Neural Processing Letters (2024) 56:223
https://doi.org/10.1007/s11063-024-11656-3

Leveraging Hybrid Deep Learning Models for Enhanced Multivariate Time Series Forecasting

Amal Mahmoud[1] · **Ammar Mohammed**[1]

Accepted: 21 May 2024

Abstract
Time series forecasting is crucial in various domains, ranging from finance and economics to weather prediction and supply chain management. Traditional statistical methods and machine learning models have been widely used for this task. However, they often face limitations in capturing complex temporal dependencies and handling multivariate time series data. In recent years, deep learning models have emerged as a promising solution for overcoming these limitations. This paper investigates how deep learning, specifically hybrid models, can enhance time series forecasting and address the shortcomings of traditional approaches. This dual capability handles intricate variable interdependencies and non-stationarities in multivariate forecasting. Our results show that the hybrid models achieved lower error rates and higher R^2 values, signifying their superior predictive performance and generalization capabilities. These architectures effectively extract spatial features and temporal dynamics in multivariate time series by combining convolutional and recurrent modules. This study evaluates deep learning models, specifically hybrid architectures, for multivariate time series forecasting. On two real-world datasets - Traffic Volume and Air Quality - the TCN-BiLSTM model achieved the best overall performance. For Traffic Volume, the TCN-BiLSTM model achieved an R^2 score of 0.976, and for Air Quality, it reached an R^2 score of 0.94. These results highlight the model's effectiveness in leveraging the strengths of Temporal Convolutional Networks (TCNs) for capturing multi-scale temporal patterns and Bidirectional Long Short-Term Memory (BiLSTMs) for retaining contextual information, thereby enhancing the accuracy of time series forecasting.

Keywords Time Series · Deep Learning · Temporal Convolutional Network · Convolutional Neural Network · Long Short-Term Memory · Gated Recurrent Network

A. Mohammed: contributed equally to this work.

✉ Amal Mahmoud
amal.mahmoud@pg.cu.edu.eg

Ammar Mohammed
ammar@cu.edu.eg

[1] Department of Computer Science, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza, Egypt

Published online: 23 August 2024

1 Introduction

Time series refers to a set of data points that are collected at regular time intervals over periods of time. Specifically, it is a sequence of observations recorded in successive time points that may either be continuous or discrete. Time series can be found across many disciplines, including meteorology [1, 2], econometrics [3, 4], energy consumption [5, 6], retail sales [7, 8], healthcare [9, 10], transportation [11, 12], and marketing [13, 14].
Time series can be classified into the following categories: univariate and multivariate. Univariate time series involves the analysis of one single variable across multiple units of time (e.g., daily stock prices). On the other hand, multivariate time series deals with various variables across different periods (e.g., temperature and air pressure over multiple locations) [15, 16]. Multivariate time series may be difficult to analyse due to the curse of dimensionality and the difficulty in capturing the relationships among the data's different features [17]. As a result of these complex characteristics, accurate forecasting of multivariate time series is very challenging [18]. Several methods for time series forecasting have been proposed [19–21], including traditional statistical methods and deep learning models, have been proposed [22]. Traditional statistical methods such as linear Auto-Regressive Integrated Moving Average (ARIMA) [23] and Vector Auto-Regression (VAR) [24] have been widely used for time series prediction, however, their performance is limited when dealing with high-dimensional and non-linear data [25, 26]. However, in many real-world time series problems, the relationships between the variables are non-linear, and the temporal dependencies can change over time. Deep learning models, with their ability to automatically learn hierarchical representations and capture complex patterns, have shown great potential in improving time series forecasting accuracy [27–29]. Models like Recurrent Neural Networks (RNNs) [30, 31], Convolutional Neural Networks (CNNs) [32, 33], and Temporal Convolutional Networks (TCNs) have been widely applied in this context [34, 35].
While deep learning offers significant potential for multivariate time series forecasting, traditional architectures have inherent limitations that hinder their effectiveness. Notably, recurrent neural networks (RNNs), despite their strength in sequence modeling, are susceptible to vanishing or exploding gradients during training [15]. This particularly affects LSTMs, which can struggle to effectively capture the intricate interdependencies between multiple time series variables, a critical element for accurate forecasting [36]. Furthermore, LSTMs, when applied to high-dimensional multivariate data, encounter difficulties in learning long-term dependencies. This is partly due to their reliance on large-weight matrices, which increase the risk of overfitting and limit their ability to capture distant temporal relationships [37, 38]. Similarly, convolutional neural networks (CNNs), with their constrained kernel sizes, struggle to grasp patterns across extended time lags, hindering their performance in longer-term forecasts [39, 40].

Capturing spatiotemporal dynamics is indeed one of the fundamental challenges in time series forecasting research. From a theoretical perspective, real-world time series data often exhibits complex interdependencies that unfold across both spatial and temporal dimensions [41]. Whether modeling energy consumption patterns across interconnected grids [42], transportation flows within urban infrastructure networks [43], or even ecological fluctuations between interlinked habitats - there are nonlocal spatial correlations that evolve dynamically over time. Traditional univariate and multivariate forecasting methodologies struggle to adequately account for such spatiotemporal intricacies [44].

In this paper, we address all these limitations by employing a hybrid deep learning model that fuses the spatial processing capabilities of Convolutional Neural Networks (CNNs) with

the temporal modeling strengths of Recurrent Neural Networks (RNNs), creating a unified architecture that includes CNN_LSTM, CNN_BILSTM, and CNN_GRU configurations. Additionally, we enhance the model's temporal reach by integrating Temporal Convolutional Networks (TCNs) with RNNs, resulting in hybrid TCN_RNN models such as TCN_LSTM, TCN_BILSTM, and TCN_GRU. These models are designed to effectively capture and analyze the spatiotemporal characteristics of multivariate time series data.

We will compare the effectiveness of our proposed hybrid model with leading deep learning models, including LSTM, GRU, and BiLSTM networks. This comparison aims to demonstrate the superiority of hybrid models in overcoming the challenges associated with spatiotemporal data in multivariate time series forecasting.

The structure of this paper is as follows: Sect. 2 presents an overview of existing research relevant to our work. Section 3 provides a comprehensive background of deep learning architectures, essential for understanding our proposed method. Section 4 presents our proposed method, detailing the architecture and methodology employed.Sect. 5 describes the methodology employed in our study, including the dataset used, experimental design, and evaluation metrics. Section 6 presents the results obtained from our experiments, demonstrating the effectiveness of our proposed method. Finally, Sect. 7 concludes the paper.

2 Related Work

The field of time series forecasting has witnessed significant advancements, with researchers exploring the application of deep learning techniques to overcome the limitations of traditional statistical and machine learning approaches. This section provides an overview of the related work conducted in this area, discussing the advancements achieved and proposed solutions.

Statistical Models for Multivariate Time Series Statistical models, such as Vector Auto-Regressive (VAR) [45, 46] and multivariate Exponential Smoothing (ES) [47, 48], have long been used for time series analysis. These models excel in capturing linear interdependencies among multiple time series variables. However, they are limited in their ability to model non-linear relationships and can suffer from overparameterization when dealing with high-dimensional data [49–51].

Traditional Machine Learning Models Traditional machine learning models like Support Vector Machines (SVM) [52, 53] and k-Nearest Neighbors (kNN) have been adapted for time series forecasting [54]. These models provide non-linear modeling capabilities but often require extensive feature engineering and do not inherently capture sequential dependencies, which can be particularly limiting for complex time series data [55, 56].

Traditional Deep Learning Models Deep learning models such as RNNs [11], LSTMs [57], and GRUs [58] have addressed some of the limitations of earlier statistical and machine learning approaches by effectively capturing long-term dependencies in sequential data. However, they can be computationally expensive, difficult to train, and may still encounter vanishing gradient issues in very long sequences [44].

3 Background of Deep Learning Architectures

This section presents a brief overview of the theory behind the selected deep learning models.

3.1 Long Short-Term Memory (LSTM)

LSTM is introduced to resolve the problems caused by vanishing and exploding gradients in RNNs [59]. It is recommended for long-term dependency relationships identified in network traffic since chained memory blocks are used as short-term memory to remember previous actions taken in time steps. Each memory block contains a memory cell as well as three gates: an input gate (i_t), an output gate (o_t), and a forget gate (f_t).

First, the memory cell executes the forget gate (f_t) to discard the information that was unnecessary in the previous state (C_{t-1}), as shown in equation(1). This step aims to ensure that the model is effective and scalable. Based on the obtained information, the input gate (i_t) contains the values that should be updated, as shown in equation (2). Additionally, the activation function generates a vector of new candidate values ($\tilde{C}_t$). To update the state of the cell, both the input gate value (i_t) and the generated vector value are multiplied together. In the next step, the value of the forget gate (f_t) is multiplied by the value of the previous cell's state (C_{t-1}) to obtain the updated value for the old cell state, as shown in equation (3). Finally, the output gate (o_t) is derived from the current cell state (C_t), as shown in equation (5).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{4}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$
$$h_t = o_t \cdot \tanh(C_t) \tag{6}$$

Here, h_{t-1} represents the output from the previous time step. The weight matrices for the input, output, and forget gates are denoted by W_i, W_o, and W_f, respectively. The sigmoid activation function is represented by σ, and tanh denotes the hyperbolic tangent function. The bias terms for the input, output, and forget gates are b_i, b_o, and b_f. The input at time t is x_t. Fig. 1 shows the architecture of the LSTM network.

