Evolutionary Intelligence (2025) 18:66
https://doi.org/10.1007/s12065-025-01053-7

RESEARCH PAPER

MCDFN: supply chain demand forecasting via an explainable multi-channel data fusion network model

Md Abrar Jahin[1,2] · **Asef Shahriar**[1] · **Md Al Amin**[1,3]

Received: 25 November 2024 / Revised: 1 March 2025 / Accepted: 29 April 2025 / Published online: 30 May 2025
© The Author(s) 2025

Abstract
Accurate demand forecasting is vital for optimizing supply chain management and enhancing organizational resilience. Traditional forecasting methods, relying on simple arithmetic, often fail to capture complex patterns caused by seasonal variability and special events. Although deep learning techniques have advanced, the lack of interpretable models hampers understanding and explaining predictions. We introduce the Multi-Channel Data Fusion Network (MCDFN), a novel hybrid deep learning architecture integrating multiple data modalities for superior demand forecasting. MCDFN utilizes Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), and Gated Recurrent Units (GRUs) to extract spatial and temporal features from time series data. Comparative benchmarking against seven other deep-learning models validates MCDFN's efficacy, showing it outperforms its counterparts across key metrics with a mean squared error (MSE) of 23.5738, root mean squared error (RMSE) of 4.8553, mean absolute error (MAE) of 3.9991, and mean absolute percentage error (MAPE) of 20.1575%. Theil's U statistic of 0.1181 ($U < 1$) of MCDFN indicates its superiority over the naive forecasting approach, and a 10-fold cross-validated statistical paired t-test with a p-value of 5% indicated no significant difference between MCDFN's predictions and actual values. To address the "black box" nature of MCDFN, we employ explainable AI techniques such as ShapTime and Permutation Feature Importance, offering insights into model decision-making processes. This research advances demand forecasting methodologies and provides practical guidelines for integrating MCDFN into existing supply chain systems.

Keywords Supply chain demand forecasting · Explainable artificial intelligence · Deep learning · Convolutional neural networks · Long short-term memory · Gated recurrent units

1 Introduction

Demand forecasting is the process of estimating the amount that consumers who wish to purchase things for personal use will need to pay for those that are now on hand [54]. Demand forecasting has always been vital to the supply chain (SC). A forecasting model with adequate accuracy can increase the company's profit margin. In a broader sense, it can make the organization's SC more responsive and resilient to failure. Traditional forecasting methods have been in practice for a long time. These methods are based on simple arithmetic and have low accuracy. However, seasonal variability, special days, and e-commerce have increased the complexity of demand forecasting nowadays. Researchers have applied sophisticated time series and machine learning (ML) models to meet this requirement. These models performed better to some extent than traditional regression models. Deep learning (DL) has recently emerged with immense potential

Md Abrar Jahin and Asef Shahriar have contributed equally to this work.

✉ Md Abrar Jahin
mdabrar.jahin@oist.jp; abrar.jahin.2652@gmail.com

✉ Md Al Amin
m.alamin@iem.kuet.ac.bd

Asef Shahriar
asef@iem.kuet.ac.bd

[1] Department of Industrial Engineering and Management, Khulna University of Engineering & Technology (KUET), Khulna 9203, Bangladesh

[2] Physics and Biology Unit, Okinawa Institute of Science and Technology Graduate University (OIST), Okinawa 904-0412, Japan

[3] Division of Engineering Management and Decision Sciences, College of Science and Engineering, Hamad Bin Khalifa University, 5825 Doha, Qatar

in every field. Their DL techniques could be a better fit for addressing the issues in SC demand forecasting. Little research focuses on applying DL networks for SC demand forecasting. Researchers have applied deep neural networks (DNN) and recurrent neural networks (RNN) in the past, but the networks and tuning process have evolved significantly. Therefore, it is high time we studied the fitness of the DL models in SC demand forecasting.

Integrating one's SC is a critical aspect of today's industry, and it's one that all upstream companies have come to understand, thanks to the growing relevance of SC integration. As a result of this type of integration, information is often shared between numerous business partners [16]. Information sharing and integration across the SC are increasingly important to businesses. Even though these programs reduce forecast mistakes, they are not universal or complete, and forecast errors continue to exist [15]. In SC management (SCM), demand is crucial information that may be shared and utilized. Demand projections that are accurate and trustworthy are essential for SC managers to use in their planning and decision-making processes. Many SC choices are based on demand forecasting, including demand planning and order fulfillment, production planning, and inventory management [2].

It is well acknowledged that information exchange throughout the SC is the most effective method of preventing demand signal distortions. Even though integrated SCs are enticing, implementing them is not always practical. Many factors could make such long-term, stable joint undertakings more difficult. Premkumar lists several essential challenges that must be solved to allow for successful SC collaboration [42]. Lengthy relationship management, unwillingness to share information, the complexity of managing an extensive SC, the ability of employees to support SC, and performance evaluation and incentives to encourage SC are just a few of the factors to consider. Despite several attempts to enable successful extended SC collaboration, most firms have failed to solve these challenges, and it is unlikely they will do so soon. More importantly, in many SCs, power regimes and sub-regimes can obstruct SC optimization [18, 58], which can be especially damaging to SC performance. Furthermore, power regimes and sub-regimes can impede SC optimization in many supply networks. So, even if integrating systems and information sharing is technically viable organizationally, it may not be feasible due to considerable disruptions in the power structure [41]. The bullwhip effect is magnified when SC participants share advanced knowledge.

Time series analysis focuses on extracting valuable statistics and other properties from time series data. This data has the most significant influence in making predictions based on data already collected. Demand forecasting in SCM has utilized various statistical analysis approaches, including time-series analysis and regression analysis [56]. It is now possible to use big data analytics (BDA), which takes advantage of the advances in information technology and improved computational efficiency, to make more precise predictions that better reflect customer needs, facilitate SC performance assessments, enhance SC efficiency, speed up SC response times, and help SC risk assessments [6]. Jahin et al. systematically identified and analyzed state-of-the-art SC forecasting strategies and technologies, proposing a novel framework integrating BDA into SCM [27]. They discussed the importance of collecting data tailored to SC strategies, differentiating forecasting needs based on time periods and SC objectives, and recommending performance evaluation metrics to optimize forecasting models and improve operational efficiency. Applying ML techniques such as neural networks (NN), SVM, and Decision Trees has been prominent in the literature since 2005. This research uses NNs most frequently [49]. Various DL models are established in this research to integrate the advanced predicting capabilities and scalability of DL in SC forecasting.

Information distortion is one of the troublesome and pricey phenomena in SC. Better collaboration and accurate forecasting are required to reduce the bullwhip effect resulting from this information distortion. In this research, our objective is to forecast future demand using hybrid DL techniques. It can boost the organization's service level and enable better scheduling and planning. Therefore, it will reduce the organization's costs to a beneficial extent. Improved demand forecasting offers considerable economic advantages by enhancing cost efficiency across various operational dimensions. Studies reveal that refined forecasting practices can yield operational cost savings of around 20%, as more precise demand estimates allow firms to streamline their operations and eliminate unnecessary expenditures. In the area of inventory management, more accurate forecasts enable companies to achieve savings between 5% and 10% on transportation, storage, and production planning while also reducing inventory holding costs and minimizing the risk of stockouts [31, 46]. Additionally, by aligning inventory more closely with actual demand, improved forecasting can decrease backorder rates by up to 30%, thereby enhancing customer satisfaction and lowering costs associated with expedited shipping and lost sales [47]. Moreover, avoiding forecasting inaccuracies is essential to mitigate opportunity costs, as even a modest 10% stockout rate can result in significant revenue losses, underscoring the pivotal role of economic precision in demand forecasting for maintaining competitive market performance [7, 38].

Despite significant advancements in applying DL and ML to demand forecasting in SCM, several key challenges remain. One major gap is the need for more robust and scalable DL models that can effectively handle the complexities and dynamic nature of SCs. Existing models often struggle with scalability and robustness, especially when applied to

large, real-world datasets. Traditional demand forecasting methods, such as autoregressive integrated moving averages (ARIMA) and exponential smoothing, face difficulties in capturing non-linearity and complex seasonality in SCs. While ML models like random forests and gradient boosting improve forecasting accuracy, they fail to handle heterogeneous data sources effectively. DL models, including LSTMs and transformers, offer further improvements but often function as black boxes, limiting interpretability and trust among practitioners. Enhancing model transparency is crucial for practical adoption in SCM. Although hybrid DL models have shown promise, there is still room for improvement, particularly in multi-horizon forecasting. To address these gaps, we propose Multi-Channel Data Fusion Network (MCDFN) as a hybrid approach integrating multi-channel parallel data inputs. It leverages CNNs for spatial feature extraction, LSTMs for capturing long-term dependencies, and GRUs for efficient sequence modeling. Additionally, its model-agnostic explainability module mitigates the "black-box" issue by providing interpretable feature importance metrics, making it a more practical solution for SC decision-making.

