Agung Santoso Gunawan[2]**

*[1] Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia*
*[2]Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia*

*E-mail address: felix017@binus.ac.id, evandiaz.fedora@binus.ac.id, aagung@binus.edu*

*Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##*

**Abstract:** Accurately forecasting stock prices movements can lead to financial gains, making it a highly sought-after area of study. In recent studies, Temporal Convolutional Network (TCN) has risen in popularity due to its use of dilated convolutions, which are adept at capturing temporal dependencies within time series data. DeepTCN, a variation of TCN designed specifically for probabilistic forecasting, is said to outperform other models in time series forecasting. As far as we know, no extensive research has been conducted to evaluate the performance of DeepTCN compared to TCN. This study conducted a comparative analysis to assess the performance of both TCN and DeepTCN in Indonesian stock price prediction. Both models will be evaluated using Mean Squared Error (MSE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) scores. The result from this comparative analysis shows that DeepTCN is superior to TCN in predicting stock prices. DeepTCN consistently outperforms TCN, with lower values of MSE, RMSE, and MAPE. This improved performance lies in the parametric approach used in DeepTCN, which allows it to better capture and adapt to fluctuations in stock trends. The findings from this comparative analysis emphasize the need to assess forecast objectives and dataset requirements when choosing between TCN and DeepTCN.

**Keywords:** Deep Learning, TCN, DeepTCN, Stock Prediction, Time Series Forecasting, Indonesian Stock Market

## 1. Introduction

The stock exchange, commonly known as the stock market, functions as a dynamic marketplace where the trading of shares takes place. It has been proven that stock market capitalization holds a crucial role in propelling a nation's economy forward [1]. Shares of publicly traded companies, having undergone the listing process on the stock market, represent tradable ownership units in these companies. Owning company shares serves as evidence of ownership, granting shareholders access to associated benefits and privileges. The movement of stock prices is primarily influenced by the interaction between demand and supply forces, further shaped by the actions of traders engaged in buying and selling shares. Share transactions aim for financial gains, like conventional transactions involving goods and services. Accurately predicting a stock's trajectory can lead to substantial financial gains, making it a highly sought-after area of study. Forecasting the stock market presents a challenge due to the market's susceptibility to national policies, global and regional economic factors, as well as psychological, human, and other variables [2].

In the era of artificial intelligence, machine learning has become crucial for time series forecasting. Machine learning algorithms are known for their improved accuracy in predicting stock prices [3]. Since the trends of the stock market are constantly changing, the amount of data generated in the stock market is huge and has significant nonlinearity. To effectively handle such dynamic data, a model that can analyze hidden patterns and understand the underlying dynamics is needed [4].

Recently, deep learning models have been increasingly used as their performance surpasses statistical and traditional models. The nonlinear dynamics within deep learning enables a comprehensive understanding of the complex characteristics in the stock market [5]. Among these models, a specialized Convolutional Neural Network (CNN) architecture, namely Temporal Convolutional Network (TCN), has gained popularity due to its dilated convolutions, which

*E-mail:author's email*

can effectively capture long-range dependencies in time series, making them well-suited for modeling temporal relationships in sequential data. In addition to TCN, a variant known as Deep Temporal Convolutional Network (DeepTCN) has been developed by [6 developer] to forecast numerous correlated time series data through an encoder-decoder architecture. DeepTCN builds upon TCN foundation by introducing additional elements like residual blocks to tackle more challenging task of probabilistic forecasting, which stated by [6] that their model demonstrates a better result when compared to state-of-the-art models like Seasonal ARIMA (SARIMA) and light gradient-boosting machine (lightGBM) in both forecasting and probabilistic forecasting tasks. However, to the best of our knowledge, the capabilities of the DeepTCN model have not been thoroughly evaluated in comparison to its baseline TCN model.

To address the issue, this study conducted a comparative analysis between TCN and DeepTCN to evaluate their performance in the sector of forecasting Indonesian stock prices. Utilizing DeepTCN's strength in probabilistic forecasting, this study also presents a novel method for Indonesian stock price prediction by integrating probabilistic forecasting framework with parametric approach to forecast time series data. This method leverages the capability of DeepTCN to deliver not only point predictions but also a measure of uncertainty, allowing for more informed decision-making in stock market investments. Various evaluation metrics, including Mean Squared Error (MSE), Root Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE) were employed to evaluate the performance of these models.

The key contributions of this study are as follows:

- Providing a comparative analysis of TCN and DeepTCN for Indonesian stock price prediction. This would involve training and testing the models on Indonesian stock data and assessing their effectiveness in forecasting future prices.
- Identifying strengths and weaknesses of TCN and DeepTCN models to analyze which model performs better for Indonesian stock price prediction. This study delves into factors that influence the performance of TCN and DeepTCN, potentially attributing it to factors like the data's characteristics or the models' suitability for the financial sector.
- Given DeepTCN's specialization in probabilistic forecasting, this study explores its ability to predict not just a single future price but also the probability distribution of possible prices. By integrating probabilistic forecasting with parametric approach into Indonesian stock price prediction, this study also explores new possibilities for enhancing forecasting accuracy and managing risks in financial contexts. This could provide valuable insights into potential risks and uncertainties associated with stock price movements in the Indonesian stock market.

The outcomes of this study not only contribute to the advancement of stock price prediction by offering a robust comparison between TCN and DeepTCN in the context of Indonesian stock market, but also illuminate the strengths and limitations of each method in this specific application. The results serve as valuable guidance for financial analysts and researchers in selecting the most appropriate model for similar tasks in the realm of stock price forecasting, promoting more accurate and reliable predictive models.

This study also introduces a novel method to Indonesian stock price prediction by integrating probabilistic forecasting with a parametric approach. This method has the potential to improve prediction reliability while offering a better way to manage uncertainty, leading to a deeper understanding of stock market trends. By uniting these techniques, the research aims to inspire new ways to address the inherent volatility in financial markets, while fostering innovative approaches to investment planning and risk management. Overall, the insights presented in this paper contribute to the growing field of AI in financial analysis, with potential implications for improving market insights, risk assessment, and investment strategies in Indonesian stock market.

The paper is structured as follows. Section 2 provides a comprehensive review of related research, focusing on the methodologies used in previous studies for stock prediction. This section outlines the various approaches taken by researchers, detailing how they have addressed stock forecasting, and it summarizes the outcomes and insights derived from these studies. It includes a discussion of the evolution of predictive models, leading up to TCN, which have shown promise in handling time series data for stock prediction. Section 3 describes the proposed methodology, outlining the research process with a flowchart, providing information on the dataset and pre-processing steps, and examining the architecture of TCN and DeepTCN in detail. Section 4 presents a detailed description of our experiments and tasks, providing a comprehensive overview of the approaches we use to adjust parameters and examine the model's characteristics, while Section 5 showcases the experimental results and evaluates them using specific metrics. Finally, Section 6 summarizes the conclusions, addresses any limitations, and proposes directions for future research.

## 2. LITERATURE REVIEW

Numerous research has been conducted to forecast the stock market for financial gains, employing a variety of techniques with diverse outcomes. Historically, conventional statistical techniques that involve different types of moving averages and simple forecasting strategies were often employed to predict stock prices [7]. However, Statistical methods are inherently linear, which restricts their effectiveness in predicting stock prices [8]. Since stock data is nonstationary, chaotic, random, and influenced by various technical factors, traditional statistical methods lack the accuracy needed for reliable forecasts [9].

The use of machine learning in stock market prediction has gained significant interest among researchers for its ability to provide efficient and accurate stock prediction [10], [11]. In 2014, an empirical study was conducted by [12] to determine the effectiveness of ARIMA in predicting 56 stocks from seven sectors in National Stock Exchange (NSE). The results state that the ARIMA model exhibits an accuracy of over 85% in predicting stock prices, suggesting its effectiveness in this field and that it has emerged as a leading approach in forecasting time series data for stock price prediction. Over the course of the year, machine learning models have been compared to ARIMA in providing better performance for problems in stock time series forecasting. In a separate study, Artificial Neural Network (ANN) was compared to ARIMA in predicting stock prices. The results indicated that while ANN performed better, the difference in accuracy was not significant [13].