3.2 Gated Recurrent Unit (GRU)

The GRU is considered less complex than the LSTM, which is its most significant advantage as it requires less training time due to its simplified architecture [61]. The GRU consists of two main components: the update gate (u_t) and the reset gate (r_t). The update gate determines which information is necessary to carry forward to the next stage (equation(7)), while the reset gate decides how much of the past information to forget as (equation(8)). This process is closely related to the current input and the previous hidden state.

$$u_t = \sigma(w_u \cdot h_{t-1}, x_t) \tag{7}$$
$$r_t = \sigma(w_r \cdot h_{t-1}, x_t) \tag{8}$$
$$h_t = (1 - u_t) \cdot h_{t-1} + u_t \cdot \tanh(w_r \cdot h_{t-1}, x_t) \tag{9}$$

where w_u and w_r are the weights for the update and reset gates, respectively. Figure 2 shows the architecture of the GRU network.

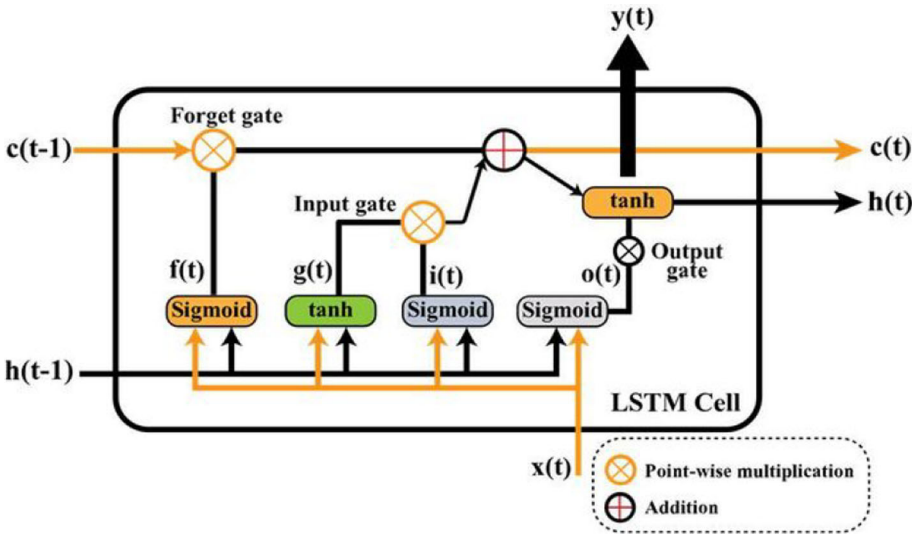

Fig. 1 Architecture of LSTM network [60]

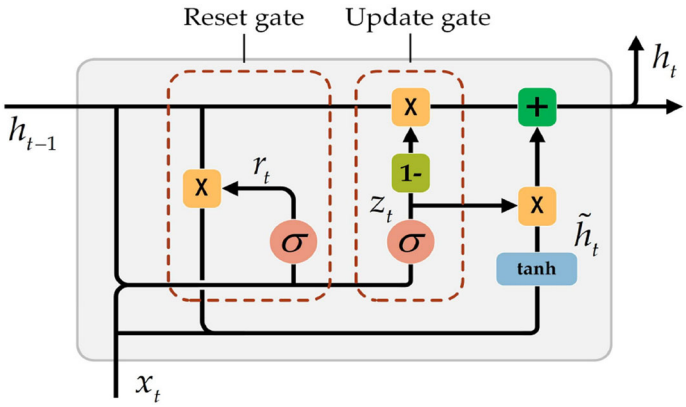

Fig. 2 Architecture of GRU network [62]

3.3 Bidirectional Long Short-term Memory (BiLSTM)

Bidirectional LSTMs (BiLSTMs) are a useful variant of the standard LSTM architecture that is well-suited for modeling time series data. Unlike regular LSTMs which only can use past context when processing sequential input, Bi-LSTMs contain two separate LSTM layers - one layer that processes the input sequence forwards in time and another layer that processes the sequence backwards in time [63].

This bidirectional design allows the BiLSTM to have access to both past and future context at each time step, as the forward layer can look at future time steps while the backward layer observes past time steps. The outputs from the two layers are then concatenated and fed into the next part of the network. This additional contextual information provided by processing sequences in both directions has been shown to improve Bi-LSTMs performance on tasks involving sequential data like time series forecasting and natural language processing when

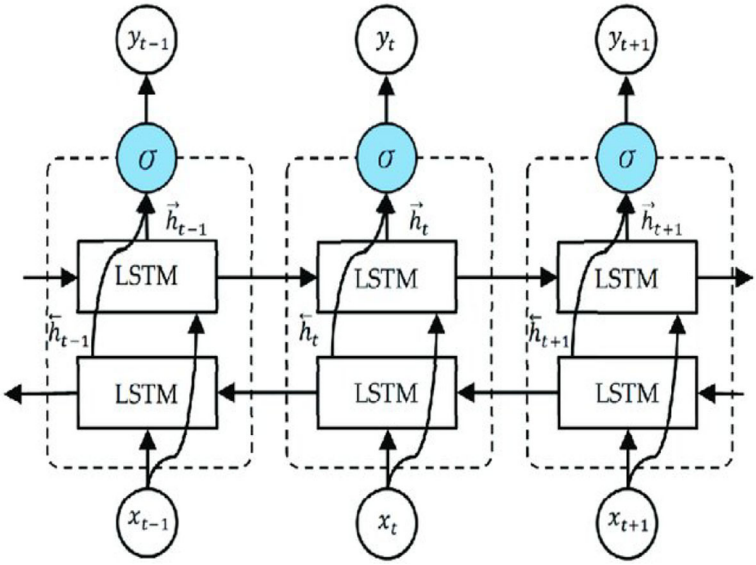

Fig. 3 Architecture of Bi-LSTM network [65]

compared to standard LSTMs [64]. However, a potential limitation is that Bi-LSTMs require more parameters than regular LSTMs due to the separate forward and backward layers, which can increase the risk of overfitting on smaller datasets and require more computational resources to train. Figure 3. illustrates the bidirectional LSTM.

3.4 Convolutional Neural Network (CNN)

CNNs were originally developed for computer vision tasks to process grid-like data such as images [66]. The key aspects of a CNN include convolutional layers and pooling layers. Convolutional layers apply linear convolution operations between the input and learned filters (typically 3D arrays of weights) to extract local spatial features from the data [67]. The filters slide across the width and height of the input volume to generate a feature map at each position.

Pooling layers perform downsampling operations after convolutional layers to reduce the spatial size of the data and control overfitting [68]. Common pooling methods are max pooling, which outputs the maximum value from the region, and average pooling [5]. CNNs stack multiple convolutional and pooling layers to learn increasingly abstract features. The learned filters in a layer are adapted in subsequent layers to best model the input data.

The basic mathematical formulation of a 1D convolution operation is equation(10):

$$y[i] = (x * w)[i] = \sum_{k=-\infty}^{\infty} x[k] \cdot w[i-k] \tag{10}$$

Where x is the input, w is the filter, $*$ denotes the convolution operation, and y is the output feature map. For multivariate time series with n variables at each time step, the input x would be a tensor of shape (number of samples, n, time steps). For time series forecasting, 1D convolutions along the temporal dimension allow CNNs to learn features

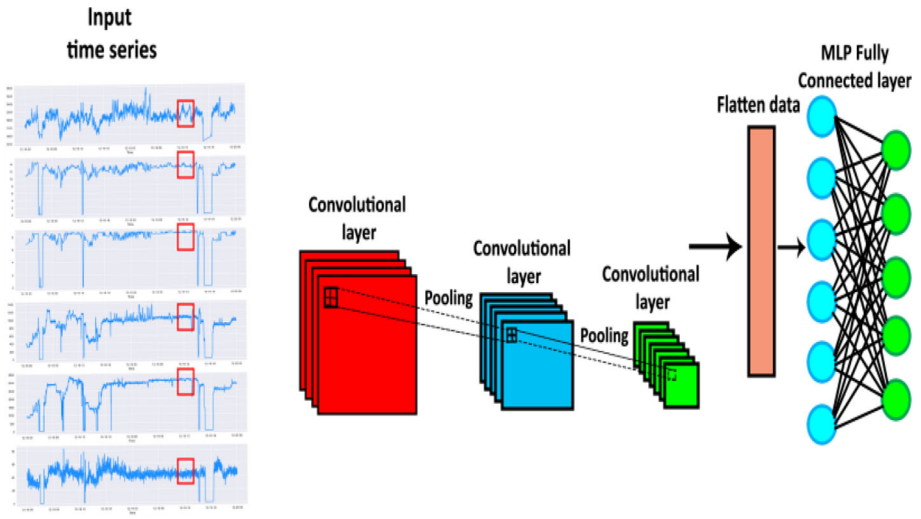

Fig. 4 Architecture of CNN [44]

directly from raw data without manual feature engineering [?]. CNNs have achieved state-of-the-art performance across domains by leveraging these properties. Figure 4. illustrates the bidirectional LSTM.