The contributions of this research article are as follows:

1. We introduce a novel hybrid neural network architecture called MCDFN, designed to analyze time series data comprehensively by incorporating multiple parallel channels within the network.
2. The MCDFN leverages diverse components such as CNN, LSTM, and GRU within its channels, enabling it to fuse complementary information from these distinct pathways to enhance overall data understanding. The MCDFN architecture integrates these channels within a unified framework, facilitating improved predictive performance and interoperability with time series data analysis.
3. The research includes advanced feature engineering and preprocessing techniques tailored to optimize the performance of MCDFN, showcasing best practices in handling complex time series data.
4. We benchmark MCDFN against seven DL models, demonstrating its superior accuracy and robustness across various evaluation metrics, including RMSE, MSE, MAPE, and MAE.
5. Through a 10-fold cross-validated statistical paired *t*-test for each benchmarked model's evaluation metrics (MSE, MAE, MAPE) against MCDFN, we show that MCDFN significantly outperforms other models at a 5% level of significance, validating its robustness and reliability.
6. We enhance the transparency of MCDFN by employing explainable artificial intelligence (XAI) techniques such as ShapTime and Permutation Feature Importance (PFI), addressing its "black box" nature.

The paper is organized as follows: In "Literature Review" section, we reviewed some previous works related to DL and forecasting. The forecasting techniques discussed include both DL and non-DL-based methods. The "Methodology" section includes the methodology related to data collection, preprocessing, feature engineering, experimental setup, model architecture, training, and hyperparameter optimization. In the "Results" section, we showed the comparative performance of the implemented models based on the benchmarking criteria. We also validated our proposed model's robustness using statistical paired *t*-tests and interpreted the model's predictions. The "Discussion" section presents our discussions on the practical and managerial implications of our research along with the benefits and significance of our proposed model. The "Conclusions" section summarized the whole research, discussed the limitations of this research, future scopes of development, and concluded the research.

2 Literature review

The application of DL in demand forecasting has seen significant advancements, especially in SCM. These innovations aim to enhance predictive accuracy and operational efficiency, which is crucial for maintaining robust and resilient SCs. This literature review explores recent studies from 2017 to 2025, highlighting the evolving methodologies and their implications in SCM. Recent research has introduced Temporal Fusion Transformers (TFT) as a powerful tool for handling multi-horizon forecasting problems [35]. TFT integrates gating mechanisms, variable selection networks, and attention-based models to enhance accuracy and interpretability. This model has demonstrated superior performance across various applications by effectively managing static and dynamic inputs and providing insights into feature importance and relevant time steps. Terrada et al. discussed the integration of AI within SCM 4.0, emphasizing the role of DL methods such as ARIMA and LSTM [53]. Their findings show that DL methods significantly improve demand forecasting accuracy using historical transaction data, thus enhancing decision-making processes and balancing supply and demand more effectively.

Recent studies have explored hybrid models for demand forecasting [19], advancements in machine learning for SC analytics [1], and XAI techniques in forecasting [20], while optimization frameworks for dynamic systems have also been investigated [21]. Attention-based models have shown great promise in demand forecasting by capturing long-term dependencies and relevant features from time series data. Google Research demonstrated that attention mechanisms could improve traditional DL models' performance by focusing on the most critical parts of the input data, thereby increasing forecast accuracy and

interoperability [5]. Techniques such as LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) have been applied to address the need for transparency in DL models. These methods help explain the predictions by showing the contribution of individual features, making DL models more user-friendly and trustworthy.

Punia et al. used a long short-term memory (LSTM) network, a variant of the DL network, for demand forecasting in multi-channel retail. They combined random forest algorithms and DL in their work. Their findings show that the proposed method can model complex relationships with elevated accuracy [44]. Punia, Singh, and Madaan proposed a cross-temporal hierarchical framework integrating DL for SC demand forecasting in another study. The resultant forecast is found to be coherent at all levels of the SC [43]. Wu et al. forecasted the COVID-19 period behavior of the oil market [60]. They used CNN to retrieve textual data from online oil news sources. Next, the Vector Autoregressive (VAR) model was utilized to ascertain the suitable lay order for the CNN outputs and historical data. Ultimately, various prediction techniques, such as MLR, BPNN, SVM, RNN, and LSTM, were employed to estimate the oil price, inventory, production, and consumption. Tang and Ge developed a material forecast model by combining CNN and LSTM and examining three independent variables: material attributes, transit warehouse inventory, and sales demand forecasting [52]. To predict respiratory disease bookings, Piccialli et al. suggested a hybrid architecture comprising CNN, LSTM-Autoencoder, and many ML algorithms [40]. Within this framework, CNN identifies short-term patterns and relationships between the forecasts produced by Ridge, Lasso, RFR, and XGB ML algorithms while the LSTM-Autoencoder extracts the features. A hybrid architecture built on the CNN and LSTM was presented by Yasutomi and Enoki to inspect belt conveyors in production or distribution processes to detect any irregularities in the belt lines [61]. The LSTM determines the relationship between the windows in this manner after the CNN's convolutional layers extract signal information from each window. A classification model utilizing an interpretable hybrid quantum-classical neural network (NN) was presented by Jahin et al. to forecast whether or not a product order will be back-ordered after benchmarking against 17 different models [26]. They used seven different preprocessing approaches to address the class imbalance issue in this particular situation.

Many researchers applied ML techniques for demand forecasting outside the SC. Bose et al. built an end-to-end ML system using Apache Spark [12]. Their primary focus was on demand forecasting in retail. Their model is superior because it includes distributed learning, evaluation, and ensembling. Ke et al. used DL to forecast short-term passenger demand under on-demand ride services [28]. They proposed a novel approach called FCL-Net. They concluded the model performed better than traditional time-series prediction methods and NN-based algorithms (e.g., ANN and LSTM). The addition of exogenous variables reduced RMSE by 50.9% in the study [28]. Electronic power industries are using DL models extensively for load demand forecasting. Qiu et al. presented an ensemble method integrating EMD with DL. EMD was used to decompose the demand series into some number of IMFs. Then DBN, a class of DNN, is employed to model these IMFs. They also included RBMs in the DBN. The model was more attractive than nine other models studied in the research [45].

Another study on electricity load application was conducted by Torres et al. Their method supports arbitrary time horizons and exhibits less than 2% error margin [55]. DL has the potential to pave the way for smart-grid technology. Amarasinghe et al. explored CNN for energy load forecasting at the building level. They mentioned CNN outperformed the support vector regression model. In conclusion, they suggested more experiments on this topic [3]. Bouktif et al. studied LSTM for electric load forecasting to provide better load scheduling and reduce unnecessary production [10]. They fused GA in their model to find the number of layers and the optimal value of time lags. Their model was successful in capturing the features of the time series. As a result, reduced MAE and RMSE are observed in the forecasting [10]. Antunes et al. applied ML techniques to reduce instantaneous response to water demand using forecasting [4]. They applied NN, RF, SVM, and KNN to data collected from Portuguese water utilities. They commented that developing and implementing forecasting-based responses in the system could reduce cost by 18% or more [4]. Law et al. studied DL methods for forecasting monthly tourism demand in Macau [32]. They stated that it has become difficult for existing models to accurately forecast data by adding many search intensity indicators. Their first contribution is developing a systemic conceptual model that uses all tourism demand factors without human intervention. Employing the attention score to portray the DL models is their second contribution [32]. Zhang et al. addressed two primary issues of tourism demand forecasting with DL: data inaccessibility and the requirement of explanatory auxiliary variables [62]. Therefore, they proposed a decomposition model, STLDADLM, rather than a complex overfitted model [62]. Bandara et al. utilized LSTM to forecast Walmart's e-commerce demand. They also developed a synthetic pre-processing unit to overcome challenges in e-commerce-based forecasting. Their model achieved noteworthy improvement over state-of-the-art techniques in some categories of products [8]. To create a customized tourism service system, Wang suggested a classification model based on CNN, DNN, and factorization machine technology [57]. The factorization machine technology in this model learns the interaction

between the extracted features while the CNN processes review information about users and tourist service items, and the DNN processes the necessary information about users and tourism service things. To choose the best forecasting model among the Naïve, Simple moving average, Single exponential smooth, Syntetos-Boylan approximation (SBA), ANN, RNN, and SVR models, Chien et al. suggested a demand forecasting framework utilizing the DRL model of DQN [17].

BDA and ML techniques are widely developed for SC demand forecasting. NN, Regression, ARIMA, SVM, and Decision Trees are the most popular methods for demand forecasting among researchers, as found by Seyedan and Mafakheri. They studied applications of BDA in demand forecasting in SCM. They shed light on the limitations of conventional methods and how BDA enables us to overcome these barriers [49]. In 2018, Liao et al. presented a DL model for urban taxi demand forecasting. They compared ST-ResNet and FLC-Net in their study. The study emphasized the proper DNN structure and domain knowledge as most DNN models are superior to conventional ML techniques with proper design and tuning [34]. Shi et al. applied pooling-based deep RNN to forecast uncertain household load. This novel approach surpassed ARIMA by a margin of 19.5%, conventional RNN by 6.5%, and SVR by 13.1% [50]. Cai et al. contrasted the DL approach with conventional time series models for building-level load forecasting. They found the highest result with multi-step formulated CNN. It is also computationally less expensive due to a smaller number of parameters in the CNN. Their CNN model provided 26.6% more accurate forecasts than seasonal ARIMAX [14]. Huber and Stuckenschmidt used quite a different approach, considering the forecasting problem as a classification problem rather than a regression problem. In their study, ML-based classification algorithms surpassed regression algorithms. Their focus was on the influence of special days on demand variability. They concluded ML models are best suited for large-scale demand forecasting applications alongside providing more accuracy [24]. Ma et al. proposed a SARIMA-LSTM-BP combination model for demand forecasting in the new energy vehicle industry. This model outperformed traditional and individual DL models such as Random Forest, Support Vector Regression (SVR), and standalone LSTM networks in terms of RMSE (2.757%), MSE (7.603%), and MAE (1.912%) metrics [36].