As machine learning evolves, deep learning is becoming more prominent in stock price prediction, providing advanced models that can identify complex patterns in financial data, which traditional machine learning models often struggle with. These deep learning models are increasingly being used to enhance the accuracy and reliability of stock forecasts. Singh and Srivastava [14] introduced a market forecasting model that utilizes principal component analysis for feature extraction, which serves as an input for a Deep Neural Network (DNN). Althelaya assesses the capabilities of various LSTM-based deep learning architectures in predicting financial time series, focusing on both short-term and long-term forecasts. The study compares bidirectional and stacked LSTM models with simpler neural networks and standard LSTM setups to evaluate their relative performance [15], [16]. In a separate study, Recurrent Neural Network (RNN) with attention mechanisms was introduced by [17] to predict multivariate time series, demonstrating its effectiveness in stock price forecasting.

Besides RNN, Convolutional Neural Network (CNN) are also recognized as powerful models for forecasting stock prices. Although CNN was originally designed for computer vision tasks, they can be applied to time series data due to their capability to capture relevant patterns and features in sequential data. CNN were utilized to predict stock prices using historical data and found that the convolutional sliding window technique can effectively capture stock movements [18]. As the study progresses, it has been observed that traditional RNN are prone to gradient explosion, while CNN are often deemed unsuitable for time series analysis [19].

In more recent studies, a form of specialized CNN architecture, namely temporal convolutional networks (TCN), has gained popularity due to its dilated convolutions, which can effectively capture long-range dependencies in time series, making them well-suited for modeling temporal relationships in sequential data. Deng et al. [20] conducted a study on stock trend prediction using TCN model. They discovered that TCN outperforms ARIMA and deep neural networks such as LSTM and CNN in sequence modeling and classification tasks. Another study [19] introduces a feature attention mechanism into the feature extraction process of TCN. Their results show that this approach enhances TCN performance across various error metrics, suggesting that the features processed by TCN with attention lead to improved predictive outcomes.

New models are regularly developed to improve the accuracy and flexibility of TCN in the field of time series forecasting [6], [21], [22]. DeepTCN is one of these innovations, designed specifically for probabilistic forecasting [6]. Unlike traditional point forecasting, which predicts a single future value, probabilistic forecasting provides a distribution of possible future outcomes. This allows for a more detailed understanding of risks and opportunities associated with future events, providing a more comprehensive view of possible outcomes. Jensen, Bianchi, and Anfinsen [23] introduced an innovative approach to probabilistic time series forecasting using DeepTCN and LSTM as their deep learning models. Their method combines Conformal Prediction (CP) to generate Prediction Intervals (PIs) with reliable coverage and ensemble learners that use Quantile Regression (QR) to address heteroscedastic data. Their findings demonstrate that probabilistic forecasting can enhance the performance of deep learning models to address the uncertainty issues in analyzing time series data.

To our knowledge, no comparative study has assessed the effectiveness of DeepTCN against TCN. This gap in research creates an opportunity to investigate the potential strengths and limitations of both models, offering insights into which might be more suitable for specific applications or datasets. By addressing this, our study aims to contribute to a better understanding of these models and their practical implications.

## 3. METHODOLOGY

In this section, we provide a comprehensive overview of the research methodology employed in this study. The model used in this study will also be described in detail, providing a structured way to analyze the collected data. Fig. 1 illustrates the research workflow, outlining the sequential steps taken from data collection to model evaluation.

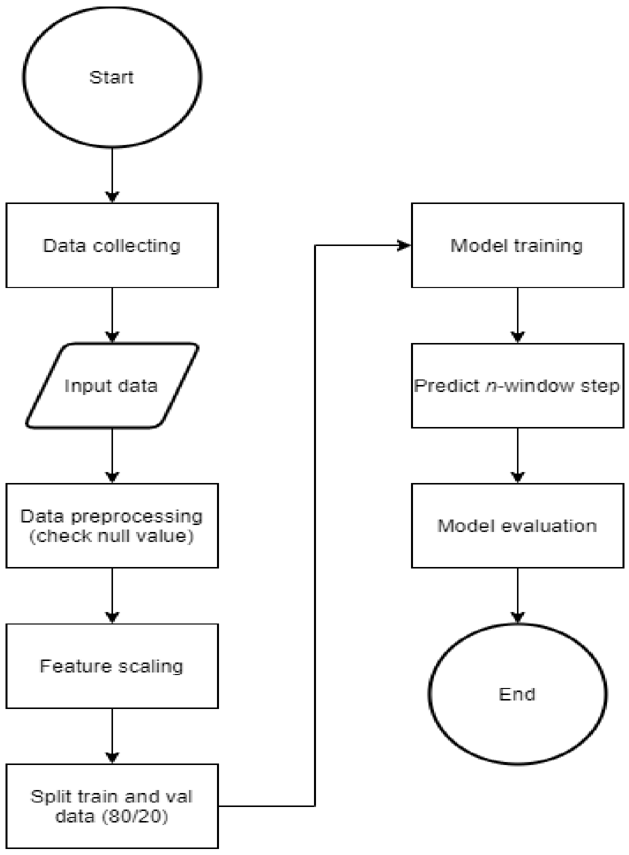


Figure 1. Main research framework

### A. Data Collection

The dataset for this stock prediction was obtained from the Kaggle public dataset titled 'Dataset Saham Indonesia / Indonesia Stock Dataset' (www.kaggle.com/datasets/muamkh/ihsgstockdata). This dataset contains historical data of stocks listed on the Indonesia Stock Exchange from April 16, 2001, to January 6, 2023 with time intervals ranging from minutes to hourly and daily. It provides comprehensive stock price information, with features including Open, Close, High, Low, and Volume of transactions for stocks from various sectors within the Indonesian stock market. The dataset is sourced from Yahoo Finance's public data and the IDX website, providing vital resources for analyzing and predicting stock price movements in the Indonesian stock market.

For this study, we use three sectors from the Indonesian stock market, represented by the following companies: Ace Hardware Indonesia Tbk (ACES) from the cyclical sector, PT Bank Negara Indonesia (Persero) Tbk (BBNI) from the finance sector, and PT Indofood Sukses Makmur Tbk (INDF) from the non-cyclical sector. The transactions we used for these stocks span from January 1, 2017 (2017-01-01), to January 6, 2023 (2023-01-06).

Some detailed information about these companies are gathered from Indonesia Stock Exchange website (www.idx.co.id) and is provided within Table I, Table II, and Table III below:

TABLE I. ACES STOCK DETAILS

| Attribute | Details |
|---|---|
| Ticker | ACES |
| Company Name | Ace Hardware Indonesia Tbk |
| Sector | Consumer Discretionary (Cyclical) |
| EPS (Q1 2024) | 11.94 |
| Revenue (2023) | IDR. 7.6 Trillion |
| Listing Date | November 6, 2007 |
| Website | www.acehardware.co.id |

TABLE II. BBNI STOCK DETAILS

| Attribute | Details |
|---|---|
| Ticker | BBNI |
| Company Name | PT Bank Negara Indonesia (Persero) Tbk |
| Sector | Finance |
| EPS (Q1 2024) | 142.81 |
| Revenue (2023) | IDR 68.3 Trillion |
| Listing Date | November 25, 1996 |
| Website | www.bni.co.id |

TABLE III. INDF STOCK DETAILS

| Attribute | Details |
|---|---|
| Ticker | INDF |
| Company Name | PT Indofood Sukses Makmur Tbk |
| Sector | Consumer Staples (Non-Cyclical) |
| EPS (Q1 2024) | 279 |
| Revenue (2023) | IDR 111.7 Trillion |
| Listing Date | July 14, 1994 |
| Website | www.indofood.com |

Table IV provides a sample of the first five records from the ACES stock, detailing the key features for each trading day. The data we extracted contains five main features: Open, Close, High, Low, and Volume. "Open" and "Close" indicate the starting and ending prices of the stock for a given trading day. "High" and "Low" denote the highest and lowest prices reached during that day, while "Volume" represents the total number of shares traded.