3.5 Temporal Convolutional Network(TCN)

Temporal Convolutional Networks (TCNs) are a type of neural network that is particularly effective for time series prediction tasks. Unlike traditional recurrent neural networks (RNNs) like LSTMs and GRUs, TCNs use dilated convolutions to capture long-term dependencies in time series data [64].

The architecture of TCNs consists of multiple stacked convolutional layers. Each layer applies a dilated convolution operation, which allows the model to consider information from a wider range of previous time steps. The dilation rate increases exponentially with each layer, enabling the model to capture long-range dependencies. Accordingly, a 1D dilated convolutional operation on an element of a sequence can be defined as equation(11):

$$f(s) = (x * df)(s) = \sum_{i=0}^{k-1} f_(i) \cdot x_{s-d \cdot i} \tag{11}$$

Where $f : \{0, \ldots, k-1\} \rightarrow \mathbb{R}$ is the convolution kernel, k is the kernel size, d is the dilation factor, and $s - d \cdot i$ represents the data of the past. d increases as the network gets deeper.

TCNs also employ residual connections that directly pass values from input to output of convolutional blocks, enabling very deep networks for capturing longer-term patterns [69]. Skip connections further allow information to skip over blocks to preserve temporal resolution

In a TCN, the output at time (t) is computed as a function of the input at time (t) and the outputs at previous time steps. This property ensures that the model's predictions at a given time step are only influenced by past data, adhering to the causality principle typical of time series data. Figure 5. illustrates the bidirectional LSTM.

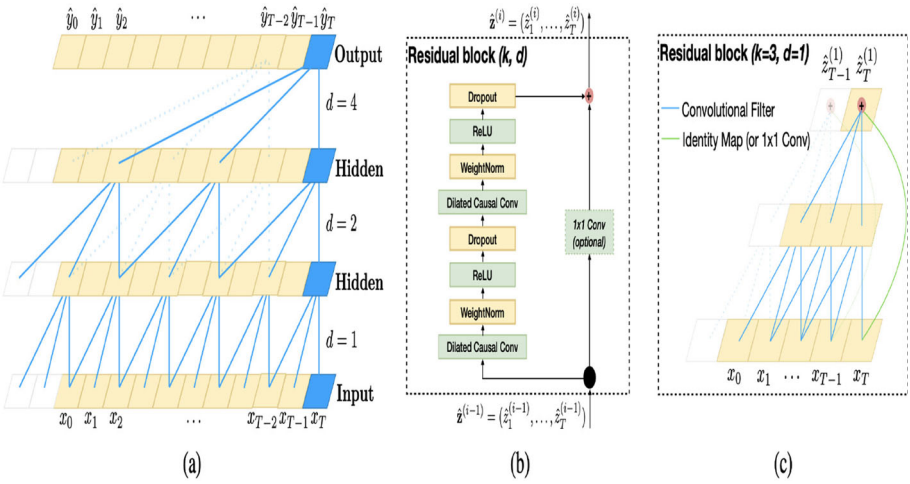

Fig. 5 Temporal Convolutional Network (TCN) architecture [64]

4 Proposed Method

In this section, we present a proposal for hybrid TCN_RNNs and CNN_RNNs models, specifically CNN_LSTM,CNN_BiLSTM,CNN_GRU, TCN_LSTM, TCN_BiLSTM, and TCN_GRU.

4.1 Hybrid CNN_RNNs(LSTM,GRU or BiLSTM) Models

Our research proposes using hybrid CNN-RNN models, specifically CNN-LSTM, CNN-BiLSTM, and CNN-GRU, to predict multivariate time series. These models combine the strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to handle the high-dimensional and spatiotemporal aspects of multivariate time series data.

The architecture of these models typically consists of three main components: the convolutional layers, the recurrent layers, and the output layer. The convolutional layers extract spatial features from the input data, while the recurrent layers capture the temporal dependencies.

In the case of CNN-LSTM, the input data is first fed into the CNN layers, which perform convolutional operations to extract relevant spatial features. The output of the CNN layers is then passed to LSTM, which enables the model to capture long-term dependencies in the temporal dimension. The LSTM layers process the sequential information and generate hidden states passed through time. Finally, the output layer, typically composed of fully connected layers, produces the predicted values for the multivariate time series. Figure 6 illustrates a novel deep learning architecture that combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks has been proposed for advanced time series analysis and forecasting. Additionally, substituting an LSTM layer with a GRU or BiLSTM layer introduces a unique set of benefits.

Similarly, CNN-BiLSTM incorporates bidirectional LSTM, which processes the input data in both forward and backward directions. This allows the model to capture temporal dependencies from past to future and from future to past, enhancing its ability to learn

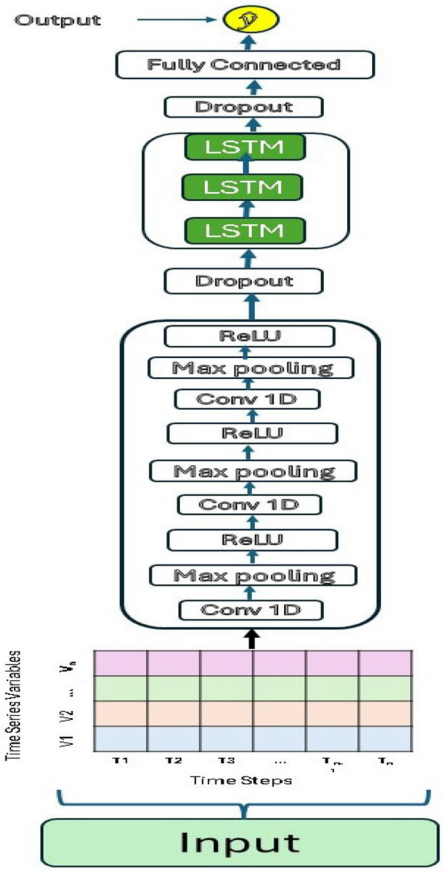

Fig. 6 Architecture of hybrid CNN-LSTM for time series forecasting

complex patterns in the data. On the other hand, CNN-GRU models replace the LSTM with gated recurrent units (GRUs). GRUs have a simplified architecture compared to LSTMs, making them computationally more efficient while still capturing temporal dependencies effectively.

In several ways, these hybrid models address the challenges of high dimensionality and spatiotemporal dependencies in multivariate time series prediction. By leveraging the convolutional layers, they can automatically extract relevant spatial features from high-dimensional input data, reducing the dimensionality and focusing on the most informative aspects. Secondly, the recurrent layers, such as LSTM and GRU, enable the models to capture the temporal dependencies and patterns in the data across different time steps. Lastly, integrating both CNN and RNN components within the hybrid architecture ensures that the model can effectively capture spatial and temporal correlations, leading to improved prediction performance.

4.2 Hybrid TCN_RNNs(LSTM,GRU or BiLSTM) Models

In our research paper, we introduce hybrid Temporal Convolutional Network (TCN)-RNN models, such as TCN-LSTM, TCN-BiLSTM, and TCN-GRU, to tackle multivariate time series forecasting. These models address the challenges of high dimensionality, spatiotempo-

ral dependencies, and other complexities inherent in multivariate time series. The architecture of a hybrid TCN-RNN model combines the parallel computation benefits of TCNs with the sequential data processing capabilities of RNNs. TCNs use causal convolutions, ensuring that predictions at a given time step are influenced only by past data, preserving the temporal order of events. This is achieved through dilated convolutions, which expand the receptive field exponentially without increasing the parameters.

On top of the TCN architecture, we integrate RNNs, such as LSTM, BiLSTM, and GRU, to capture both forward and backward temporal dependencies, further enriching the model's predictive capacity. The LSTM, for instance, includes gates that regulate the flow of information. In Fig. 7, we can observe the overall structure of the hybrid TCN-LSTM architecture. The input time series data is passed through the TCN layers responsible for capturing local patterns and conducting temporal convolutions. Similarly, replacing the LSTM layer with a GRU or BiLSTM layer offers a different set of advantages.

These hybrid TCN-RNN models effectively manage the high-dimensional nature of multivariate time series by capturing relevant features through the TCN layers and modeling temporal sequences via RNNs. The spatiotemporal component is addressed by the TCN's ability to process multiple time steps simultaneously and by the RNN's proficiency in capturing temporal sequences, making the models suitable for forecasting tasks where both spatial and temporal factors are critical.

5 Experiments

In the Experiments section, we describe the dataset used for evaluation, the experimental setting, and the evaluation metric employed to assess the performance of our proposed method.