3 Methodology

Table 1 provides an overview of the technologies and tools used in implementing and experimenting with our DL models. Figure 1 illustrates the comprehensive framework of the MCDFN model, designed for enhanced demand forecasting. It begins with raw data extraction from a point of sale (POS) system or ERP, focusing on date and sales information. The data preprocessing phase includes several critical steps: date feature extraction, which breaks down date information into granular components such as year, week, day of the month, and month; cyclic feature encoding, where these components are transformed using sine and cosine functions to capture seasonal patterns and periodicity; and the inclusion of a binary feature to indicate holidays, accounting for irregular demand spikes. Subsequently, the data is normalized to standardize the input range, an essential step for model performance and convergence. The dataset is then partitioned into training, validation, and test sets, ensuring a robust model training and evaluation process. Sliding windows of time series data are generated to create input sequences for the NN, enabling the model to learn from temporal patterns.

The core of the figure highlights the MCDFN model, which incorporates multiple parallel channels utilizing CNN, LSTM, and GRU to process the same input data through distinct pathways. This multi-channel approach allows for the extraction of diverse representations or features, which are then fused to enhance the overall understanding of the data. The model outputs are used to generate demand forecasts, which can inform inventory management systems and future order sizes or frequencies, thereby aiding suppliers in optimizing their operations. Additionally, the figure outlines the application of XAI techniques such as PFI and ShapTime, which provide insights into the model's decision-making process, enhancing interpretability and trust in the model's predictions. The implications of these forecasts are significant, impacting inventory management systems and supplier operations and demonstrating the practical utility of the MCDFN model in real-world scenarios.

3.1 Data description and preprocessing

The historical sales dataset used in this research was meticulously collected from a prominent Bangladeshi retailer, ensuring a comprehensive and high-quality data source. The dataset spans 1826 days, covering the period from January 1, 2013, to December 31, 2017, by Jahin and Shahriar [25], as depicted in Fig. 2. This extensive timeframe allows for capturing seasonal variations, trends, and other temporal dynamics critical for accurate demand forecasting.

3.1.1 Dataset characteristics

The dataset comprises 1826 daily sales data for a specific product, recorded precisely to ensure reliability. Each record in the raw sales data includes two essential columns:

1. **Timestamp feature:** This column records the exact date of each sale, providing a continuous chronological

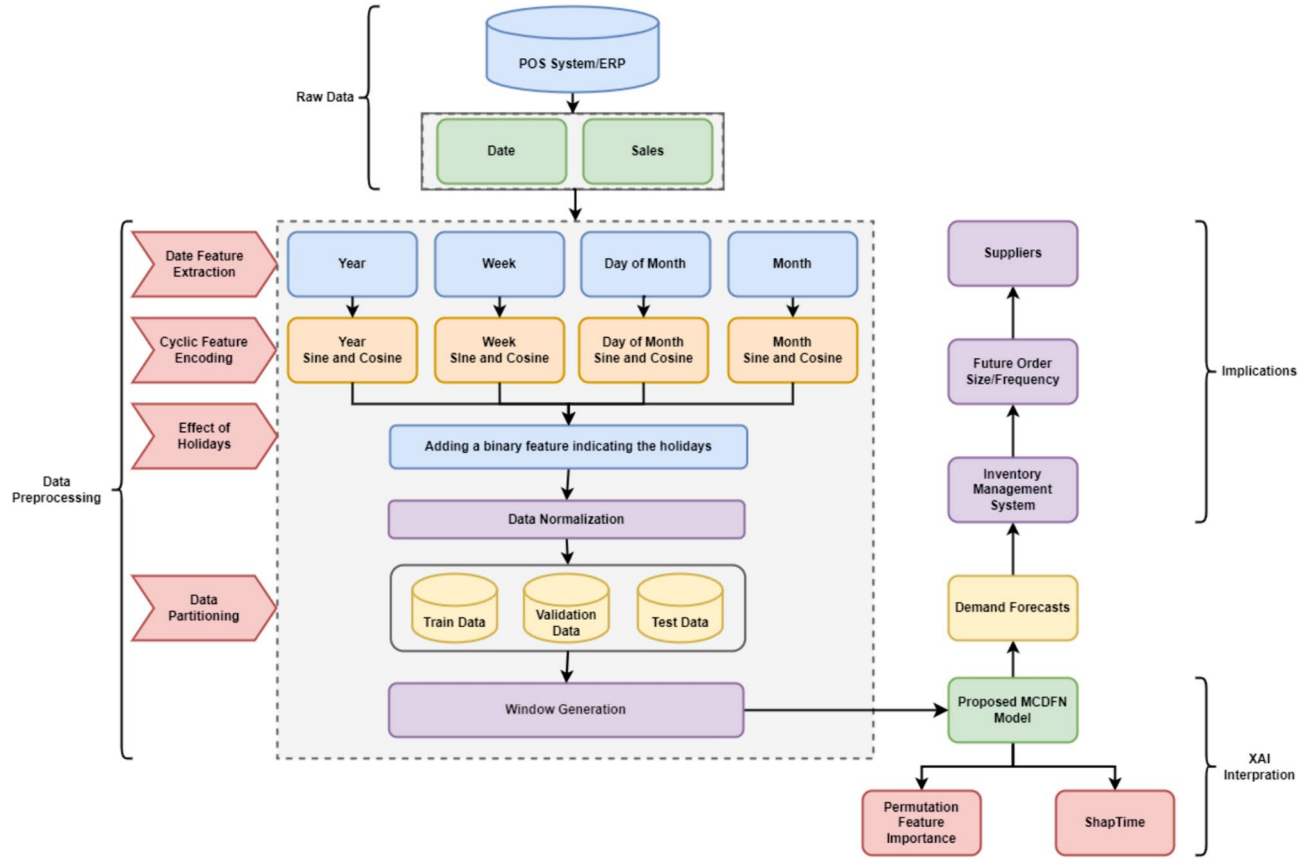


Fig. 1 Proposed MCDFN methodological framework for supply chain demand forecasting

Table 1 Experimental setup for conducting this research

Component	Details
Programming language	Python 3.7
GPU	1 x Tesla K80 (2496 CUDA cores)
Processor	1 x single core hyperthreaded Xeon Processor @2.3GHz
RAM	13 GB RAM
Storage	108 GB Runtime HDD
Operating system	Linux Kernel
Libraries	Pandas, NumPy, Matplotlib, TensorFlow, Keras, KerasTuner

sequence vital for time series analysis. The timestamps are formatted in a standard date format (YYYY-MM-DD), facilitating easy parsing and manipulation for analytical purposes.

2. **Quantity sold feature:** This column reflects the units sold on each corresponding day. The sales quantities are recorded as integers, representing the daily demand for the product.

In addition to these primary features, external factors such as holidays were encoded as binary variables (e.g., *is_holiday*) to capture periods of potentially higher sales. While the dataset does not include external factors like weather or macroeconomic indicators, including holiday information enhances the model's ability to account for irregular demand spikes. Key observations are:

- **Seasonality:** Product demand exhibits seasonality, being lower at the beginning and end of each year and peaking in the middle.
- **External Factors:** Holidays significantly influence sales patterns, necessitating their inclusion as a binary feature.
- **Missing Values:** The dataset is complete, with no missing values reported. However, preprocessing steps were implemented to ensure robustness in case of future datasets with incomplete data.

3.1.2 Preprocessing steps

To prepare the dataset for training and evaluation, several preprocessing steps were undertaken:

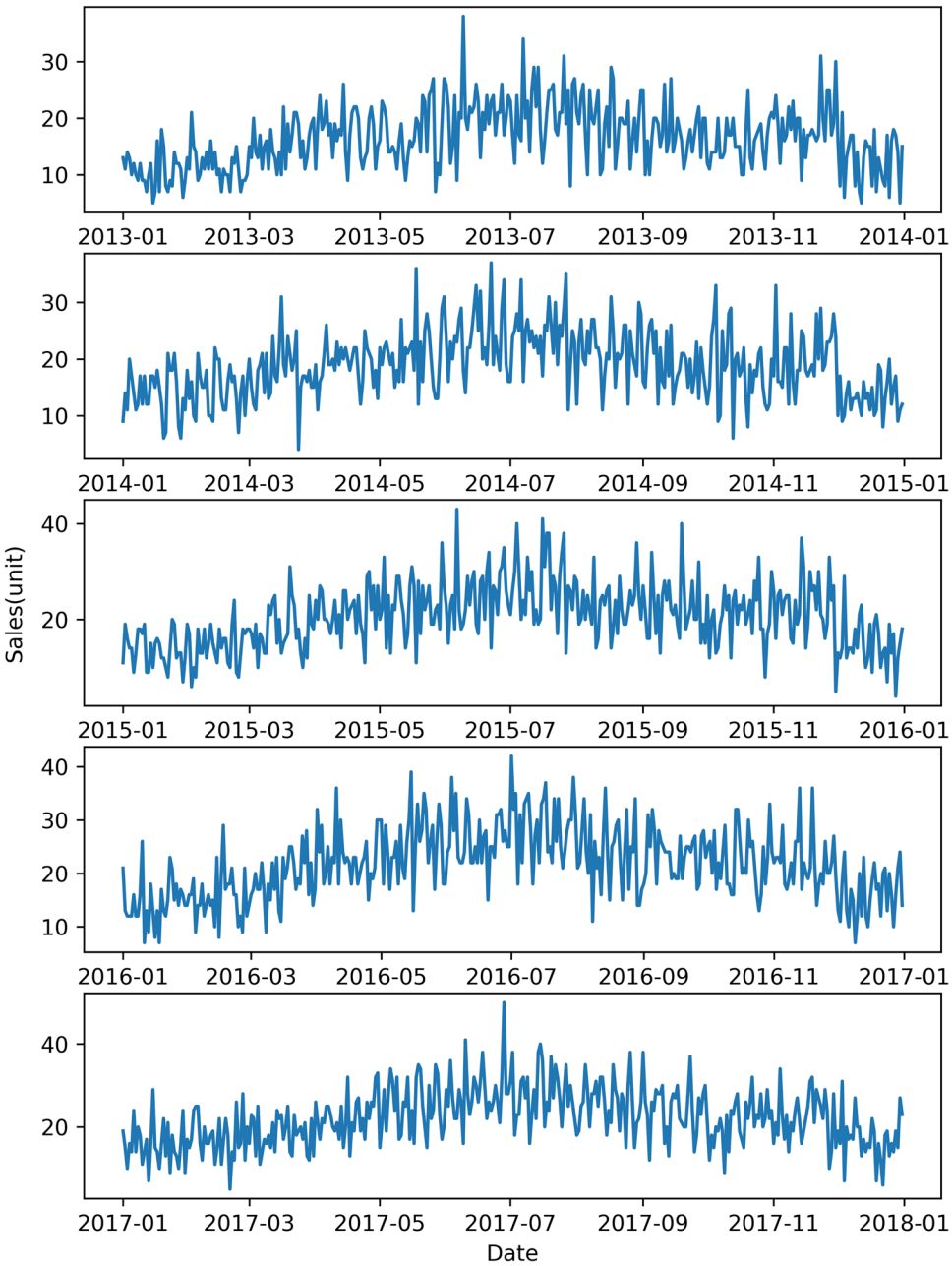


Fig. 2 Yearly product consumption pattern (from 2013 to 2017)

Cyclic Encoding: To handle the periodic nature of time series data, DateTime features such as 'Day of Month', 'Week', 'Month', and 'Year' were extracted and transformed into cyclical features using sine and cosine transformations. This approach ensures that the periodicity is represented in a form that machine learning models can interpret accurately. Equations 1-4 illustrate the transformation process:

$$\text{month_sin} = \sin\left(\text{month} \times \frac{2 \cdot \pi}{\max(\text{month})}\right) \tag{1}$$

$$\text{month_cos} = \cos\left(\text{month} \times \frac{2 \cdot \pi}{\max(\text{month})}\right) \tag{2}$$

$$\text{day_sin} = \sin\left(\text{day_of_month} \times \frac{2 \cdot \pi}{\max(\text{day_of_month})}\right) \tag{3}$$

$$\text{day_cos} = \cos\left(\text{day_of_month} \times \frac{2 \cdot \pi}{\max(\text{day_of_month})}\right) \tag{4}$$

Here, max(month) and max(day_of_month) represent the maximum values for the month (12) and days of the month

(30), respectively. The sine and cosine functions transform these values into cyclical features, capturing the periodic nature of these time components. Later equations 5, 6, 7 and 8 are utilized to convert 'Week' and 'Year' features into cyclical signals.

$$week_{sin} = \sin(timestamp * \left(2 * \frac{\pi}{week_{sec}}\right) \tag{5}$$

$$week_{cos} = \sin(timestamp * \left(2 * \frac{\pi}{week_{sec}}\right) \tag{6}$$

$$year_{sin} = \sin(timestamp * \left(2 * \frac{\pi}{week_{sec}}\right) \tag{7}$$

$$year_{cos} = \cos(timestamp * \left(2 * \frac{\pi}{year_{sec}}\right) \tag{8}$$

where,

$week_{sec} = 24 \times 60 \times 60 \times 7 = 604800$ s for a week,

$year_{sec} = 24 \times 60 \times 60 \times 365.2425 = 220898664$ s for a year.

Figure 3 depicts multiple subplots showcasing various aspects of sales data and cyclic features over time. The main subplot (top) displays the sales trend over the observed period, illustrating fluctuations in sales over the course of the year. Below, four additional subplots exhibit cyclic features represented by sine and cosine functions for different time components: day of the month, month, week, and year. Each subplot illustrates the cyclic patterns present in the corresponding time component, with sine and cosine curves depicting the seasonal variations throughout the year.

Holiday Feature: Some products have higher sales during holidays. An 'is_holiday' feature was included to capture periods of potentially higher sales.

Normalization: Ensuring proper feature scaling is imperative in the preparatory stages of NN training. In

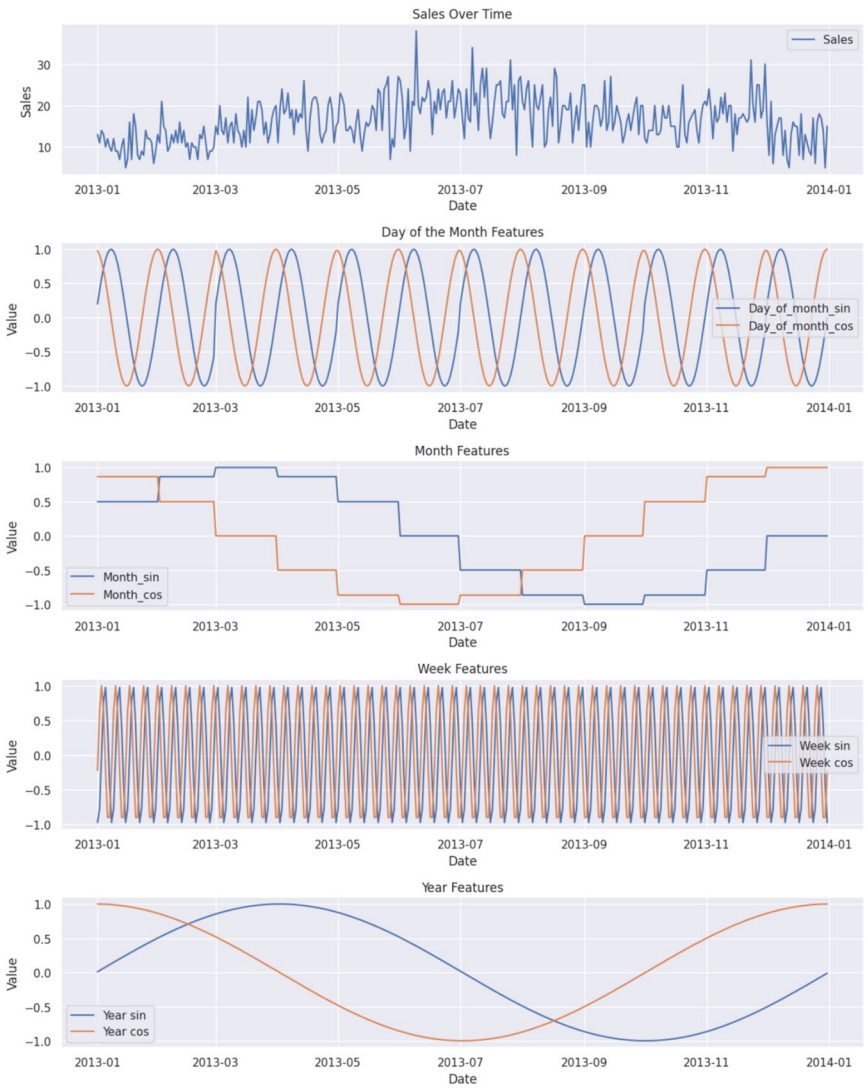


Fig. 3 Time-series analysis of sales data and feature engineered cyclic features over the course of a year

sales forecasting, there is a scale difference between sales and other features. A prevalent method employed for this purpose is normalization, wherein the mean and standard deviation of each feature (except for the 'is_holiday' feature) are subtracted and divided, respectively (Fig. 4). This procedure aids in standardizing the range of feature values, thereby enhancing the efficiency of network convergence. The 'is_holiday' feature is already in binary format, requiring no further normalization. Consequently, accurate training cannot be performed when inputs are used without being normalized. Moreover, sales follow Gaussian distribution inherently. After reviewing available normalization methods, the sale feature is standardized using the following equation.

$$z_{sales} = \frac{sales_t - \mu_{sales}}{\sigma_{sales}} \tag{9}$$

Where,

z_{sales} = standardized sales value,
$sales_t$ = original sales value,
μ_{sales} = mean of the sales feature, and
σ_{sales} = standard deviation of the sales feature.

Crucially, the computation of mean and standard deviation was solely based on the training dataset to prevent any accidental influence from the validation and test datasets. This segregation ensured the integrity of model evaluation metrics during the validation and testing phases.

Handling Missing Values: Although the dataset used in this study is complete, preprocessing pipelines were designed to handle missing values in future datasets. Techniques such as forward-filling, backward-filling, or interpolation could be applied to address gaps in time-series data.

3.1.3 Data partitioning

The dataset was divided into training (70%), validation (20%), and testing (10%) sets sequentially, respecting the time-series structure of the data (Fig. 5). The absence of random shuffling of data before partitioning is motivated by two primary considerations. Firstly, this practice facilitates data segmentation into contiguous windows of consecutive samples, thereby preserving temporal coherence and facilitating sequential data analysis. Secondly, the deliberate avoidance of random shuffling ensures that the validation and test outcomes reflect a more authentic evaluation scenario, as they are based on data instances collected subsequent to the training phase of the model. This approach enhances the generalization capability of the model by simulating real-world conditions more accurately.