TABLE IV. SUMMARY OF STOCK EXTRACTION RESULTS

| Features | | | | | |
|---|---|---|---|---|---|
| **Timestamp** | **Open** | **Close** | **High** | **Low** | **Volume** |
| 2017-01-01 | 835 | 835 | 835 | 835 | 0 |
| 2017-01-02 | 835 | 795 | 835 | 810 | 10432600 |
| 2017-01-03 | 810 | 780 | 810 | 800 | 6997800 |
| 2017-01-04 | 805 | 785 | 810 | 795 | 2313800 |
| 2017-01-05 | 800 | 785 | 800 | 795 | 2208600 |

## B. *Pre-Processing*

Pre-processing is a vital step in data analysis, ensuring raw data is transformed and structured for further use. This process is critical for maintaining consistency, accuracy, and readiness for modeling or analysis. In this study, the pre-processing steps are as follows:

- The dataset used in this study consists of 1,570 stock records collected from January 1, 2017, to January 6, 2023. We divided this dataset into two parts: training and validation, following an 80:20 ratio. As a result, 1,256 data points are allocated for training, and 314 are set aside for validation.
- Before being processed by the model, the data needs to be normalized to make its distribution more readable for the model. To normalize the data, we apply MinMax scaling, specifically because we use multiple features such as opening price, closing price, high price, low price, and volume. This scaling method ensures all these features are on a consistent scale, minimizing the risk that some features disproportionately affect the model due to their differing magnitudes.
- For feature selection, we categorize the data into two groups: univariate and multivariate. In univariate approach, there's only one feature, and the goal is to predict this feature itself. Here, the target for prediction is the closing price. Multivariate approach considers multiple variables simultaneously, with the objective of predicting all these variables as targets at once. The features used for multivariate prediction are the opening price, closing price, high price, low price, and volume.

## C. *TCN*

Temporal Convolutional Network (TCN) is a category of convolutional neural network (CNN) uniquely designed to effectively manage time series data [24]. Originally proposed for action segmentation and detection [25], TCN consists of a series of cascaded 1D convolutional layers, enabling the mapping of inputs of varying lengths to output sequences of same length [26]. The network structure of a TCN expands upon that of a 1D CNN, where multiple layers of 1D convolutions are layered consecutively. The fundamental of 1D convolution layer is depicted in (1) [27].

$$F(x_t) = (x * f)(t) = \sum_{j=0}^{k-1} f_j^T x_{t-j}, t \geq k$$
$$u = (F(x_k), F(x_{k+1}), \dots, F(x_n)) \quad (1)$$

Table V outlines the TCN algorithm for stock price prediction, offering a step-by-step guide on the architecture and key components. It covers the specific configurations employed to handle time-series data. Additionally, it includes information on the hyperparameters, training strategies, and specific metrics utilized to validate the model's performance.

TABLE V. TCN ALGORITHM

```
Algorithm 1 TCN
1   INPUT
2   data
3   arch
4   OUTPUT
5   p(val)
6   eval
7   check_null(data)
8   scaler(data)
9   data(train), data(valid)
10  Model <- build_model(arch)
11  Model <- train(data(train))
12  p(val) <- predict(Model. window_step, data(val))
13  MSE, MAPE, RMSE <- (data(valid), p(val))
14  Return MSE, MAPE, RMSE
```

### 1. *Causal Convolutions*

Unlike conventional CNNs, TCN employs causal and dilated convolutions. In causal convolutions, the output at a given time point *t* is convolved solely with elements from time *t* and earlier in the preceding layer. This ensures that there is no information leakage from future time points to past ones [28]. A causal convolutional layer addresses the issue by appending zero padding of length $k-1$ at the start of the input sequence which is represented in (2) [27].

$$F(x_t) = (x * f)(t) = \sum_{j=0}^{k-1} f_j^T x_{t-j} \qquad x_{\leq 0} \coloneqq 0$$
$$u = (F(x_1), F(x_2), \dots, F(x_n)) \quad (2)$$

### 2. *Dilated Convolutions*

An additional concern with the basic 1D CNN is its receptive field, which scales linearly with the number of layers. This is undesirable for our purpose as we intend to capture long-term dependencies. To address this, dilated convolution is employed as a technique that enables receptive fields to grow exponentially with the number of layers. More precisely, when integrated with causal convolution, the dilated convolutional layer at the *r*-th level can be expressed using (3) [27]

$$F(x_t) = (x *_{l_r} f)(t) = \sum_{j=0}^{k-1} f_j^T x_{t-l_r \cdot j} \qquad x_{\leq 0} := 0$$

$$u = (F(x_1), F(x_2), \ldots, F(x_n)) \qquad (3)$$

## D. DeepTCN

Deep Temporal Convolutional Network (DeepTCN) is a forecasting model that does not rely on autoregressive methods. Instead, it employs CNN to analyze extensive sets of interconnected time series data. DeepTCN is a variant of the Temporal Convolutional Network (TCN) developed by [6], with an architecture that includes two distinct probabilistic forecasting frameworks. The first framework utilizes a parametric approach, enabling the generation of probabilistic forecasts for future observations by directly predicting the parameters of the hypothetical distribution through maximum likelihood estimation. The second framework, on the other hand, is nonparametric and creates a set of forecasts based on specific quantile values of interest.

Both TCN and DeepTCN utilize encoder-decoder architecture to process time series data as shown in Fig. 2. The encoder extracts features, and the decoder uses those features to generate predictions. Both models use causal convolutions to ensure that they only rely on previous information when making predictions, which aligns with the causality principles in time series data [6]. Dilated convolutions might be employed to expand the receptive field and capture long-range dependencies within the sequence.

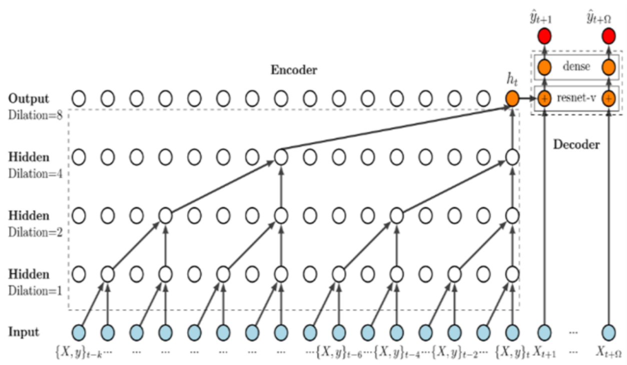


Figure 2. TCN Architecture

DeepTCN builds upon the TCN architecture, but the key difference lies in its use of stacked residual blocks as shown in Fig. 3. These blocks allow the network to learn even more intricate temporal relationships by creating a "shortcut" path for the information to flow [6]. This helps address the vanishing gradient problem, a common challenge in training deep neural networks on long sequences [6].

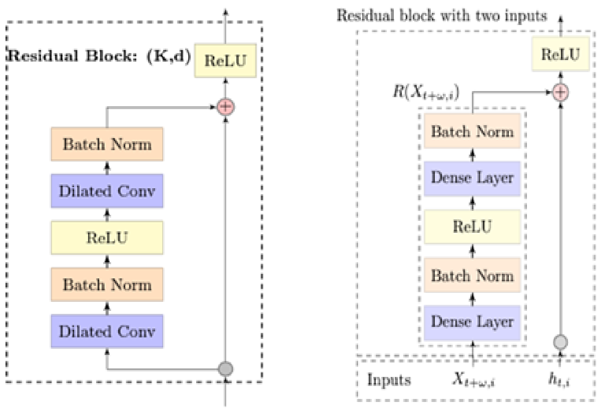


Figure 3. Stacked Residual Blocks

Table VI provides an overview of the DeepTCN method for forecasting stock prices, maintaining the same input and output structure as TCN algorithm. The primary variation lies in line 10, where the model incorporates a parametric approach into the algorithm.