5.1 Dataset description

In our study, we employ two distinct datasets: one focusing on air quality and the other on traffic volume control. The air quality dataset,[1] which spans from March 2004 to February 2005, provides historical data on various air pollutants, including carbon monoxide (CO), nitrogen dioxide (NO2), ozone (O3), and others. This dataset is derived from hourly averages collected by a set of five metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device, positioned at road level in a significantly polluted area within an Italian city. On the other hand, the traffic volume control dataset is designed to analyze and manage traffic flow, offering insights into the dynamics of vehicular movement and congestion patterns. By leveraging these datasets, we aim to develop and evaluate models that can effectively predict and control both air quality and traffic volume, contributing to environmental sustainability and efficient urban planning.

The traffic volume dataset[2] contains hourly Interstate 94 westbound traffic volumes for (MN DoT ATR) station -301 from 2012 to 2018, roughly halfway between Minneapolis and St Paul, Minnesota. The traffic volume is affected by hourly weather conditions and holidays. A detailed description of the datasets is represented in Table 1. Traffic volume is our target output to predict the traffic flow, and other data are treated as the model's input. These columns dropped because min, Q1, Q2, and Q3 were all zero for rain and snow features.

[1] https://archive.ics.uci.edu/dataset/360/air+quality

[2] https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume

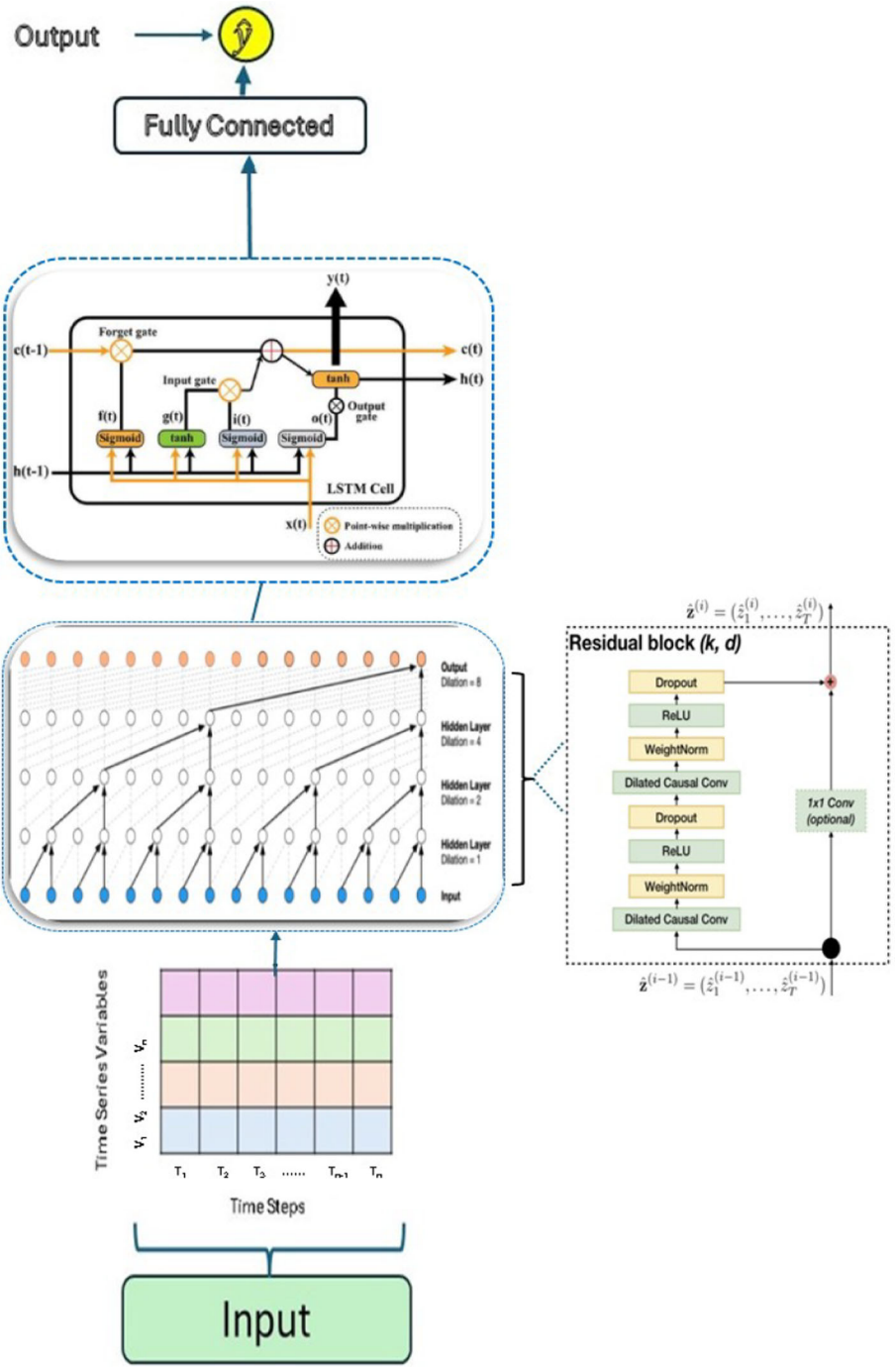

Fig. 7 Architecture of hybrid TCN-LSTM for time series forecasting

Table 1 Dataset description

Dataset	Features	No. of rows	Frequency
Traffic volume	Holiday, rain, snow, temperature, clouds, weather_main, date_time, traffic_volume, weather_description	48204	Hourly
Air quality	CO(GT)-concentration CO in mg/m^3, snow, PT08.S1(CO)sensor response (nominally CO targeted), NMHC(GT)Non Metanic HydroCarbons concentration in microg/m^3, C6H6(GT) Benzene concentration in microg/m^3, NOx(GT)NOx concentration in ppb, PT08.S2(NMHC)sensor response, PT08.S3(NOx) sensor response (nominally NOx targeted), NO2(GT)NO2 concentration in microg/m^3	9358	Hourly

5.2 Data preprocessing

Data preprocessing plays a vital role in the overall effectiveness of deep learning algorithms. In this experiment, specific steps have been taken to preprocess the data. Firstly, missing values are addressed by replacing them with the mean value before and after the missing value (known as the near mean approach). Outliers, which can disrupt the learning process, are also identified and treated as missing values. When dealing with categorical data, deep learning algorithms face challenges in effectively representing them. To overcome this, a technique called "one-hot encoding" is employed. This method transforms categorical data with n possible values into n indicator features, with only one active feature at a given time. In this case, "one-hot encoding" is used for the categorical features of holiday, weather_main, and weather_description. We perform normalization to ensure a fair comparison and appropriate weighting of variables with different scales. This involves scaling the data to values between 0 and 1 [70]. Deep learning networks are susceptible to scaling, and the min-max scaler suits these networks. The min-max scaler is shown in equation (12):

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{12}$$

Where x represents the dataset, and x_{scaled}represents the normalized dataset.

To forecast traffic flow in the traffic volume dataset and true hourly averaged concentration CO in mg/m^3" in Air Quality for the next h time steps, a sliding window approach is employed to transform the input time series data into input–output pairs. This is achieved by considering a size window w, where x_t represents the time-series data, h denotes the forecasting horizon, and f indicates the deep learning model established through training. The input–output pairs can be represented as equation (13):

$$[x_{t+1} + x_{t+2}, \ldots, x_{t+h}] = f(x_t, x_{t-1}, \ldots, x_{t-w}) \tag{13}$$

5.3 Experimental Setting

This study aims to implement a hybrid deep learning model for multivariate time series forecasting. For each model, 80% of the training data is selected for training, and 20% is selected for validation. Fig. 8 illustrates the process of reshaping and splitting the dataset for further analysis and model training.

The selection of hyperparameters plays a critical role in the performance of any deep learning algorithm. After fine-tuning our forecasting model by using grid search to obtain the best-performing model across the whole dataset [71], we conducted a manual grid search over a series of combinations to select the best hyper-parameters. We use the Adam optimization algorithm to optimize the model parameters [72], which can adapt the learning rate. Table 2 outlines the hyperparameter search space for the deep learning models employed in this research, detailing the range of parameters explored to optimize model performance.

Table 3 delineates the optimal hyperparameters for the hybrid deep learning models applied to our datasets, showcasing the configuration that yielded the best performance.

We propose Multi-Layer, Convolutional, and Recurrent Networks as basic building blocks and then combine them into heterogeneous architectures with different variants, trained with optimization strategies like drop_out = 0.2 and skip connections, early stopping, adaptive learning rates, filters, and kernels of various sizes, between others. This study uses the ReLU since it most effectively forecasts noisy non-stationary time series [73]. In addition, the ReLU reduces training time, simplifying the model.