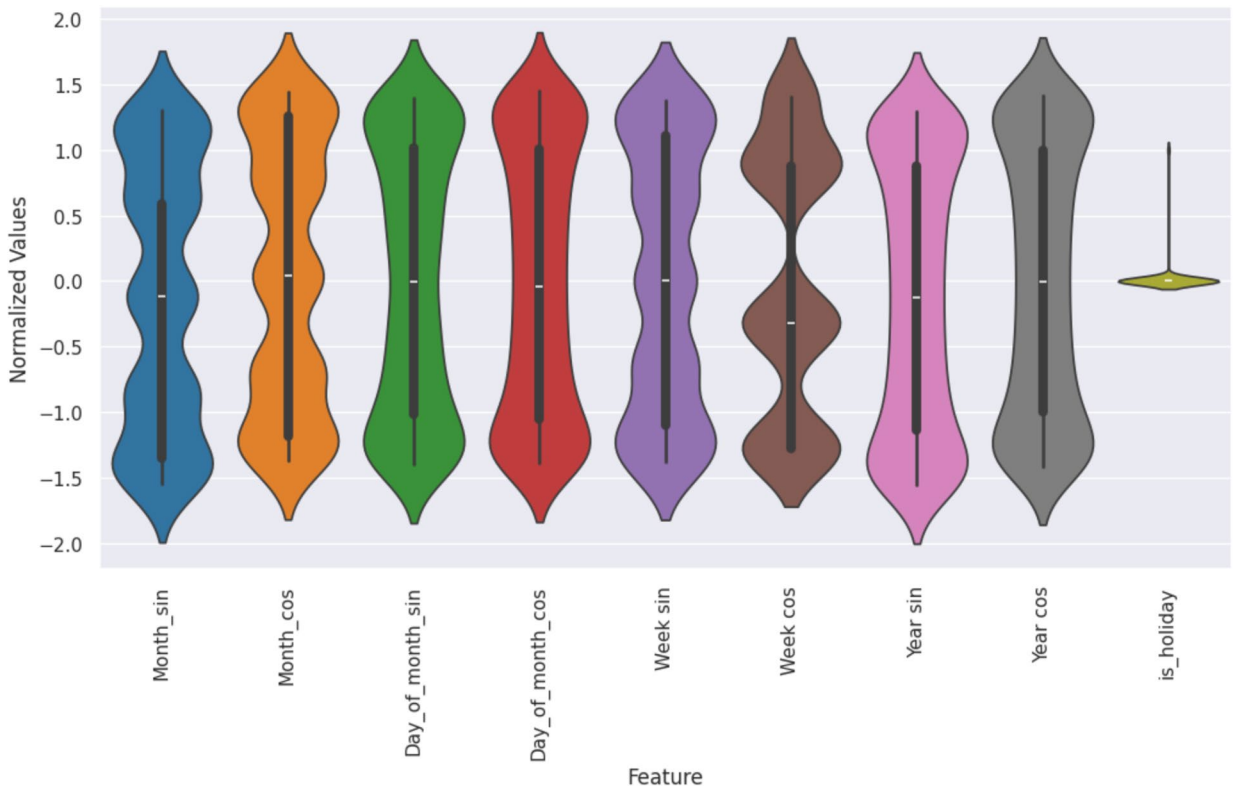


Fig. 4 Violin plot of the normalized features

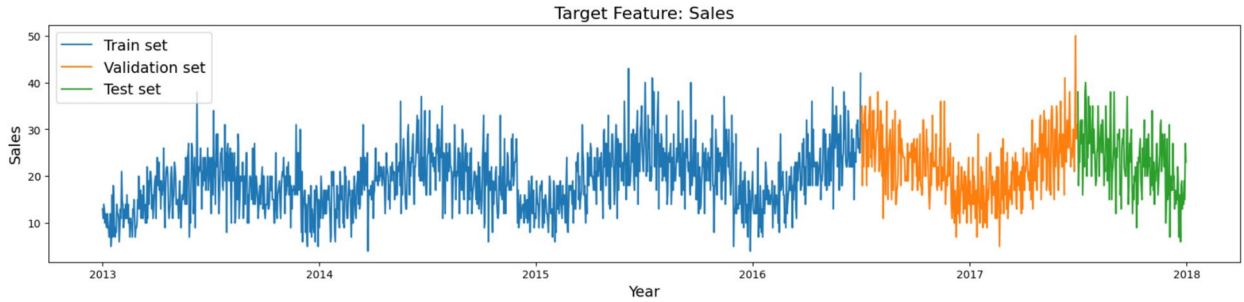


Fig. 5 Data splitting into training, validation, and test sets

3.1.4 Window generation

The windowing approach is pivotal in structuring the input–output relationships for training DNNs in time series forecasting tasks. Here, we present a comprehensive windowing methodology tailored for temporal data analysis, specifically aimed at enhancing the effectiveness of predictive modeling tasks.

Previous sales history contains vital information about the specific behaviors of the customers. Data windows with 30×10 inputs and 30×1 outputs for every window were created to incorporate this information. The windows primarily symbolize input and output sequences. The number 30 represents the data spanning 30 days of a month. We structured the input by mapping 30 consecutive days per sequence. Subsequently, the model forecasted the data sequence with the same length, shifting 30 days from the beginning of the input sequence. Consequently, the model utilizes 30 days of sales data as an input reference to forecast demand for the subsequent 30 days into the future, as illustrated in Figure 6. Within the 30×10 framework, the number 10 signifies the number of features anticipated in the following step. These features encompass encoded signals indicating the day, week, year, sales data, and binary indicators denoting holiday occurrences. In Figure 7, the x-axis represents the period (days), indicating the temporal progression of the data. The y-axis depicts the normalized actual sales values.

3.2 Deep learning models

In multi-step prediction tasks, the modeling paradigm shifts from forecasting individual future points to predicting a sequence of future values, requiring a more sophisticated

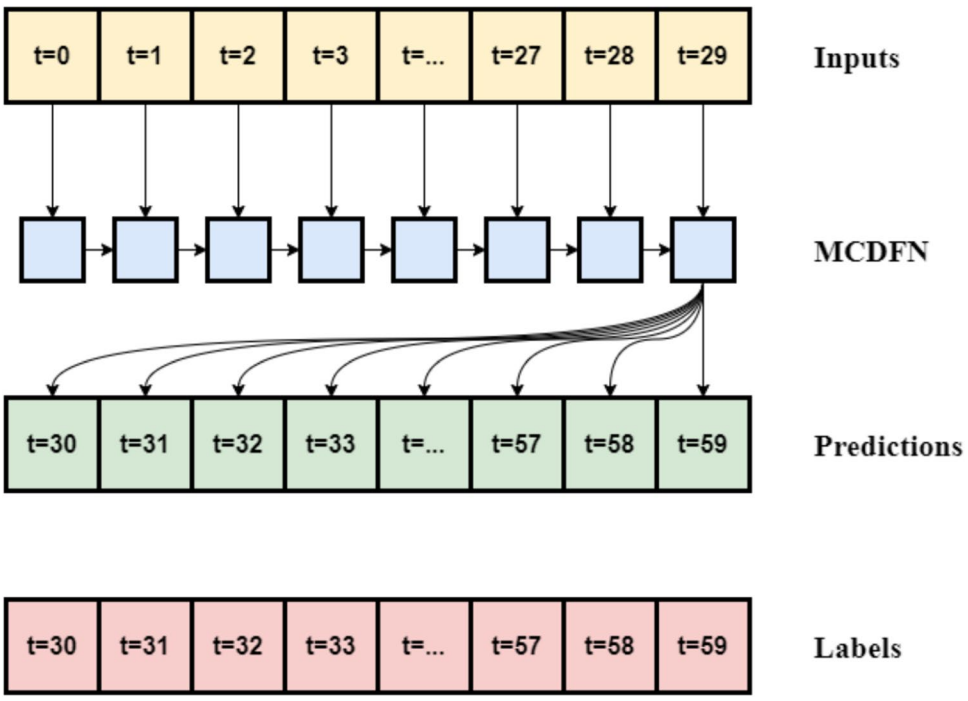


Fig. 6 Illustration of the window generation process for single-shot deep learning models, where the model accumulates internal state over a 30-day period and subsequently predicts the demand for the following 30 days