TABLE VI. DEEPTCN ALGORITHM

```
Algorithm 2 DeepTCN
1   INPUT
2   Data
3   Arch
4   OUTPUT
5   p(val)
6   eval
7   check_null(data)
8   scaler(data)
9   data(train), data(valid)
10  Model <- build_model(arch, parametric_approach)
11  Model <- train(data(train))
12  p(val) <- predict(Model. window_step, data(val))
13  MSE, MAPE, RMSE <- (data(valid), p(val))
14  Return MSE, MAPE, RMSE
```

### 1. *Parametric Approach*

In statistics, the parametric approach assumes that the data follows a specific distribution characterized by parameters such as mean and standard deviation. The predetermined distribution is utilized in the parametric approach, and the maximum likelihood estimation is employed to determine the associated parameters. In [6], the parameters of the distribution generated by the network are explained using Gaussian distributions, namely the mean ($\mu$) and standard deviation ($\sigma$), for each target value ($y$). Afterwards, the loss function is constructed as the negative log-likelihood function in (4).

$$L_G = -\log \ell\,(\mu, \sigma \mid y)$$

$$= -\,log(\,(2\pi\sigma^2)^{-1/2} exp[-(y-\mu)^2 /\, (2\sigma^2)]\,)$$

$$= \frac{1}{2}\; log(\,(2\pi)\, + \, log(\sigma)\, + \,\frac{(y-\mu)^2}{2\sigma^2}) \qquad (4)$$

In this study, we will use half-normal distribution as our parametric approach. The half-normal distribution is a variation of the folded normal and truncated normal distributions [29]. The half-normal distribution is not symmetrical, with values extending from zero to positive infinity. It can be visualized as a standard normal distribution that's been "folded" at its mean [30], creating a distribution where all values are positive. This type of distribution is characteristic of the half-normal distribution, where the "folding" at zero eliminates all negative values, resulting in a shape that's like the right half of a typical normal distribution. The probability density function of half-normal distribution is defined in (5)

$$y = f(x|\mu, \sigma) = \sqrt{\frac{2}{\pi}\frac{1}{\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}\,; x \geq \mu \qquad (5)$$

where $\mu$ defines the location parameter and $\sigma$ defines the scale parameter. The probability density function (pdf) is undefined if $x \leq \mu$.

### 2. *Nonparametric Approach*

In the nonparametric approach, predictions within the quantile regression framework can be derived [6]. In quantile regression, the observed value and the predicted value for a specific quantile level $q$ are denoted as $y$ and $\hat{\boldsymbol{y}}^{\boldsymbol{q}}$, respectively [6]. The models are trained to reduce the quantile loss, as specified in (6).

$$L_q = (y, \hat{\boldsymbol{y}}^{\boldsymbol{q}}) = q(y - \hat{\boldsymbol{y}}^{\boldsymbol{q}})^+ + (1-q)(\hat{\boldsymbol{y}}^{\boldsymbol{q}} - y)^+ \qquad (6)$$

where, $(y)^+ = \max(0, y)$ and $q \in (0,1)$. The corresponding forecasts for a set of quantile levels $Q = (q1, \ldots, qm)$ can be derived by minimizing the total quantile loss, which is defined in (7).

$$L_Q = \sum_{j=1}^{m} L_{q_j}(y, \hat{\boldsymbol{y}}^{\boldsymbol{q_j}}) \qquad (7)$$

## 4. EXPERIMENTAL PROCEDURE

### A. *Experiment Setup*

The experiments were conducted on a high-performance workstation featuring an AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx CPU @ 2.1GHz, 20GB of RAM, and an AMD Radeon (TM) Vega 8 GPU. This hardware configuration provided the computational power required for training and evaluating machine learning models efficiently. The machine learning models were implemented using Python programming language (version 3.9.7). We utilized popular libraries such as NumPy, Pandas, and Scikit-learn for data preprocessing, and model evaluation. As for the model used (TCN), the library used is Darts. All experiments were conducted using Python within a Jupyter Notebook environment.

### B. *Hyperparameter Tuning*

The Darts implementation incorporates a TCN model. TCN and DeepTCN will have the same configuration in terms of basic structure. The parameters used for both TCN and Deep TCN models are as follows:

1) *Batch_size:* processes input data in batches of 32 sequences at a time during training.
2) *Epoch:* the model is trained for 50 epochs.
3) *Input_chunk_length:* Each input sequence is chunked into segments of length 300. This parameter determines how much historical data the model considers at each step.
4) *Output_chunk_length*: The model produces output sequences of length 30. This parameter determines the length of the output sequences it generates.
5) *Dropout:* Regularization technique used to prevent overfitting. Here, a dropout rate of 0.2 is applied, meaning that 20% of the neurons are randomly set to zero during each training epoch to prevent them from overly relying on specific inputs.
6) *Kernel_size:* Convolutional layers with a kernel size of 3. This parameter defines the size of the filter that moves across the input data during convolution.
7) *Num_filters:* The model has 4 filters in its convolutional layers. Filters are the building blocks of convolutional neural networks and are responsible for detecting features in the input data.
8) *Optimizer:* Adam optimizer is used for optimizing the model's parameters during training.

Although they share this similarity, DeepTCN in this study uses a parametric approach called half-normal disribution. The half-normal distribution is a continuous probability distribution with values that are strictly positive (where x ≥ 0). It is created by taking a standard normal distribution and extracting the absolute values of its random variable, eliminating any negative values. This results in a distribution that only represents positive outcomes. In the training process, we add a parameter known as past covariates. These are external variables that offer historical context for the time series data we're studying. They act as features that describe past conditions or events that might influence the target variable [31]. In this study, we'll use volume as our past covariate.

### C. *Evaluation Metrics*

This study uses mostly error assessment metric to see how good the model is to predict stock prices. The metrics that are used in this study are:

1) *Mean Squared Error (MSE):* The Mean Squared Error quantifies the average squared deviation between the actual values and the predicted values [32]. It assigns equal importance to both large and small errors, which makes it particularly sensitive to outliers. MSE is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(A_i - P_i)^2 \tag{8}$$

where $n$ = number of data points, $A_i$ = actual value, $P_i$ = predicted value

2) *Root Mean Squared Error (RMSE):* The Root Mean-Squared Error (RMSE) is calculated by taking the square root of the MSE and serves` as an indicator for typical size of errors in the original data, it prioritizes significant errors over minor ones. In contrast to MSE, RMSE offers an error metric that uses the same units as the target variable [32]. RMSE is calculated as:

$$RMSE = \sqrt{MSE} \tag{9}$$

3) *Mean Average Percentage Error (MAPE):* The Mean Average Percentage Error calculates the average percentage difference between the actual and predicted values [33]. MAPE is expressed as a percentage, making it easier to interpret. MAPE is calculated as:

$$MAPE = \frac{\sum \frac{|A-P|}{A}\, x\, 100}{N} \tag{10}$$

where *A* represents the actual value, *P* is the predicted value, and *N* is the number of observations.

## 5. Result and Discussion

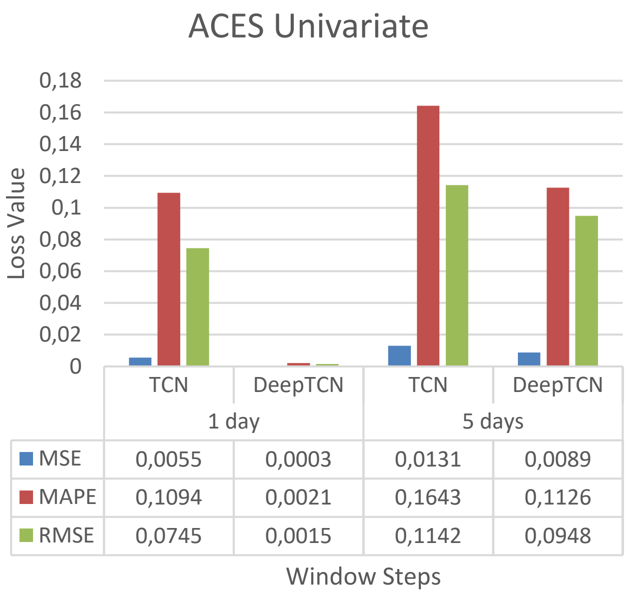
### A. *Performance Evaluation*

For the performance evaluation, we established a naive model as a baseline to provide a point of comparison for TCN & DeepTCN in 1 day prediction. The naive model, often representing a simple yet effective prediction method, helps to contextualize the improvements brought by more complex architectures. By comparing the predictive accuracy and other performance metrics of the TCN and DeepTCN against the naive model, we can assess the capability of these advanced models. Our findings as shown in Table VII, Table VIII, and Table IX indicate that both TCN and DeepTCN significantly outperform the naïve model, showcasing their capability to capture intricate temporal patterns and dependencies in the stock data.

We also evaluate the models into two categories for each stock, univariate and multivariate. Univariate evaluation examines single variables independently, whereas multivariate evaluation explores the connections between several variables to offer a more profound understanding of the studied phenomenon. In this study, the univariate variable to be predicted is the closing price. For multivariate analysis, the variables to be examined include the opening price, closing price, high price, and low price. Window steps refer to the number of units a window moves when creating subsets of data for time series analysis or forecasting.