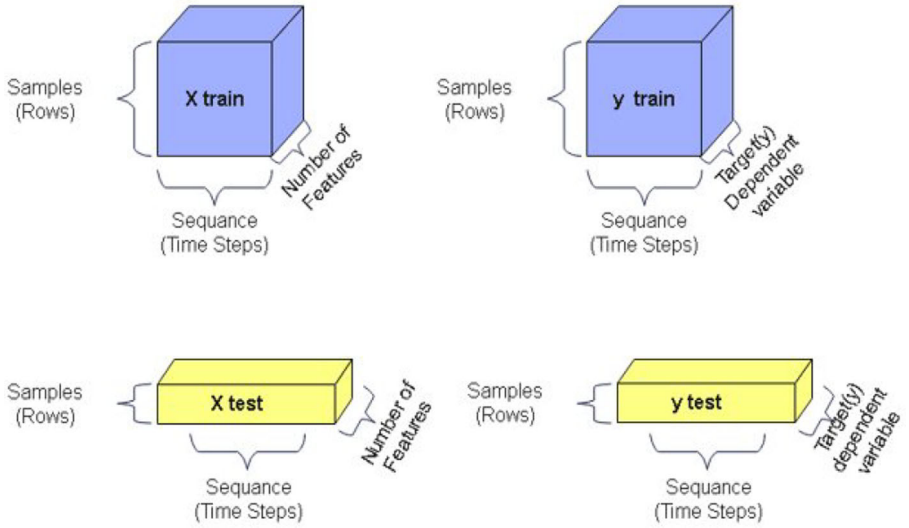

Fig. 8 Data Reshape And Split

Table 2 Hyperparameter search space for deep learning models (grid search)

Model	Hyperparameters
LSTM, BiLSTM, GRU	Number of hidden units: [32, 64, 128, 256]
	Dropout rate: [0.1, 0.2, 0.3, 0.4]
	Number of layers: [1, 2, 3]
	Learning rate: [0.001, 0.0001, 0.005,0.0005]
	Number of Epochs:[32,64,128,256]
	Number of Batch:[32,64,128,256]
CNN	Number of Filters: [32, 64, 128, 256]
	Kernel Size: [3, 5, 7, 9]
	Pooling Size: [2, 3, 4, 5]
	Dropout Rate: [0.1, 0.2, 0.3, 0.4]
	Number of Layers: [1, 2, 3]
	Learning Rate: [0.001, 0.0001, 0.005, 0.0005]
	Number of Epochs:[32,64,128,256]
	Number of Batch:[32,64,128,256]
TCN	Number of filters: [32, 64, 128, 256]
	Kernel size: [2, 4, 8]
	Number of stacks: [2, 4, 6]
	Dilations: [[1, 2, 4], [2, 4, 8], [4, 8, 16]]
	Dropout rate: [0.1, 0.2, 0.3, 0.4]
	Learning rate: [0.001, 0.0001, 0.005,0.0005]
	Number of Epochs:[32,64,128,256]
	Number of Batch:[32,64,128,256]

Table 3 Hyperparameter selections using the grid search

Model	Configuration	Training Parameters
LSTM Bi-LSTM GRU	3 Layers(128 units)	Epochs: 64
	2 layers,64 units for air quality	Batch_size: 64
CNN	Layer: 3 1D Conv Layers, 2 for Air Quality	Epochs: 256, Batch_size: 512
	Filter: 16, Kernel size: 4	
	Filter:8 Air quality Padding: same, MaxPooling1D(pool_size=2)	
TCN	Filter: 8, Kernel size: 4	Epochs: 64, Batch_size: 64
	Stacks: 4	
	Dilations: (2, 4, 8, 16)	
	Padding: causal	
CNN-LSTM	3 1DConv Layers, 3 RNNs Layers (256 units),2Layers(128 Units)Air quality	Epochs: 200, Batch_size: 256
CNN-BiLSTM	Filter: 64, Kernel size: 2	
CNN-GRU	Padding: same, MaxPooling(pool_size=2)	
TCN-LSTM	3 Layers- 2 Layers for Air Quality, 128 units	Epochs: 64, Batch_size: 64
TCN-BiLSTM	Filter: 8, Kernel size: 4	
	Padding: causal, Stacks: 4	
TCN-GRU	Dilations: (2, 4, 8, 16),(2,4,8) for Air quality	

5.3.1 Baseline Models

Multilayer Perceptron (MLP): MLP is a type of artificial neural network that consists of multiple layers of interconnected nodes. It is commonly used for regression and classification tasks, including time-series forecasting. MLP learns complex nonlinear relationships between input and output variables by adjusting the weights and biases of the network during training. MLP's ability to capture intricate patterns and dependencies in the data [74].
Support Vector Regression (SVR): SVR is a machine learning algorithm that extends the concept of support vector machines (SVM) to regression problems. It aims to find a function that best fits the training data while minimizing the error and maximizing the margin between the data points. SVR is particularly effective when dealing with nonlinear relationships in time-series data [75].
Vector Autoregression (VAR): VAR employs statistical modeling to examine and forecast the interconnectedness among multiple time series variables. VAR models leverage the lagged values of each variable as predictors to anticipate their future states [76].
Autoregressive Integrated Moving Average (ARIMA): The ARIMA model is a fundamental statistical tool for time-series analysis that combines autoregressive, differencing, and moving average components. It is widely utilized in time-series forecasting to capture and model the underlying patterns and trends within the data. ARIMA is particularly effective when dealing with stationary time series data, where the mean and variance remain constant over time [77].
Seasonal Autoregressive Integrated Moving Average with eXogenous variables (SARIMAX): SARIMAX builds upon the ARIMA model by integrating seasonal components and external variables. It is particularly useful for time-series forecasting when the data exhibits seasonal patterns and incorporates additional information (exogenous variables) to enhance prediction accuracy [78].
Table 4 presents a structured overview of the hyperparameters for the baseline model, including their names, descriptions, and default values where applicable.

5.4 Evaluation Metric

The evaluation of a model's performance is a critical aspect of assessing its effectiveness. Several commonly employed evaluation metrics in the literature for time series models include the Mean Absolute Error (MAE), Mean-Squared Error (MSE), and R-squared (R^2) [79].
Mean Squared Error (MSE): MSE measures the average squared difference between the predicted values and the actual target values. Equation(14) Define MSE as:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{14}$$

where n is the number of data points, y_i is the actual target value, and $\hat{y}_i$ is the predicted value.
Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual target values.Equation(15) define the MAE as:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{15}$$

R-squared (R^2): R^2 measures the proportion of the variance in the target variable that is predictable from the predictors. R^2 ranges from 0 to 1, with higher values indicating a better

Table 4 Hyperparameters of Baseline Models

Model	Hyperparameters
MLP	Number of hidden layers:3
	Number of neurons per hidden layer:128,64,32
	Activation function:Relu
	Epochs:150
	Batch_size:32
SVR	Kernel type (RBF)
	Regularization parameter (C=1.0)
	Kernel coefficient (gamma= scale)
VAR	Lag order (p)=2
ARIMA	Order of AR component (p)=1
	Order of I component (d)=1
	Order of MA component (q)=1
SARIMAX	Order of seasonal AR component (P)=1
	Order of seasonal I component (D)=1
	Order of seasonal MA component (Q)=1
	Order of non-seasonal AR component (p)=1
	Order of non-seasonal I component (d)=1
	Order of non-seasonal MA component (q)=24

fit of the model to the data. Equation(16) defines the R^2 as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} \left(y_t^i - \hat{y}t^i\right)^2}{\sum_{i=1}^{N} \left(y_t^i - \bar{y}_t^i\right)^2} \tag{16}$$

Mean Absolute Percentage Error (MAPE): MAPE measures the average percentage error between the predicted values and the actual target values. Equation(17) defines the MAPE as:

$$MAPE = \frac{1}{N} \sum_{i=1}^{N} \frac{\left|y_t^i - \hat{y}t^i\right|}{\left|yt^i\right|} \tag{17}$$

6 Results and Discussion

In this study, we comprehensively evaluated various deep learning models for time series forecasting on two datasets: Traffic Volume and Air Quality. The models were assessed using several evaluation metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R-squared (R^2).

Table 5 presents the evaluation results for the different models, categorized into baseline models, state-of-the-art neural networks, and our proposed models.