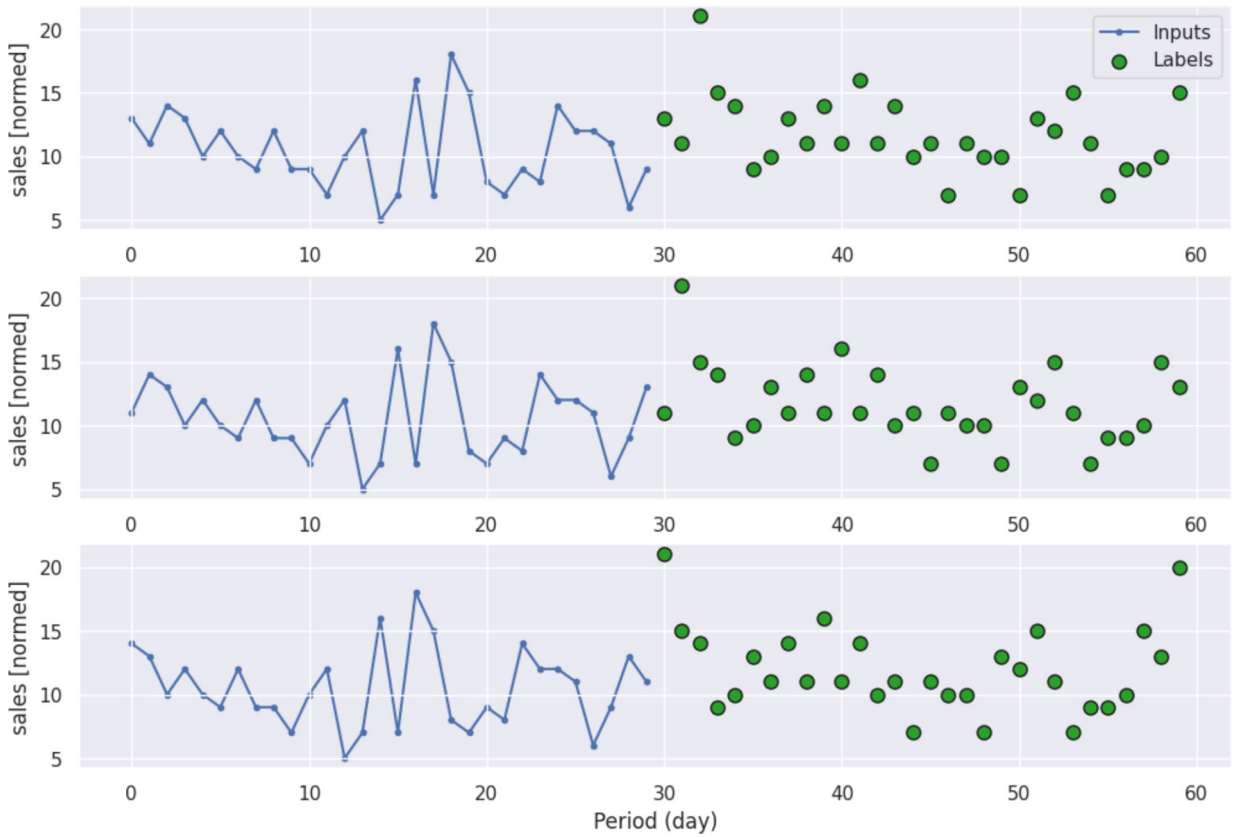


Fig. 7 Instances of generated windows for training, testing, and validation

approach. Unlike single-step forecasting, which predicts only a single future point, a multi-step model generates a coherent sequence of future observations. A prominent strategy for this is the "single-shot" model, where the entire sequence prediction is formulated and executed in a single inference step. In our research, we adopted the single-shot prediction methodology, enabling the model to forecast the complete time series trajectory simultaneously. This approach ensures that the demands across all output time steps are predicted concurrently, fostering a comprehensive understanding of the temporal dynamics in the dataset. Specifically, our multi-step models are trained using daily samples as input data to predict 30 days into the future based on a preceding window of 30 days' historical data (Figures 6 and 7). We developed and benchmarked MCDFN against seven fine-tuned state-of-the-art DL models in SC demand forecasting, including BiLSTM, CNN, RNN, Vanilla LSTM, Stacked LSTM, FCN, and GRU.

3.2.1 BiLSTM

The bidirectional Long Short-Term Memory (BiLSTM) model represents a pivotal component in our predictive modeling pipeline, tailored specifically for demand forecasting. This structure aligns with the temporal nature of the forecasting task, where each input sequence spans multiple time steps, with 10 features characterizing each time point. The model architecture incorporates a BiLSTM layer of 192×2 units strategically chosen to leverage both forward and backward temporal information during sequence processing. Within this layer, the number of LSTM units is dynamically determined through hyperparameter tuning facilitated by the Keras Tuner library. BiLSTM utilizes equations 10 and 11 as forward and backward passes.

Forward LSTM:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{10}$$

Backward LSTM:

$$
\begin{aligned}
i'_t &= \sigma(W'_{xi}x'_t + W'_{hi}h'_{t+1} + W'_{ci}c'_{t+1} + b'_i)\\
f'_t &= \sigma(W'_{xf}x'_t + W'_{hf}h'_{t+1} + W'_{cf}c'_{t+1} + b'_f)\\
c'_t &= f'_t \odot c'_{t+1} + i'_t \odot \tanh(W'_{xc}x'_t + W'_{hc}h'_{t+1} + b'_c)\\
o'_t &= \sigma(W'_{xo}x'_t + W'_{ho}h'_{t+1} + W'_{co}c'_t + b'_o)\\
h'_t &= o'_t \odot \tanh(c'_t)
\end{aligned}
\tag{11}
$$

Where:

x_t = the input at time step t
h_t = the hidden state at time step t
c_t = the cell state at time step t
i_t, f_t, o_t = the input, forget, and output gates, respectively
σ = the sigmoid activation function
$\odot$ = element-wise multiplication
W = the weights
b = the biases.

The activation functions, namely the hyperbolic tangent (tanh) and the sigmoid function for recurrent connections imbue the model with non-linearity and memory retention capabilities crucial for sequence modeling. A dropout layer with a dropout rate of 0.2 is employed after the LSTM layer to mitigate overfitting and enhance generalization performance. Following sequence processing, a flattening operation transforms the output into a one-dimensional vector, facilitating subsequent dense layer operations. A densely connected layer with an output dimensionality of 30 is introduced, initializing the kernel weights with zeros. This layer maps the extracted temporal features into the desired output space, effectively reconstructing the predicted sequence.

3.2.2 CNN

The Convolutional Neural Network (CNN) model architecture employed in this study comprises multiple layers designed to extract relevant features from temporal sequences. The model begins with a Conv1D layer comprising 64 filters and kernel size 1 to capture local patterns within the input sequences. Subsequently, an AveragePooling1D layer is applied to down-sample the feature maps, effectively reducing computational complexity while preserving essential information. Following the pooling operation, the feature maps are flattened into a 1D vector using the Flatten layer. The equations involved related to the convolution and pooling operations are shown in equations 12 and 13.

Convolution operation:

$$
z_{i,j} = (X \times W)_{i,j} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X_{i+m,j+n} \cdot W_{m,n} \tag{12}
$$

Where X is the input data, W is the filter/kernel, and $z_{i,j}$ is the output of the convolution operation at position (i,j).

Pooling operation:

$$
y_{i,j} = \text{Pooling}(x_{i,j}) = \text{Max}(x_{i,j}) \tag{13}
$$

Where $x_{i,j}$ is a region of the input data and $y_{i,j}$ is the output of the pooling operation.

A dense layer with 192 units is introduced to enable non-linear transformations and higher-level feature extraction. Finally, the output layer, consisting of a dense layer with 30 units, predicts the target values for the subsequent 30 time steps. A Reshape layer is applied to reshape the output into the desired format, aligning with the temporal structure of the forecasting task. The Hyperband tuner selected the above optimal hyperparameters when the model demonstrated a minimized validation mean squared error of 26.23 after 85 trials.

3.2.3 RNN

The recurrent neural network (RNN) model architecture used in our study begins with an input layer tailored to the shape of our data, followed by a SimpleRNN layer. This RNN layer's units are determined through hyperparameter tuning, allowing values between 32 and 512 in increments of 32, with the 'tanh' activation function effectively capturing sequential dependencies. The RNN model uses equation 14 as the forward pass.

$$
h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{14}
$$

Where:

h_t is the hidden state at time step t
x_t is the input at time step t
W_{hh} and W_{xh} are the weights, and
b_h is the bias.

A Dropout layer, with a dropout rate tunable from 0 to 0.5 in steps of 0.1, is included to mitigate overfitting. The model then flattens the RNN outputs and passes them through a Dense layer initialized with zeros, followed by a reshaping layer to ensure the output dimensions match the desired forecast horizon of the next 30 days. The Hyperband tuner selected the optimal recurrent units of 128 neurons and dropout of 0.1 for RNN when the model demonstrated a minimum validation mean squared error of 23.37 after 84 trials.

3.2.4 Vanilla LSTM

Our research used a vanilla LSTM model consisting of an input layer with shape (30, 10), followed by a single LSTM layer with 480 units, determined through hyperparameter tuning,

using 'tanh' activation and 'sigmoid' recurrent activation. The LSTM output is processed through a Dense layer, initialized with zeros, to ensure unbiased predictions and reshaped to match the required output dimensions of (30, num_features). Vanilla LSTM achieved the best validation mean squared error of 39.93 after 6 trials.

3.2.5 Stacked LSTM

The stacked LSTM model begins with an input layer for data shaped (30, 10), followed by two LSTM layers. Each LSTM layer is configured with 512 units, which was determined through hyperparameter tuning and employs 'tanh' activation and 'sigmoid' recurrent activation functions. Both LSTM layers use the 'return_sequences=True' parameter to pass the sequence output to the next layer. Between the LSTM layers, a Dropout layer is included to mitigate overfitting, with 0 dropout rate after tuning. After the LSTM layers, the model output is flattened and passed through a Dense layer, initialized with zeros, ensuring unbiased predictions. Finally, the output is reshaped to the desired dimensions (30, 1). Stacked LSTM was evaluated and achieved the best validation mean squared error of 28.17 after 58 trials, demonstrating its ability to capture complex temporal dependencies and provide accurate demand forecasts for an output sequence length of 30 steps.

3.2.6 FCN

Our Fully Connected Network (FCN) model starts with an input layer to handle data shaped (30, 10), followed by a dense layer with 512 units chosen through hyperparameter tuning. The activation function for this layer was selected as 'tanh' from the options 'relu' and 'tanh'. A Dropout layer with a rate of 0.0 is included to prevent overfitting. Finally, the model outputs predictions through a dense layer with a linear activation function to match the dimensionality of the target feature. The FCN model was evaluated and achieved the best validation mean squared error of 27.79 after 54 trials.