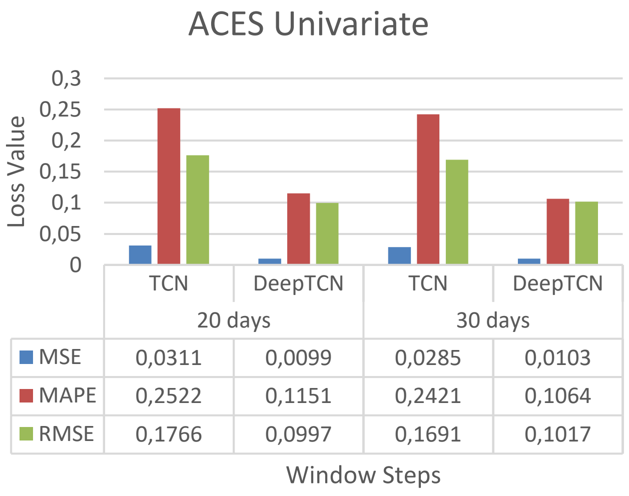
TABLE VII. 1 Day ACES prediction between TCN, DeepTCN and Naïve model

| | **1 day** | | |
|---|---|---|---|
| | TCN | DeepTCN | Naïve |
| **MSE** | 0,0055 | **0,0003** | 561,3642 |
| **MAPE** | 0,1094 | **0,0021** | 1,8991 |
| **RMSE** | 0,0745 | **0,0015** | 23,6931 |



| | 1 day | | 5 days | |
|---|---|---|---|---|
| | TCN | DeepTCN | TCN | DeepTCN |
| 🟦 MSE | 0,0055 | 0,0003 | 0,0131 | 0,0089 |
| 🟥 MAPE | 0,1094 | 0,0021 | 0,1643 | 0,1126 |
| 🟩 RMSE | 0,0745 | 0,0015 | 0,1142 | 0,0948 |

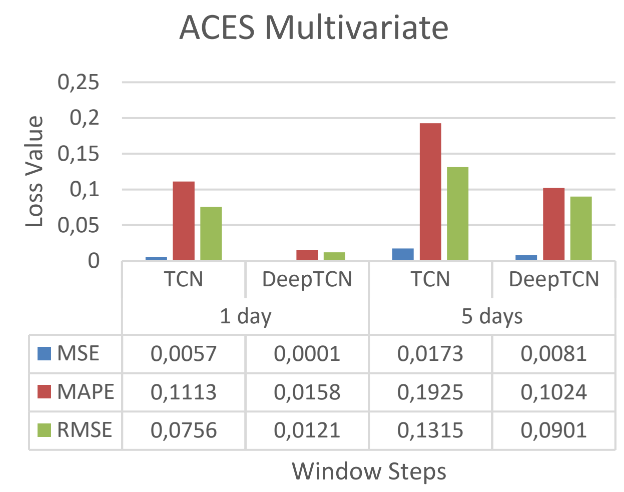
Figure 4. ACES comparison between TCN and DeepTCN for univariate data (1 and 5 days)



| | 20 days | | 30 days | |
|---|---|---|---|---|
| | TCN | DeepTCN | TCN | DeepTCN |
| 🟦 MSE | 0,0311 | 0,0099 | 0,0285 | 0,0103 |
| 🟥 MAPE | 0,2522 | 0,1151 | 0,2421 | 0,1064 |
| 🟩 RMSE | 0,1766 | 0,0997 | 0,1691 | 0,1017 |

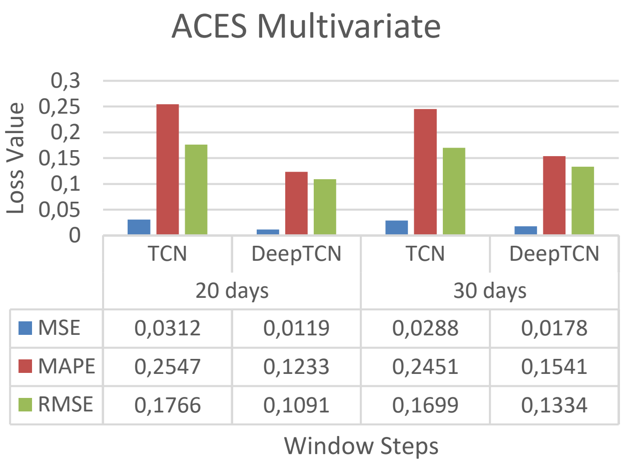
Figure 5. ACES comparison between TCN and DeepTCN for univariate data (20 and 30 days)

According to Fig. 4 and Fig. 5, DeepTCN consistently surpasses TCN in performance across all window steps. For a 1-day window step, the MSE is reduced by about 18%, from 0.0055 with TCN to 0.0003 with DeepTCN. With a 5-day window steps, the MSE drops by around 32.06%, decreasing from 0.0131 (TCN) to 0.0089 (DeepTCN). Similarly, for the 20-day and 30-day window steps, the MSE reductions are even more significant, with differences of up to 68% and 63%, respectively. As for the MAPE score, DeepTCN also outperforms TCN. In a 1-day window step, DeepTCN managed to reduce the MAPE value by 98%. In 5-day window steps, the decrease in MAPE is not as drastic, only around 31.8%. Meanwhile, for window steps of 20 and 30 days, DeepTCN successfully reduced the MAPE values by 54.35% and 56.07% respectively. The RMSE values also exhibit the same pattern, with DeepTCN still surpassing TCN. In a 1-day window step, DeepTCN achieved 0.0015, whereas TCN had an RMSE value of 0.0745. For 5-day window steps, the RMSE value decreased by 17%. And for 20 and 30-day window steps, there were reductions of 43.55% and 39.87% respectively.



| | 1 day | | 5 days | |
|---|---|---|---|---|
| | TCN | DeepTCN | TCN | DeepTCN |
| 🟦 MSE | 0,0057 | 0,0001 | 0,0173 | 0,0081 |
| 🟥 MAPE | 0,1113 | 0,0158 | 0,1925 | 0,1024 |
| 🟩 RMSE | 0,0756 | 0,0121 | 0,1315 | 0,0901 |

Figure 6. ACES comparison between TCN and DeepTCN for multivariate data (1 and 5 days)



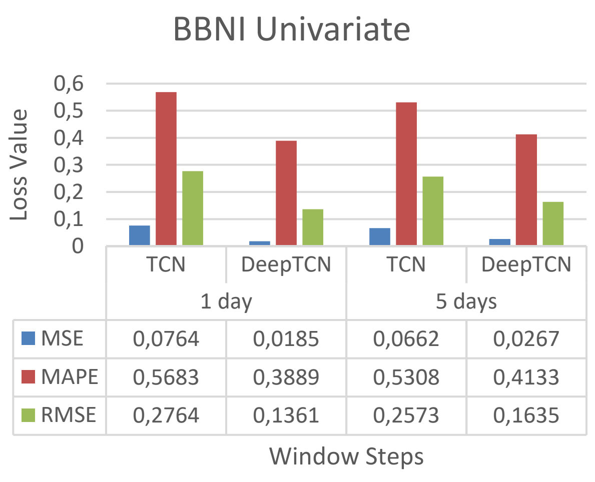
| | 20 days | | 30 days | |
|---|---|---|---|---|
| | TCN | DeepTCN | TCN | DeepTCN |
| 🟦 MSE | 0,0312 | 0,0119 | 0,0288 | 0,0178 |
| 🟥 MAPE | 0,2547 | 0,1233 | 0,2451 | 0,1541 |
| 🟩 RMSE | 0,1766 | 0,1091 | 0,1699 | 0,1334 |

Figure 7. ACES comparison between TCN and DeepTCN for multivariate data (20 and 30 days)

DeepTCN outperforms TCN across all window steps even when dealing with multivariate data, as shown in Fig. 6 and Fig. 7. With a 1-day window step, DeepTCN reduces the MSE by 61.38% compared to TCN. For a 5-day window steps, TCN has an MSE of 0.0173, while DeepTCN achieves a much lower MSE of 0.0081. Similarly, DeepTCN delivers MSE reductions of 61.8% for a 20-day window step and 38.2% for a 30-day window step. For the MAPE value in a 1-day window step, there is a significant difference with DeepTCN achieving a result of 0.0158 and TCN achieving 0.1113. During a 5-day window steps, DeepTCN managed to reduce the MAPE value generated by TCN by 46.8%. And for 20 and 30-day window steps, there were decreases of 51.61% and 37.12% respectively. The RMSE value also experienced a decrease. In a 1-day window step, DeepTCN reached a value of 0.1091, while TCN achieved a value of 0.1766. For a 5-day window steps, the RMSE value decreased by around 31.54%. A similar trend also occurred for 20 and 30-day window steps, with reductions of approximately 38.23% and 21.5% respectively.