The baseline models, including Multilayer Perceptron (MLP),Support Vector Regression (SVR),Vector Autoregression (VAR), Autoregressive Integrated Moving Average (ARIMA), and Seasonal Autoregressive Integrated Moving Average (SARIMAX), showed relatively

Table 5 Results summary of all methods with two datasets (24 time steps)

Model	Traffic volume					Air quality				
	MSE	RMSE	MAE	MAPE	R^2	MSE	RMSE	MAE	MAPE	R^2
Baseline models										
MLP	0.0125	0.0751	0.111	194	0.80	0.0059	0.065	0.076	181	0.74
SVR	38.4	8	9.02	inf	0.32	24.5	4.9	4.02	inf	0.30
VAR	3892413	1973	1725	238	−0.0002	1.9	1.4	1.1	150	−0.04
ARIMA	3892415	1972	1725	nan	−0.0003	3.3	1.8	1.3	67	−0.7
SARIMAX	3893371	1973	1726	nan	−0.0005	1.5	1.2	0.9	113	0.2
State of art models										
LSTM	0.0042	0.063	0.045	185	0.920	0.0019	0.0441	0.0303	36	0.89
GRU	0.0045	0.067	0.040	186	0.935	0.0021	0.0456	0.0308	35	0.90
BiLSTM	0.0032	0.057	0.39	185	0.932	0.0023	0.0479	0.0312	33	0.91
TCN	0.0023	0.047	0.36	188	**0.950**	0.0018	0.0424	0.0299	31	**0.93**
Proposed models										
CNNLSTM	0.0023	0.048	0.033	184	0.933	**0.0012**	0.045	0.034	34	0.91
CNNBiLSTM	**0.0020**	**0.045**	**0.030**	189	**0.960**	0.0018	.0432	0.030	32	0.92
CNNGRU	0.0025	0.05	0.031	180	0.948	0.0025	0.036	0.050	58	0.90
TCNLSTM	0.0020	0.044	0.025	179	0.951	0.0019	0.034	0.55	32	0.92
TCNBiLSTM	0.0018	0.042	0.022	178	**0.976**	**0.0017**	0.422	**0.027**	**29**	**0.94**
TCNGRU	0.0021	0.046	0.024	176	0.972	0.0022	0.456	0.032	34	0.91

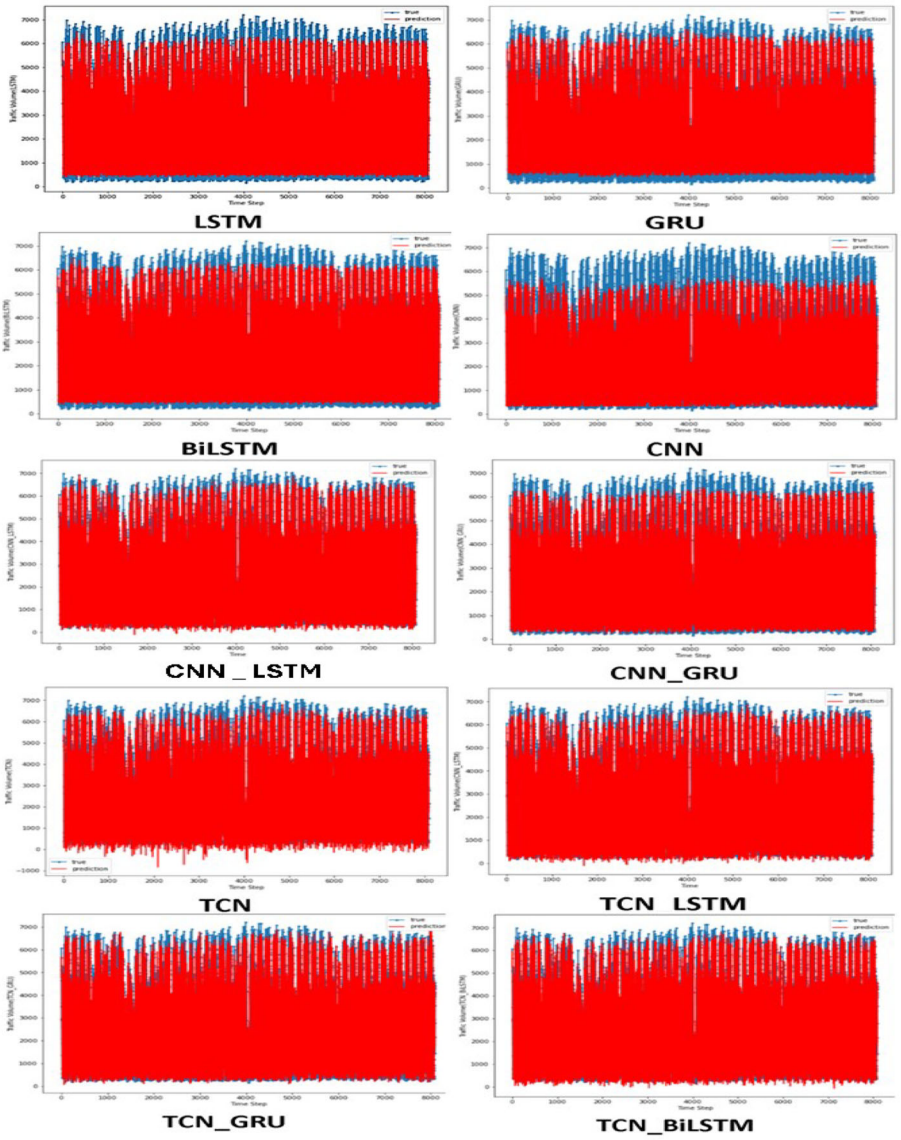

Fig. 9 Comparison of forecasting values and actual values for traffic flow (Timesteps =24)

poor performance on both datasets. For Traffic Volume, they had high MSE, RMSE, MAE, and MAPE values, and negative R^2 values, indicating that they did not fit the data well. Similarly, for Air Quality, they had high MSE, RMSE, MAE, and MAPE values, and low R^2 values, suggesting that they struggled to capture the underlying patterns in the data. Among the baseline models, SARIMAX achieved the best performance for the Air Quality dataset, with an MSE of 1.5, RMSE of 1.2, MAE of 0.9, MAPE of 113, and an R^2 of 0.2.In comparison, the MLP and SVR models demonstrated better performance on both datasets. MLP achieved an MSE of 0.0125, RMSE of 0.0751, MAE of 0.111, and an R2 of 0.80 for

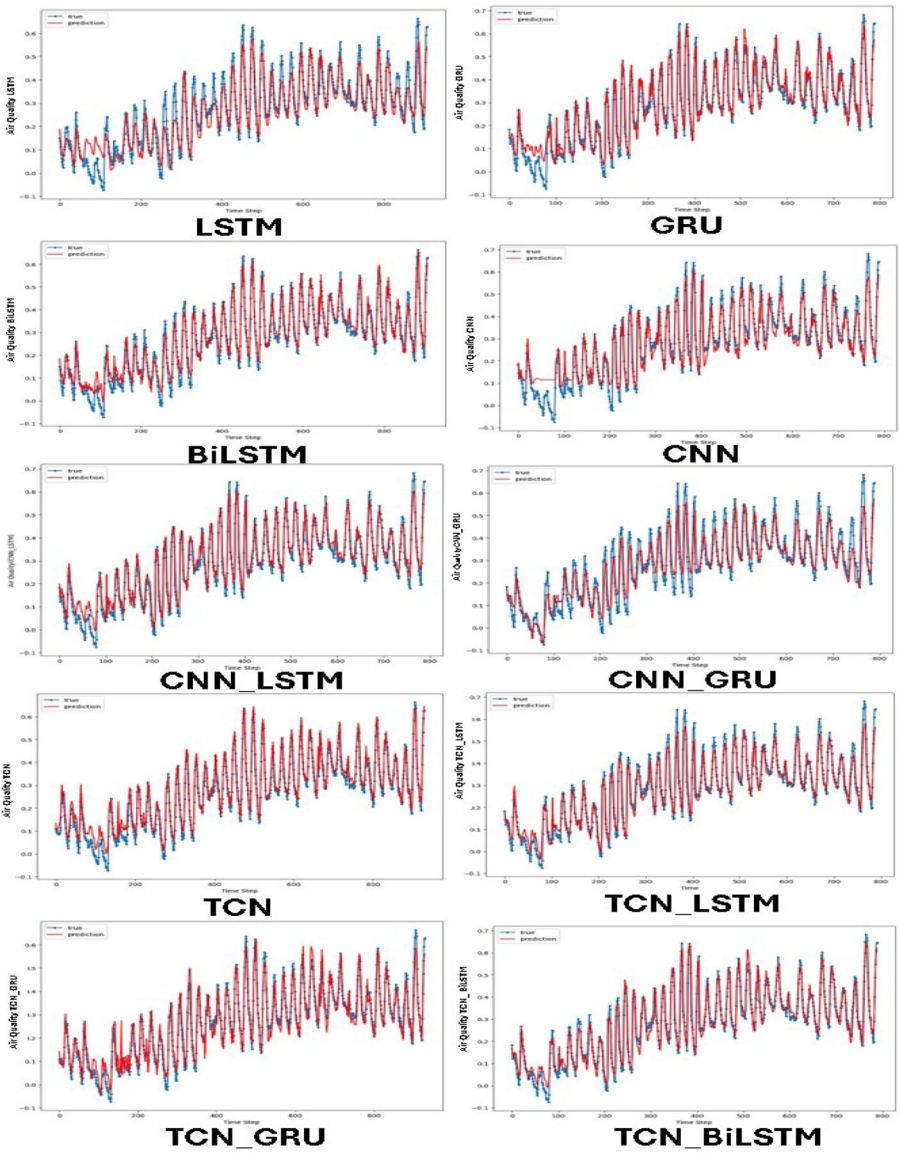

Fig. 10 Comparison of forecasting values and actual values for Air Quality (Timesteps =24)

Traffic Volume, and an MSE of 0.0059, RMSE of 0.065, MAE of 0.076, and an R2 of 0.74 for Air Quality. SVR, on the other hand, had an MSE of 38.4, RMSE of 8, MAE of 9.02, and R2 of 0.32 for Traffic Volume, and an MSE of 24.5, RMSE of 4.9, MAE of 4.02, and an R2 of 0.30 for Air Quality.