The input layer of an FCN receives the input vector $\mathbf{x}$. For each hidden layer l (where $l = 1, 2, \ldots, L$ with L being the number of hidden layers), the output $\mathbf{h}^{(l)}$ is computed using equation 15.

$$\mathbf{h}^{(l)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{15}$$

Here, $\mathbf{h}^{(0)}$ is the input vector $\mathbf{x}$, $\mathbf{W}^{(l)}$ is the weight matrix for layer l, $\mathbf{b}^{(l)}$ is the bias vector for layer l, and σ is an activation function (e.g., ReLU, sigmoid, tanh). The output layer produces the final output vector $\mathbf{y}$ using equation 16.

$$\mathbf{y} = \phi\left(\mathbf{W}^{(L+1)}\mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}\right) \tag{16}$$

Here, $\mathbf{W}^{(L+1)}$ is the weight matrix for the output layer, $\mathbf{b}^{(L+1)}$ is the bias vector for the output layer, and ϕ is the activation function for the output layer (could be the same as σ or different, depending on the task).

These equations describe the forward propagation process of an FCN, capturing the transformations from the input through the hidden layers to the output, with each layer applying a linear transformation followed by a non-linear activation function to produce the final prediction.

3.2.7 GRU

Our research employed a Gated Recurrent Unit (GRU) model, which begins with an input layer to handle data shaped (30, 10), followed by a GRU layer configured with 192 units, a hyperparameter value determined through extensive tuning. The GRU layer uses 'return_sequences=True' to maintain the sequential nature of the data. A Dropout layer with a dropout rate of 0.4 is included to prevent overfitting. After the GRU and dropout layers, the model's output is flattened and passed through a Dense layer, initialized with zeros, to ensure unbiased predictions. Finally, the output is reshaped to match the required dimensions of (30, 1). GRU model achieved the best validation mean squared error of 23.84 during tuning after 16 trials, demonstrating its capability to accurately capture temporal dependencies and provide robust demand forecasts for an output sequence length of 30 steps. This architecture and its hyperparameters were optimized to enhance prediction accuracy while managing model complexity.

A GRU consists of two gates: a reset gate and an update gate. These gates control the flow of information and help the network retain long-term dependencies while mitigating the vanishing gradient problem. The equations governing the GRU are shown in equations 17, 18, 19, and 20.

Update Gate (z_t) determines the amount of past information to carry forward to the future (equation 17).

$$z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \tag{17}$$

Here, σ is the sigmoid activation function, W_z and U_z are weight matrices for the update gate, x_t is the input at time t, h_{t-1} is the hidden state from the previous time step, and b_z is the bias term for the update gate. The reset gate (r_t) determines how much past information can be forgotten (equation 18).

$$r_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \tag{18}$$

Here, W_r and U_r are weight matrices for the reset gate, and b_r is the bias term for the reset gate. Candidate activation ($\tilde{h}_t$) computes the candidate's hidden state, which is a potential update for the hidden state (equation 19).

$$\tilde{h}_t = \tanh(W \cdot x_t + r_t \odot (U \cdot h_{t-1}) + b_h) \quad (19)$$

Here, tanh is the hyperbolic tangent activation function, W and U are weight matrices for the candidate activation, b_h is the bias term for the candidate activation, and $\odot$ denotes the element-wise multiplication. The final hidden state (h_t) at time t is computed by combining the previous hidden state and the candidate hidden state using the update gate (equation 20).

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \quad (20)$$

This equation shows how the update gate controls the balance between the previous hidden state and the candidate hidden state.

In these equations:

x_t = input vector at time step t.
h_{t-1} = hidden state vector at the previous time step.
h_t = hidden state vector at the current time step.
z_t = update gate vector.
r_t = reset gate vector.
$\tilde{h}_t$ = candidate hidden state vector.
W_z, W_r, W = input weight matrices.
U_z, U_r, U = recurrent weight matrices.
b_z, b_r, b_h = bias vectors.
σ = the sigmoid function.
tanh = the hyperbolic tangent function.
$\odot$ = the Hadamard (element-wise) product.

3.2.8 Proposed MCDFN model

We introduce a novel neural network architecture termed the MCDFN, designed to analyze time series data comprehensively. The name "Multi-Channel" signifies incorporating multiple parallel channels within the network, each processing the same input data through distinct pathways. These channels leverage diverse components such as CNN, LSTM, and GRU to extract different representations or features from the input time series. The term "Data Fusion" emphasizes the integrated nature of the model, where outputs from the individual channels are fused to capture complementary information and enhance the overall understanding of the data. The "Network" aspect highlights the NN architecture, showcasing the interconnected layers of neurons that facilitate information processing and prediction. By accumulating an internal state over a 30-day period, the model can generate a single prediction for the subsequent 30 days. The novelty of the MCDFN lies in its ability to combine these different architectures within a unified framework, enhancing its ability to capture complex patterns in time series data.

The MCDFN model comprises four main components: a CNN, a BiLSTM network, a Bidirectional GRU (BiGRU) network, and a multi-layer LSTM network. These components are strategically combined to capture both spatial and temporal dependencies in the data. Figure 8 provides a detailed breakdown of the layers and parameters of the MCDFN model.

1. **Input Layer**: The model accepts input sequences of shape (30, 10), where 30 is the number of time steps to forecast, and 10 represents the number of features.
2. **CNN Branch**: Two Conv1D layers with a hyperparameter-defined number of filters and kernel sizes, both using ReLU activation, process the input sequences. This branch captures local patterns and trends in the data. Following the convolutional layers, a MaxPooling1D layer reduces the spatial dimensions, and a Dense layer with 128 units further processes the data. The output of this branch is flattened to create a feature vector.
3. **BiLSTM Branch**: A Bidirectional LSTM layer with 192 units processes the input sequences, capturing temporal dependencies in both forward and backward directions. A Dropout layer with a rate of 0.2 is applied to prevent overfitting. The output is flattened to create a feature vector.
4. **BiGRU Branch**: A BiGRU layer processes the input sequences with a hyperparameter-defined number of units. GRUs are particularly effective for capturing long-term dependencies in time series data. A Dropout layer with a rate of 0.4 is applied to reduce overfitting further. The output is flattened to create a feature vector.
5. **Multi-Layer LSTM Branch**: This branch consists of two LSTM layers, each with a hyperparameter-defined number of units. The first LSTM layer processes the input sequences, and the second LSTM layer further refines the temporal features. Dropout layers with a rate of 0.2 are applied between the LSTM layers. The output is flattened to create a feature vector.
6. **Concatenation and Dense Layer**: The outputs of the CNN, BiLSTM, BiGRU, and multi-layer LSTM components are concatenated into a single tensor, combining their respective learned representations. A fully connected Dense layer with 30 units processes the concatenated tensor to produce the final output. The activation function for this layer is linear and appropriate for regression tasks. The final output is reshaped to match the desired output shape (30, 1).

We used hyperparameter tuning to optimize the number of filters in the CNN layers, kernel sizes, and units in the LSTM and GRU layers. After 90 trials of hyperparameter search, the optimal CNN filters, kernel size, and LSTM/GRU units were found to be 352, 1, and 64, respectively, in the 16th