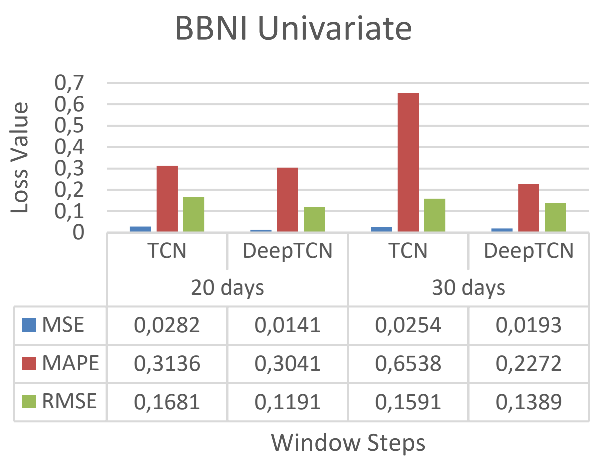
TABLE VIII. 1 DAY BBNI PREDICTION BETWEEN TCN, DEEPTCN AND NAÏVE MODEL

| | 1 day | | |
|---|---|---|---|
| | TCN | DeepTCN | Naïve |
| **MSE** | 0,0764 | **0,0185** | 19394,969 |
| **MAPE** | 0,5683 | **0,3889** | 1,1697 |
| **RMSE** | 0,2764 | **0,1361** | 139,2658 |



| | TCN | DeepTCN | TCN | DeepTCN |
|---|---|---|---|---|
| | 1 day | | 5 days | |
| 🟦 MSE | 0,0764 | 0,0185 | 0,0662 | 0,0267 |
| 🟥 MAPE | 0,5683 | 0,3889 | 0,5308 | 0,4133 |
| 🟩 RMSE | 0,2764 | 0,1361 | 0,2573 | 0,1635 |

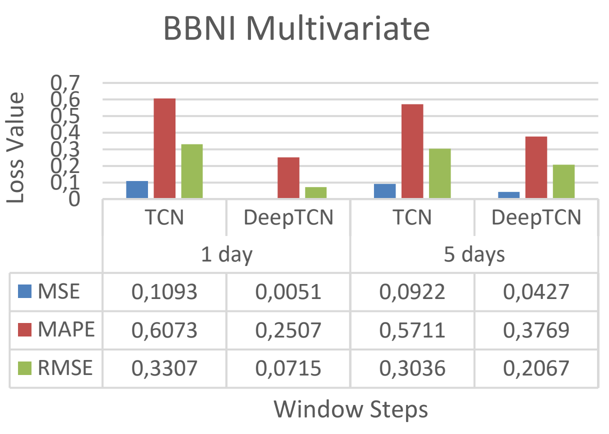
Figure 8. BBNI comparison between TCN and DeepTCN for univariate data (1 and 5 days)



| | TCN | DeepTCN | TCN | DeepTCN |
|---|---|---|---|---|
| | 20 days | | 30 days | |
| 🟦 MSE | 0,0282 | 0,0141 | 0,0254 | 0,0193 |
| 🟥 MAPE | 0,3136 | 0,3041 | 0,6538 | 0,2272 |
| 🟩 RMSE | 0,1681 | 0,1191 | 0,1591 | 0,1389 |

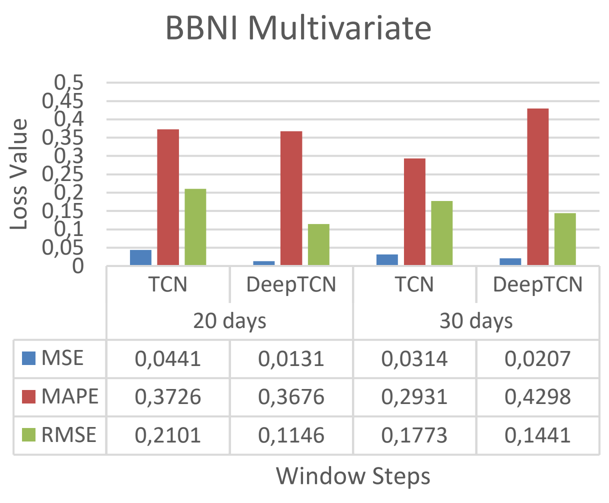
Figure 9. BBNI comparison between TCN and DeepTCN for univariate data (20 and 30 days)

Fig. 8 and Fig. 9 present the error values from each model for BBNI stock price. As with the previous companies, DeepTCN demonstrates better performance, with lower MSE, MAPE, and RMSE values compared to TCN. With a 1-day window step, DeepTCN reduces the MSE by 75.2%. For a 5-day window steps, the MSE drops by 59.3%. The pattern continues with longer window steps: for 20-day and 30-day windows, the MSE decreases by 50% and 24.02%, respectively. TCN achieved 0.5683 over a 1-day window step. Additionally, DeepTCN successfully diminished the MAPE value by 22.14% for a 5-day window step. Similar reductions in MAPE were observed for window steps spanning 20 and 30 days, with declines of 3.03% and 65.24% respectively. Correspondingly, reductions were noted in RMSE values, with a 50.76% decrease for a 1-day window step, a 40.61% decrease for a 5-day window steps, and reductions of 29.16% and 12.7% for window steps of 20 and 30 days respectively.



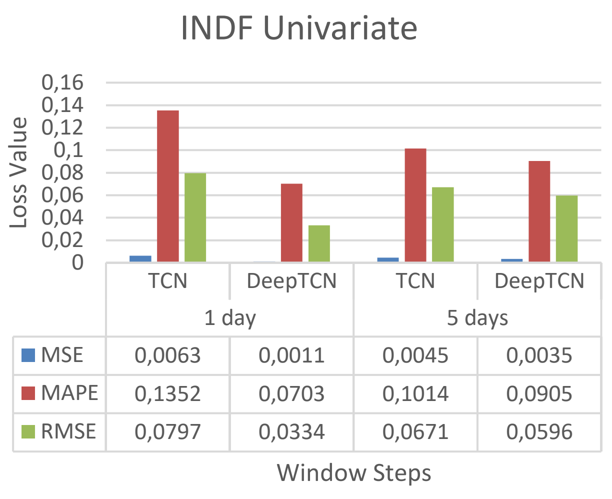
| | TCN | DeepTCN | TCN | DeepTCN |
|---|---|---|---|---|
| | 1 day | | 5 days | |
| 🟦 MSE | 0,1093 | 0,0051 | 0,0922 | 0,0427 |
| 🟥 MAPE | 0,6073 | 0,2507 | 0,5711 | 0,3769 |
| 🟩 RMSE | 0,3307 | 0,0715 | 0,3036 | 0,2067 |

Figure 10. BBNI comparison between TCN and DeepTCN for multivariate data (1 and 5 days)



| | 20 days | | 30 days | |
|---|---|---|---|---|
| | TCN | DeepTCN | TCN | DeepTCN |
| 🟦 MSE | 0,0441 | 0,0131 | 0,0314 | 0,0207 |
| 🟥 MAPE | 0,3726 | 0,3676 | 0,2931 | 0,4298 |
| 🟩 RMSE | 0,2101 | 0,1146 | 0,1773 | 0,1441 |

Figure 11. BBNI comparison between TCN and DeepTCN for multivariate data (20 and 30 days)