The state-of-the-art models, including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional LSTM (BiLSTM), and Temporal Convolutional Network (TCN), showed significant improvements over the baseline models. For Traffic Volume, they had lower MSE, RMSE, MAE, and MAPE values, and higher R^2 values, indicating better per-

formance. For Air Quality, they also showed improvements in all metrics.TCN demonstrated the highest performance for both datasets. For the Traffic Volume dataset, TCN achieved an MSE of 0.0023, RMSE of 0.047, MAE of 0.36, MAPE of 188, and an R^2 of 0.950. Similarly, for the Air Quality dataset, TCN obtained an MSE of 0.0018, RMSE of 0.0424, MAE of 0.0299, MAPE of 31, and an R^2 of 0.93.

Our proposed models, which combine different architectures, showed notable improvements in forecasting performance. For the Traffic Volume dataset, the CNN-BiLSTM model achieved the lowest MSE (0.0020), RMSE (0.045), and MAE (0.030), along with an R^2 of 0.960. The TCN-BiLSTM model also performed exceptionally well, with an MSE of 0.0018, RMSE of 0.042, MAE of 0.022, MAPE of 178, and the highest R^2 of 0.976.

For the Air Quality dataset, the CNN-LSTM model achieved the lowest MSE of 0.0012, while the TCN-BiLSTM model outperformed all others with an RMSE of 0.422, MAE of 0.027, MAPE of 29, and the highest R^2 of 0.94.

Figs. 9 and 10, show the comparison of forecasting values and actual values for Traffic Flow and Air Quality respectively:

Figure 9: Comparison of forecasting values and actual values for Traffic Flow (Timesteps = 24). The sub-figures show the actual vs. predicted values for each state-of-the-art model (LSTM, GRU, BiLSTM, TCN) and our proposed models (CNN-LSTM, CNN-BiLSTM, CNN-GRU, TCN-LSTM, TCN-BiLSTM, TCN-GRU).

The TCN-BiLSTM model achieves the closest match between actual and predicted values, demonstrating its superior performance for Traffic Flow forecasting. The CNN-BiLSTM and TCN-LSTM models also show good accuracy, while the LSTM, GRU, and BiLSTM models have larger forecasting errors.

Figure 10: Comparison of forecasting values and actual values for Air Quality (Timesteps = 24).

The TCN-BiLSTM model again achieves the closest match between actual and predicted values, demonstrating its effectiveness for Air Quality forecasting. The CNN-BiLSTM and TCN-LSTM models also show good accuracy, while the LSTM, GRU, and BiLSTM models have larger forecasting errors.

Overall, the visual comparisons in Fig. 9 and 10 confirm the quantitative results from Table 5. The TCN-BiLSTM model consistently achieves the closest match between actual and predicted values across both datasets. The CNN-BiLSTM and TCN-LSTM models also show promising performance. These findings provide further validation of the effectiveness of the proposed hybrid CNN-RNN and TCN-RNN models for multivariate time series forecasting. The superior performance of our proposed hybrid models can be attributed to their ability to effectively capture both spatial and temporal dependencies in the multivariate time series data. By combining convolutional and recurrent modules, these architectures can extract relevant features and dynamics, leading to more accurate forecasting results.

7 Conclusion

In this paper, we comprehensively evaluated various deep learning models for multivariate time series forecasting on Traffic Volume and Air Quality datasets. Our proposed hybrid CNN-RNN and TCN-RNN models significantly outperformed both baseline and state-of-the-art models.These results highlight the effectiveness of combining convolutional and recurrent modules for multivariate time series forecasting. The hybrid models were able to capture both spatial and temporal dependencies, leading to more accurate predictions. The TCN-BiLSTM

model achieved the best overall performance, with the lowest error and highest R^2 values for both datasets. The CNN-BiLSTM and TCN-LSTM models also showed promising results. The hybrid models presented in our paper have demonstrated an outstanding ability to address some of the traditional challenges faced by other models in time series forecasting. These challenges include capturing complex non-linear patterns, dealing with noisy data, and accounting for the seasonality and trend components inherent in many time series datasets.
In essence, these hybrid models combine the best of both worlds-the feature extraction capabilities of CNNs and TCNs and the sequential data processing strengths of RNNs. This results in a more robust and accurate forecasting model that can outperform models relying on a single approach. As evidenced by our results,the hybrid models have consistently achieved lower error rates and higher R^2 values, signifying their superior predictive performance and generalization capabilities. While our proposed hybrid models have shown promising results, there are several avenues for future research to further improve the accuracy and efficiency of time series forecasting. Firstly, we plan to explore the application of Transformer-based models, which have recently shown great success in natural language processing and computer vision tasks. By leveraging the self-attention mechanism, Transformer models can potentially capture more complex dependencies in multivariate time series data, leading to even more accurate forecasting results. Secondly, our objective is to investigate the feasibility of real-time time series forecasting using our proposed models. By developing efficient and scalable architectures, we can enable real-time forecasting applications in various domains.

Author Contributions Ammar Mohammed contributed to the Conceptualization, Methodology, Validation, Writing, Editing, and Supervision of this manuscript. Amal Mahmoud contributed to the Conceptualization, Methodology, Validation, Writing, Editing, Formal Analysis, Investigation, Software, Data Curation, and Visualization for this manuscript.

Funding 'Open Access funding provided by The Egyptian Knowledge Bank (EKB)'. Not applicable.

Data availability The dataset for this study may be made available upon request to the authors.

Declarations

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

Ethics approval Not applicable.

Consent to participate Not applicable.

References

1. Salman AG, Kanigoro B, Heryadi Y (2015) Weather forecasting using deep learning techniques. In: 2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 281–285. IEEE
2. Hossain M, Rekabdar B, Louis SJ, Dascalu S (2015) Forecasting the weather of nevada: A deep learning approach. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–6. IEEE
3. Sezer OB, Gudelek MU, Ozbayoglu AM (2020) Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Appl Soft Comput 90:106181
4. Lee SI, Yoo SJ (2020) Multimodal deep learning for finance: integrating and forecasting international stock markets. J Supercomput 76:8294–8312
5. Somu N, Ramamritham K (2021) A deep learning framework for building energy consumption forecast. Renew Sustain Energy Rev 137:110591
6. Deb C, Zhang F, Yang J, Lee SE, Shah KW (2017) A review on time series forecasting techniques for building energy consumption. Renew Sustain Energy Rev 74:902–924
7. Ensafi Y, Amin SH, Zhang G, Shah B (2022) Time-series forecasting of seasonal items sales using machine learning-a comparative analysis. Int J Inf Manag Data Insights 2(1):100058
8. Rafiei MH, Adeli H (2016) A novel machine learning model for estimation of sale prices of real estate units. J Constr Eng Manag 142(2):04015066
9. Tuarob S, Tucker CS, Kumara S, Giles CL, Pincus AL, Conroy DE, Ram N (2017) How are you feeling?: a personalized methodology for predicting mental states from temporally observable physical and behavioral information. J Biomed Inform 68:1–19
10. Morid MA, Sheng ORL, Dunbar J (2023) Time series prediction using deep learning methods in healthcare. ACM Trans Manag Inf Syst 14(1):1–29
11. Zhang Y, Cheng T, Ren Y (2019) A graph deep learning method for short-term traffic forecasting on large road networks. Computer-Aided Civ Infrastruct Eng 34(10):877–896
12. Nguyen H, Kieu L-M, Wen T, Cai C (2018) Deep learning methods in transportation domain: a review. IET Intel Transport Syst 12(9):998–1004
13. Wang W, Yildirim G (2021) Applied time-series analysis in marketing. In: Handbook of Market Research, pp. 469–513. Springer
14. Bouzidi Z, Amad M, Boudries A (2022) Deep learning-based automated learning environment using smart data to improve corporate marketing, business strategies, fraud detection in financial services, and financial time series forecasting. In: International Conference on Managing Business Through Web Analytics, pp. 353–377. Springer
15. Lim B, Zohren S (2021) Time-series forecasting with deep learning: a survey. Phil Trans R Soc A 379(2194):20200209
16. Hyndman RJ, Athanasopoulos G (2018) Forecasting: Principles and Practice. OTexts
17. Wei WW (2018) Multivariate time series analysis and applications. Wiley
18. Debnath A, Waghmare G, Wadhwa H, Asthana S, Arora A (2021) Exploring generative data augmentation in multivariate time series forecasting: opportunities and challenges. Solar-Energy 137:52–560
19. Faloutsos C, Gasthaus J, Januschowski T, Wang Y (2018) Forecasting big time series: old and new. Proceedings of the VLDB Endowment 11(12):2102–2105
20. Spiliotis E (2023) Time series forecasting with statistical, machine learning, and deep learning methods: Past, present, and future. In: Forecasting with Artificial Intelligence: Theory and Applications, pp. 49–75. Springer
21. He X (2023) A survey on time series forecasting. In: 3D Imaging-Multidimensional Signal Processing and Deep Learning: Multidimensional Signals, Video Processing and Applications, Volume 2, pp. 13–23. Springer
22. Kalouptsoglou I, Tsoukalas D, Siavvas M, Kehagias D, Chatzigeorgiou A, Ampatzoglou A (2022) Time series forecasting of software vulnerabilities using statistical and deep learning models. Electronics 11(18):2820
23. Rojas I, Valenzuela O, Rojas F, Guillén A, Herrera LJ, Pomares H, Marquez L, Pasadas M (2008) Soft-computing techniques and arma model for time series prediction. Neurocomputing 71(4–6):519–537
24. Akaike H (1969) Fitting autoregreesive models for prediction. In: Selected Papers of Hirotugu Akaike, pp. 131–135. Springer
25. Brownlee J (2018) Deep Learning for Time Series Forecasting: Predict the Future with MLPs. CNNs and LSTMs in Python, Machine Learning Mastery
26. Lim S, Kim SJ, Park Y, Kwon N (2021) A deep learning-based time series model with missing value handling techniques to predict various types of liquid cargo traffic. Expert Syst Appl 184:115532