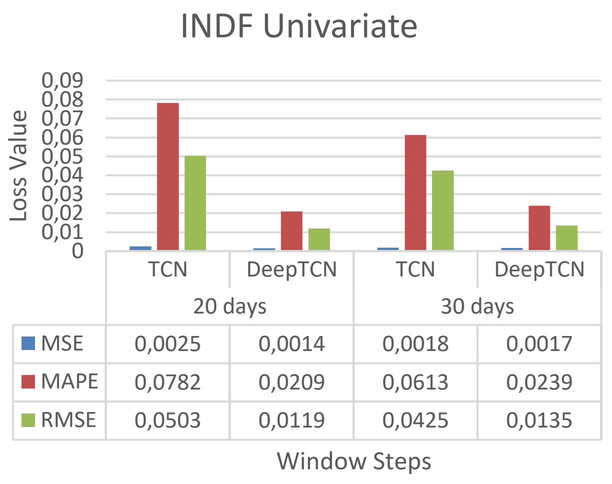
| | 1 day | | 5 days | |
|---|---|---|---|---|
| | TCN | DeepTCN | TCN | DeepTCN |
| 🟦 MSE | 0,0063 | 0,0011 | 0,0045 | 0,0035 |
| 🟥 MAPE | 0,1352 | 0,0703 | 0,1014 | 0,0905 |
| 🟩 RMSE | 0,0797 | 0,0334 | 0,0671 | 0,0596 |

Figure 12. INDF comparison between TCN and DeepTCN for univariate data (1 and 5 days)

Fig. 10 and Fig. 11 show the performance of both models in predicting multivariate data for BBNI. With a 1-day window step, DeepTCN achieves a 64.75% reduction in MSE compared to the other model. For a 5-day window step, the MSE is lowered by 37.63%. This trend continues with longer window steps: the MSE decreases by 70.07% for a 20-day window step and by 34.6% for a 30-day window steps. In terms of MAPE-based model evaluation, DeepTCN continues to outperform TCN, with reductions of approximately 58.72% for a 1-day window step, 34% for a 5-day window steps, 1.34% for a 20-day window steps, and 31.9% for a 30-day window step. Additionally, RMSE values experienced decreases of 78.38% for a 1-day window step, 31.94% for a 5-day window steps, 31.185% for a 20-day window steps, and 18.72% for a 30-day window steps.

TABLE IX. 1 DAY INDF PREDICTION BETWEEN TCN, DEEPTCN AND NAÏVE MODEL

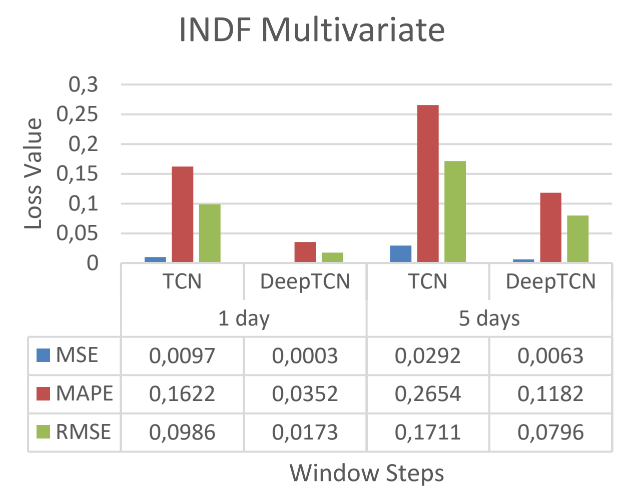
| | 1 day | | |
|---|---|---|---|
| | TCN | DeepTCN | Naïve |
| **MSE** | 0,0063 | **0,0011** | 6675,3193 |
| **MAPE** | 0,1352 | **0,0703** | 0,9078 |
| **RMSE** | 0,0797 | **0,0334** | 81,7026 |



| | 20 days | | 30 days | |
|---|---|---|---|---|
| | TCN | DeepTCN | TCN | DeepTCN |
| 🟦 MSE | 0,0025 | 0,0014 | 0,0018 | 0,0017 |
| 🟥 MAPE | 0,0782 | 0,0209 | 0,0613 | 0,0239 |
| 🟩 RMSE | 0,0503 | 0,0119 | 0,0425 | 0,0135 |

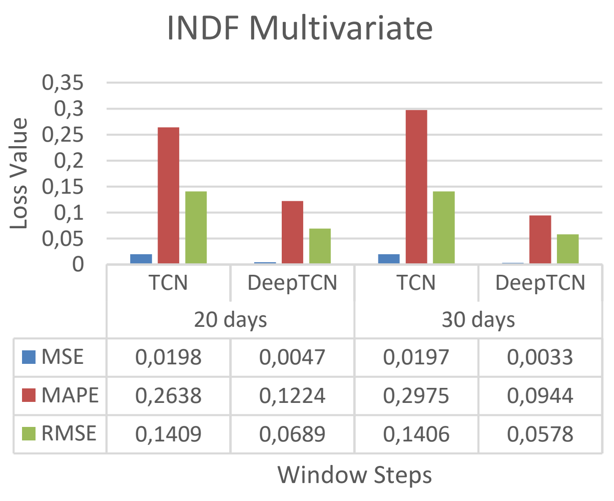
Figure 13. INDF comparison between TCN and DeepTCN for univariate data (20 and 30 days)

In the case of the final company, INDF, DeepTCN continues to outperform TCN, achieving lower MSE, MAPE, and RMSE values, as outlined in Fig. 12 and Fig. 13. With a 1-day window step, DeepTCN reduces the MSE by 82.54%. For a 5-day window step, the MSE drops by 22.3%, from 0.0045 to 0.0035. This trend is also observed with longer window steps: for 20-day and 30-day window steps, the MSE is reduced by 44% and 5.56%, respectively. DeepTCN also demonstrated lower MAPE values compared to TCN. For a 1-day window step, there was a decrease of 47.96% in MAPE value. For a 5-day window step, the MAPE value decreased by 2.32%. For window steps of 20 and 30 days, there were reductions of 73.34% and 60.95% respectively. In terms of RMSE values, DeepTCN managed to reduce them by 58.03% for a 1-day window step, 11.2% for a 5-day window steps, 76.32% for a 20-day window steps, and 68.25% for a 30-day window steps.



| | TCN | DeepTCN | TCN | DeepTCN |
|---|---|---|---|---|
| | 1 day | | 5 days | |
| 🟦 MSE | 0,0097 | 0,0003 | 0,0292 | 0,0063 |
| 🟥 MAPE | 0,1622 | 0,0352 | 0,2654 | 0,1182 |
| 🟩 RMSE | 0,0986 | 0,0173 | 0,1711 | 0,0796 |

Figure 14. INDF comparison between TCN and DeepTCN for multivariate data (1 and 5 days)



| | TCN | DeepTCN | TCN | DeepTCN |
|---|---|---|---|---|
| | 20 days | | 30 days | |
| 🟦 MSE | 0,0198 | 0,0047 | 0,0197 | 0,0033 |
| 🟥 MAPE | 0,2638 | 0,1224 | 0,2975 | 0,0944 |
| 🟩 RMSE | 0,1409 | 0,0689 | 0,1406 | 0,0578 |

Figure 15. INDF comparison between TCN and DeepTCN for multivariate data (20 and 30 days)

The multivariate prediction results for the company INDF, presented in Fig. 14 and Fig. 15, demonstrate a significant reduction in MSE when using DeepTCN compared to TCN. For a 1-day window step, the MSE with DeepTCN is just 0.0003, a remarkable drop from the 0.0097 recorded with TCN. Similarly, the 5-day window step shows a 78.77% reduction in MSE. This trend continues with larger window steps: DeepTCN reduces the MSE by 76.7% for the 20-day window step and by 83.25% for the 30-day window step. Additionally, DeepTCN performs better than TCN in terms of MAPE. With a 1-day window step, the MAPE decreases by 78.29%. The 5-day window steps shows a reduction of 55.46%, while the 20-day and 30-day steps exhibit reductions of 53.65% and 68.45%, respectively. Regarding RMSE values, DeepTCN successfully reduced them by 82.45% for a 1-day window step, 53.48% for a 5-day window step, 51.1% for a 20-day window steps, and 58.89% for a 30-day window steps.

### B. *Results Summary*

In this study, we found that using a parametric approach in DeepTCN significantly improved prediction outcomes, particularly for stock prices. The DeepTCN model with this parametric approach outperformed traditional TCN models without parametric features. Even when tested across three different stocks, the DeepTCN model showed consistent and stable performance. Both of these models perform much better than a basic, simple naïve model when it comes to predicting stock data.