27. Ismail Fawaz H, Forestier G, Weber J, Idoumghar L, Muller P-A (2019) Deep learning for time series classification: a review. Data Min Knowl Disc 33(4):917–963
28. Sharma A, Jain SK (2021) Deep learning approaches to time series forecasting. In: Recent Advances in Time Series Forecasting, pp. 91–97. CRC Press
29. Tripathy N, Hota S, Prusty S, Nayak SK (2023) Performance analysis of deep learning techniques for time series forecasting. In: 2023 International Conference in Advances in Power, Signal, and Information Technology (APSIT), pp. 639–644. IEEE
30. Hewamalage H, Bergmeir C, Bandara K (2021) Recurrent neural networks for time series forecasting: Current status and future directions. Int J Forecast 37(1):388–427
31. Zhu L, Laptev N (2017) Deep and confident prediction for time series at uber. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 103–110. IEEE
32. HADHBI A, KACEM M (2022) Convolutional neural networks for time series forecasting. PhD thesis, Université Ibn Khaldoun-Tiaret-
33. Li J, Wang Y (2021) Application of time-series smoothed excitation cnn model. In: 2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE), pp. 217–220. IEEE
34. Wan R, Mei S, Wang J, Liu M, Yang F (2019) Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. Electronics 8(8):876
35. Wang H, Zhang Z (2022) Tatcn: time series prediction model based on time attention mechanism and tcn. In: 2022 IEEE 2nd International Conference on Computer Communication and Artificial Intelligence (CCAI), pp. 26–31. IEEE
36. Sen R, Yu H-F, Dhillon IS (2019) Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. Advances in neural information processing systems **32**
37. Masood Z, Gantassi R, Choi Y (2022) A multi-step time-series clustering-based seq2seq lstm learning for a single household electricity load forecasting. Energies 15(7):2623
38. Lindemann B, Müller T, Vietz H, Jazdi N, Weyrich M (2021) A survey on long short-term memory networks for time series prediction. Procedia Cirp 99:650–655
39. Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P (2017) Geometric deep learning: going beyond Euclidean data. IEEE Signal Process Mag 34(4):18–42
40. Benidis K, Rangapuram SS, Flunkert V, Wang Y, Maddix D, Turkmen C, Gasthaus J, Bohlke-Schneider M, Salinas D, Stella L et al (2022) Deep learning for time series forecasting: tutorial and literature survey. ACM Comput Surv 55(6):1–36
41. Cressie N, Wikle CK (2015) Statistics for Spatio-temporal Data. Wiley
42. Gilanifar M, Wang H, Ozguven EE, Zhou Y, Arghandeh R (2019) Bayesian spatiotemporal gaussian process for short-term load forecasting using combined transportation and electricity data. ACM Transactions on Cyber-Physical Systems 4(1):1–25
43. Wang S, Zhang M, Miao H, Peng Z, Yu PS (2022) Multivariate correlation-aware spatio-temporal graph convolutional networks for multi-scale traffic prediction. ACM Trans Intell Syst Technol (TIST) 13(3):1–22
44. Casolaro A, Capone V, Iannuzzo G, Camastra F (2023) Deep learning for time series forecasting: advances and open problems. Information 14(11):598
45. Zivot E, Wang J (2006) Vector autoregressive models for multivariate time series. Modeling financial time series with S-PLUS®, 385–429
46. Kalaitzidakis P, Korniotis G (2000) The solow growth model: vector autoregression (var) and cross-section time-series analysis. Appl Econ 32(6):739–747
47. Pfeffermann D, Allon J (1989) Multivariate exponential smoothing: method and practice. Int J Forecast 5(1):83–98
48. Athanasopoulos G, De Silva A (2012) Multivariate exponential smoothing for forecasting tourist arrivals. J Travel Res 51(5):640–652
49. Lütkepohl H (2005) New Introduction to Multiple Time Series Analysis. Springer, Berlin
50. Yapar G, Yavuz I, Selamlar HT (2017) Why and how does exponential smoothing fail? An in depth comparison of ata-simple and simple exponential smoothing. Turk J Forecast 1(1):30–39
51. Fernandez P, et al (2016) Improving the vector auto regression technique for time-series link prediction by using support vector machine. In: MATEC Web of Conferences, vol. 56, p. 01008. EDP Sciences
52. Müller K-R, Smola AJ, Rätsch G, Schölkopf B, Kohlmorgen J, Vapnik V (1997) Predicting time series with support vector machines. In: International Conference on Artificial Neural Networks, pp. 999–1004. Springer
53. Thissen U, Van Brakel R, De Weijer A, Melssen W, Buydens L (2003) Using support vector machines for time series prediction. Chemom Intell Lab Syst 69(1–2):35–49

54. Boubrahimi SF, Ma R, Aydin B, Hamdi SM, Angryk R (2018) Scalable knn search approximation for time series data. In: 2018 24th International Conference on Pattern Recognition (ICPR), pp. 970–975. IEEE
55. Rooke C, Smith J, Leung KK, Volkovs M, Zuberi S (2021) Temporal dependencies in feature importance for time series predictions. arXiv preprint arXiv:2107.14317
56. Kurochkin A (2020) Discovering long term dependencies in noisy time series data using deep learning. arXiv preprint arXiv:2011.07551
57. Siami-Namini S, Tavakoli N, Namin AS (2019) The performance of lstm and bilstm in forecasting time series. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 3285–3292. IEEE
58. Zhang X, Shen F, Zhao J, Yang G (2017) Time series forecasting using gru neural network with multi-lag after decomposition. In: Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14–18, 2017, Proceedings, Part V 24, pp. 523–532. Springer
59. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
60. Tamal MBA, Alam MA, Sharker MN, Sazib MI (2022) Forecasting of solar photovoltaic output energy using LSTM machine learning algorithm. In: 2022 4th International Conference on Sustainable Technologies for Industry 4.0 (STI), pp. 1–6. IEEE
61. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555
62. Li P, Luo A, Liu J, Wang Y, Zhu J, Deng Y, Zhang J (2020) Bidirectional gated recurrent unit neural network for chinese address element segmentation. ISPRS Int J Geo Inf 9(11):635
63. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
64. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271
65. Li Y-H, Harfiya LN, Purwandari K, Lin Y-D (2020) Real-time cuffless continuous blood pressure estimation using deep learning model. Sensors 20(19):5606
66. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
67. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
68. Boureau Y-L, Ponce J, LeCun Y (2010) A theoretical analysis of feature pooling in visual recognition. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 111–118
69. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, pp. 630–645. Springer
70. Maetschke S, Tennakoon R, Vecchiola C, Garnavi R (2017) Nuts-flow/ml: data pre-processing for deep learning. arXiv preprint arXiv:1708.06046
71. Liashchynskyi P, Liashchynskyi P (2019) Grid search, random search, genetic algorithm: a big comparison for nas. arXiv preprint arXiv:1912.06059
72. Zhang Z (2018) Improved adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), pp. 1–2. Ieee
73. Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375
74. Popescu M-C, Balas VE, Perescu-Popescu L, Mastorakis N (2009) Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems 8(7):579–588
75. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14:199–222
76. Toda H (1991) Vector Autoregression and Causality. Yale University
77. Shumway RH, Stoffer DS, Shumway RH, Stoffer DS (2017) Arima models. Time series analysis and its applications: with R examples, 75–163
78. Vagropoulos SI, Chouliaras G, Kardakos EG, Simoglou CK, Bakirtzis AG (2016) Comparison of sarimax, sarima, modified sarima and ann-based models for short-term pv generation forecasting. In: 2016 IEEE International Energy Conference (ENERGYCON), pp. 1–6. IEEE
79. Tatachar AV (2021) Comparative assessment of regression models based on model evaluation metrics. Int J Innovat Technol Explor Eng 8(9):853–860