We used a parametric approach based on the half-normal distribution which ensures that the dataset's variance remains positive, a critical aspect when dealing with stock prices where negative values are not possible. Although we chose half-normal distribution, there are many other parametric approaches available, depending on the dataset's specific needs and characteristics. Some approaches can handle negative values, while others work with univariate or multivariate data, discrete or continuous data. The choice of approach should align with the dataset's requirements and the goals of the analysis.

## 6. Conclusion

In this study, a comparison analysis was conducted between TCN and DeepTCN models in forecasting stock prices using 3 Indonesian stock historical price data. In summary, DeepTCN demonstrates its superiority in stock price prediction compared to TCN. DeepTCN is capable of outperforming TCN by achieving lower values of MSE, MAPE, and RMSE. The half-normal distribution parametric approach used in this study has proven to make DeepTCN better at capturing fluctuating stock trends.

The future work recommended in this study involves evaluating the performance of TCN and DeepTCN on a varied set of datasets. This evaluation aims to understand the capabilities of DeepTCN across different types of data characteristics and tasks. By assessing these models on diverse datasets, researchers can gain insights into how well DeepTCN generalizes and performs in various scenarios.

When considering parametric approaches, it is important to choose a method that aligns with the characteristics of the dataset and the specific goals of the analysis. This evaluation can provide valuable insight for understanding the strengths and limitations of DeepTCN and guide its application in real world datasets across different domains.

## REFERENCES

[1] A. Aali-Bujari, F. Venegas-Martínez, and G. Pérez-Lechuga, "Impact of the stock market capitalization and the banking spread in growth and development in Latin American: A panel data estimation with System GMM," Contad. Adm., vol. 62, no. 5, pp. 1427–1441, 2017.

[2] M.-W. Hsu, S. Lessmann, M.-C. Sung, T. Ma, and J. E. V. Johnson, "Bridging the divide in financial market forecasting: machine learners vs. financial economists," Expert Syst. Appl., vol. 61, pp. 215–234, 2016.

[3] L.-P. Chen, "Using machine learning algorithms on prediction of stock price," J. Model. Optim., vol. 12, no. 2, pp. 84–99, 2020.

[4] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017.

[5] Z. Hu, Y. Zhao, and M. Khushi, "A survey of Forex and stock price prediction using deep learning," Appl. Syst. Innov., vol. 4, no. 1, p. 9, 2021.

[6] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," Neurocomputing, vol. 399, pp. 491–501, 2020.

[7] I. Bhattacharjee and P. Bhattacharja, "Stock price prediction: A comparative study between traditional statistical approach and machine learning approach," in 2019 4th International Conference on Electrical Information and Communication Technology (EICT), 2019.

[8] A. Durgapal and V. Vimal, "Prediction of stock price using statistical and ensemble learning models: A comparative study," in 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), 2021.

[9] H. Grigoryan, "A stock market prediction method based on support vector machines (svm) and independent component analysis (ica)," Database Systems Journal, vol. 7, no. 1, pp. 12–21, 2016.

[10] M. Usmani, S. H. Adil, K. Raza, and S. S. A. Ali, "Stock market prediction using machine learning techniques," in 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), 2016.

[11] K. Raza, "Prediction of Stock Market performance by using machine learning techniques," in 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT), 2017.

[12] P. Mondal, L. Shit, and S. Goswami, "Study of effectiveness of time series modeling (Arima) in forecasting stock prices," Int. J. Comput. Sci. Eng. Appl., vol. 4, no. 2, pp. 13–29, 2014.

[13] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of ARIMA and artificial neural networks models for stock price prediction," J. Appl. Math., vol. 2014, pp. 1–7, 2014.

[14] R. Singh and S. Srivastava, "Stock prediction using deep learning," Multimed. Tools Appl., vol. 76, no. 18, pp. 18569–18584, 2017.

[15] K. A. Althelaya, E.-S. M. El-Alfy, and S. Mohammed, "Evaluation of bidirectional LSTM for short-and long-term stock market prediction," in 2018 9th International Conference on Information and Communication Systems (ICICS), 2018.

[16] A. Sethia and P. Raut, Application of lstm, gru and ica for stock price prediction," in Information and communication technology for intelligent systems. Springer, 2019.

[17] Y. Liu, C. Gong, L. Yang, and Y. Chen, "DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction," Expert Syst. Appl., vol. 143, no. 113082, p. 113082, 2020.

[18] Hiransha, Gopalakrishnan, V. K. Menon, and Soman, "NSE stock market prediction using deep-learning models," Procedia Comput. Sci., vol. 132, pp. 1351–1362, 2018.

[19] Y. Fu and H. Xiao, "Stock price prediction model based on dual attention and tcn," SSRN Electron. J., 2022.

[20] S. Deng, N. Zhang, W. Zhang, J. Chen, J. Z. Pan, and H. Chen, "Knowledge-driven stock trend prediction and explanation via temporal convolutional network," in Companion Proceedings of The 2019 World Wide Web Conference, 2019.

[21] H. V. Dudukcu, M. Taskiran, Z. G. Cam Taskiran, and T. Yildirim, "Temporal Convolutional Networks with RNN approach for chaotic time series prediction," Appl. Soft Comput., vol. 133, no. 109945, p. 109945, 2023.

[22] H. Liu, T. Zhao, S. Wang, and X. Li, "A stock rank prediction method combining industry attributes and price data of stocks," Inf. Process. Manag., vol. 60, no. 4, p. 103358, 2023.

[23] V. Jensen, F. M. Bianchi, and S. N. Anfinsen, "Ensemble conformalized quantile regression for probabilistic time series forecasting," IEEE Trans. Neural Netw. Learn. Syst., vol. PP, 2022.

[24] P. Lara-Benítez, M. Carranza-García, J. M. Luna-Romera, and J. C. Riquelme, "Temporal Convolutional Networks applied to energy-related time series forecasting," Appl. Sci. (Basel), vol. 10, no. 7, p. 2322, 2020.

[25] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks for Action Segmentation and Detection," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[26] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep learning for time series forecasting: Advances and open problems," Information (Basel), vol. 14, no. 11, p. 598, 2023.

[27] J. You, Y. Wang, A. Pal, P. Eksombatchai, C. Rosenburg, and J. Leskovec, "Hierarchical temporal convolutional networks for dynamic recommender systems," in The World Wide Web Conference, 2019.

[28] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting," Electronics (Basel), vol. 8, no. 8, p. 876, 2019.

[29] K. Cooray and M. M. A. Ananda, "A generalization of the half-normal distribution with applications to lifetime data," Commun. Stat. Theory Methods, vol. 37, no. 9, pp. 1323–1337, 2008.

[30] M. Tsagris, C. Beneki, and H. Hassani, "On the Folded Normal Distribution," Mathematics, vol. 2, no. 1, pp. 12–28, 2014.

[31] J. Palet, V. Manquinho, and R. Henriques, "Multiple-input neural networks for time series forecasting incorporating historical and prospective context," Data Min. Knowl. Discov., 2023.

[32] V. Plevris, G. Solorzano, N. Bakas, and M. Ben Seghier, "Investigation of performance metrics in regression analysis and machine learning-based prediction models," in 8th European Congress on Computational Methods in Applied Sciences and Engineering, 2022.

[33] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," Int. J. Forecast., vol. 32, no. 3, pp. 669–679, 2016.

**Felix** is a graduate student pursuing a Master's degree in Computer Science at Bina Nusantara University. He has a strong enthusiasm for creating deep learning models, analyzing sentiment, and gaining insights from data science. His research interests involve exploring artificial intelligence, machine learning, and deep learning applications in the field of finance.

**Evandiaz Fedora** is a graduate student pursuing a Master's degree in Computer Science at Bina Nusantara University. He has a passion for artificial intelligence and creating deep learning models. His research interests involve exploring artificial intelligence, machine learning, and deep learning applications in the field of finance.

**Alexander Agung Santoso Gunawan** is a Associate Professor at Bina Nusantara University, Jakarta in Indonesia, with 15 years of research experience. He holds a Bachelor of Science in Mathematics from Bandung Institute of Technology, a Master of Automation Engineering from Darmstadt University of Applied Sciences, Germany and an Master of Electrical Engineering from Bandung Institute of Technology. He completed his PhD in Computer Science at University of Indonesia. Alexander has published over 100 research papers. His research interests include Data Science, Geo-AI, Computer Vision, IoT, and Robotics.