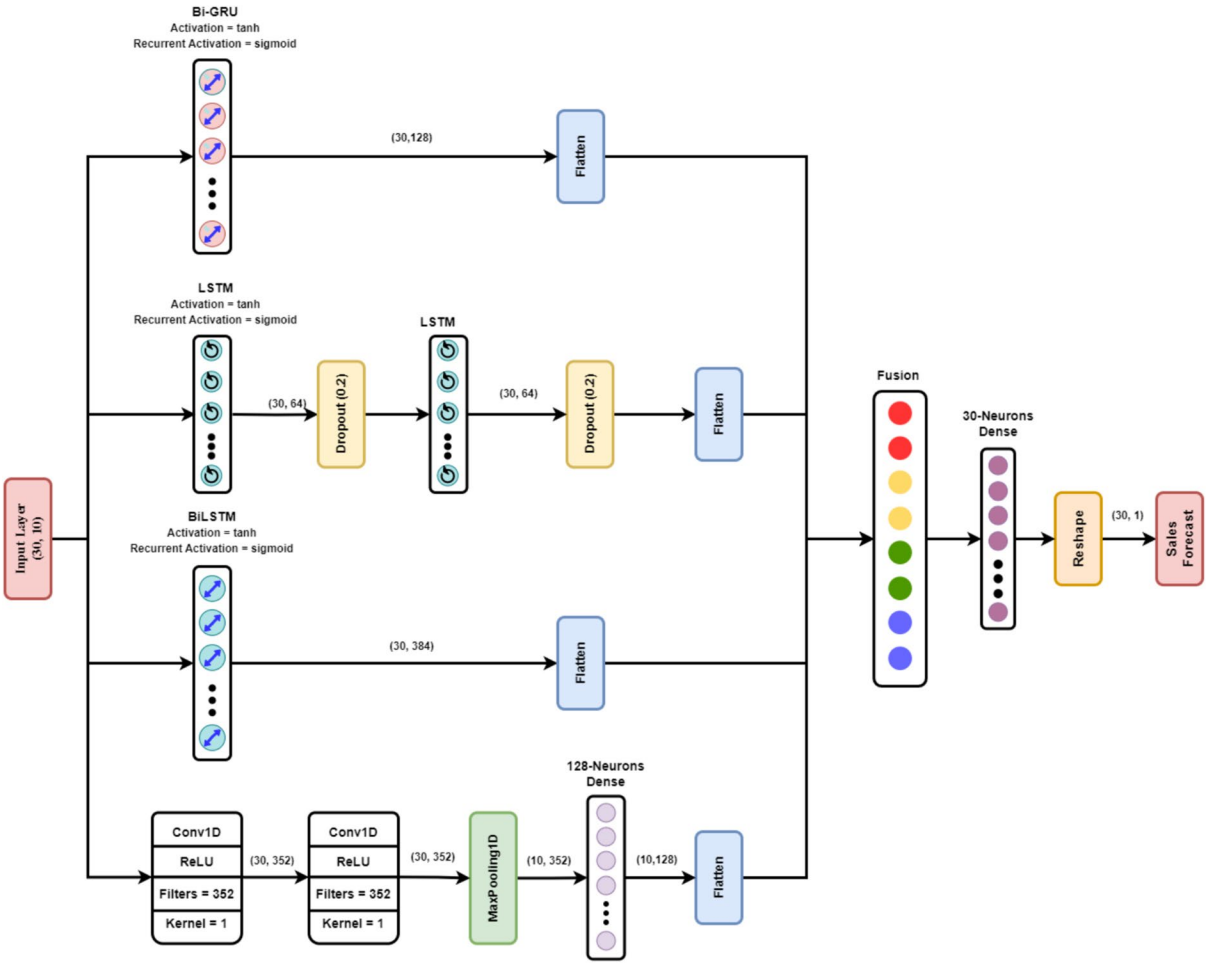


Fig. 8 Model architecture of the proposed MCDFN

trial. The model with these hyperparameters achieved the lowest validation MSE of 23.29 during trials and the tuning process.

3.3 Training and tuning method

To predict demand, the models were trained with the evaluation metrics described in the "Evaluation Criteria" subsection. The generated windows are applied in batches to the input of the models according to the required shapes. The models take features of the preceding 30 days as inputs and predict the demand for the next 30 days.

We employ the Keras Tuner library, a powerful tool for hyperparameter tuning to optimize model performance and parameter selection. Specifically, we instantiate a Hyperband tuner object to efficiently explore the hyperparameter space and identify optimal configurations for our hybrid model architecture. The Hyperband algorithm, a variant of the popular random search approach, orchestrates a series of successive halving iterations, progressively allocating resources to the most promising configurations while discarding less performant ones. By iteratively leveraging the performance feedback obtained during each training epoch, Hyperband dynamically allocates computational resources, thereby maximizing the efficiency of the hyperparameter search process. Several key parameters govern the tuning process when configuring the Hyperband tuner. The objective parameter defines the optimization criterion, with 'val_mean_squared_error' serving as our chosen metric for assessing model performance on the validation dataset. Additionally, we constrain the tuning process to a maximum of 10 epochs per configuration. We set the number of Hyperband iterations to 5, striking a balance between exploration and exploitation of the hyperparameter space.

The windowed input and output pair were fed to the networks for training. For training, the batch size was set to 32, and the epoch was set to 50. In the training process, the loss metric was MSE. The choice is justified as it is one of the most popular primary performance metrics for regression and forecasting tasks [9].

We optimized the models using the prominent stochastic optimization technique Adam. It is a method for solving stochastic optimization problems using adaptive lower-order moment estimations. Adam is computationally efficient and requires less memory to run [30]. Keras 'EarlyStopping' callback function ends the training process if the training metrics do not improve over a patient of one hundred epochs. This can help reduce the program from being overfit.

Apart from that, to prevent the models from overfitting, we employed dropout regularization in the networks. Dropout techniques sample from an exponential number of networks and activate/deactivate the neurons to suppress overfitting and improve the network's performance [51]. Figure 9 exhibits the ideal loss curve for MCDFN as the training loss exhibits a steep decline, reflecting the model's rapid learning of data patterns. Both training and validation losses converged and stabilized, indicating minimal overfitting and robust generalization to new data. The loss curve remained smooth throughout the training process, signifying stable learning dynamics and well-tuned hyperparameters.

4 Results

For performance evaluation, our architecture utilized Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). These metrics were chosen due to their widespread use in demand forecasting and their ability to provide complementary insights into model performance. Below, we justify the choice of these metrics and discuss their relevance to SC forecasting.

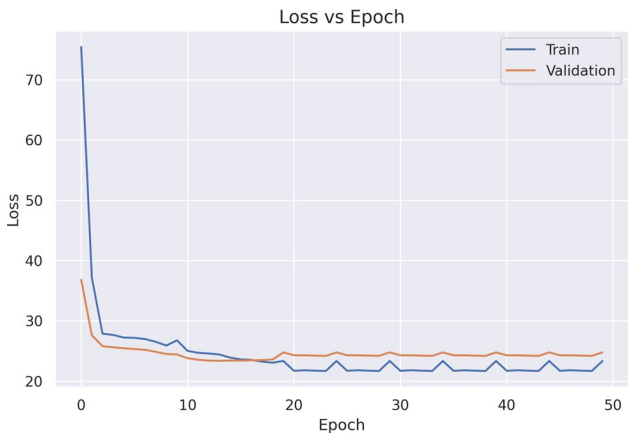


Fig. 9 Loss per epoch curve of the proposed MCDFN model

4.1 Evaluation criteria

For performance evaluation, our architecture used Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

4.1.1 Mean absolute error (MAE)

MAE estimates the mean deviation of the observed value from actual values. For example, there are n data points in a sample, $\hat{y}$, and y represent the vector of prediction values and the vector of True values, respectively. Therefore, the equation for MAE is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left(|\hat{y}_l - y_i|\right) \tag{21}$$

MAE is particularly useful for SC forecasting because it provides an intuitive measure of average error magnitude without penalizing larger deviations disproportionately. This makes it suitable for evaluating models where consistent accuracy across all predictions is critical, such as inventory management and order fulfillment.

4.1.2 Mean squared error (MSE)

MSE estimates the mean of the squared deviation of estimated values from the true values. Suppose there are n data points in a sample. We have generated the prediction vector for all the data of this sample. For this case, the MSE is computed as the equation given below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_t - \hat{y}_t\right)^2 \tag{22}$$

MSE is sensitive to outliers due to the squaring of errors, making it particularly useful for identifying models that perform poorly on extreme demand spikes or drops. In SC contexts, minimizing MSE ensures that the model avoids large prediction errors, which can lead to stockouts or overstocking.

4.1.3 Root mean squared error (RMSE)

RMSE is the squared root of MSE. It is more comprehensive as it directly represents the actual value rather than a processed version like squared or absolute value.

$$RMSE = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}\left(y_t - \hat{y}_t\right)^2} \tag{23}$$

RMSE is widely used in forecasting because it emphasizes larger errors more than MAE, providing a balanced view of overall model performance. In SC applications, RMSE helps prioritize models that minimize significant deviations, which are often costly in terms of lost sales or excess inventory.

4.1.4 Mean absolute percentage error (MAPE)

MAPE exhibits a more general idea about forecasting accuracy as it is a percentage. It can be determined using the following equation:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_t - \hat{y}_t|}{y_t} \tag{24}$$

MAPE is particularly valuable in SC forecasting because it provides a relative error measure, making it easier to interpret and compare across different datasets or products. However, it is sensitive to zero or near-zero actual values, which can occur in intermittent demand scenarios.

4.1.5 Theil's U statistic

Theil's U statistic is a relative accuracy measure used to evaluate the performance of a predictive model against a naive benchmark. It is particularly useful in regression and time-series forecasting, where assessing the model's effectiveness relative to simple heuristics is essential. Theil's U statistic is computed as follows:

$$U = \frac{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}}{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - y_{t-1})^2}} \tag{25}$$

where y_t represents the actual observed value at time t, $\hat{y}_t$ is the predicted value at time t, y_{t-1} is the previous observed value (naive forecast), n is the total number of observations. The numerator represents the RMSE of the benchmarked model, while the denominator represents the RMSE of a naive forecast, typically assuming that the best prediction for the next step is the previously observed value. $U = 1$ indicates that the predictive model performs equivalently to the naive benchmark. $U > 1$ suggests that the model performs worse than the naive forecast. $U < 1$ signifies that the predictive model outperforms the naive approach, demonstrating better predictive accuracy. $U = 0$ represents a perfectly accurate model with no prediction error.

4.2 Comparative analysis of results

Table 2 presents the comparative evaluation of various models, including BiLSTM, CNN, RNN, Stacked LSTM, Vanilla LSTM, FCN, GRU, and Proposed MCDFN. A comparative analysis of these results provides valuable insights into the effectiveness of each model for the given task.

Among the models evaluated, the CNN-LSTM-GRU architecture, proposed as MCDFN, demonstrated promising performance (see Figure 10). It achieved the lowest test loss and competitive scores across all evaluation metrics compared to other models. The model exhibited a balanced trade-off between accuracy and complexity, leveraging the strengths of CNNs, BiLSTM, stacked LSTMs, and GRUs.

Table 2 Performance evaluation of the benchmarked models on validation and test datasets. **Bold** indicates the best test results

Model	Sample of data	Loss	MSE	RMSE	MAE	MAPE (%)
BiLSTM	Validation	23.4374	23.0871	4.8049	3.9128	21.1238
	Test	24.5546	24.4077	4.9404	4.0614	20.4474
CNN	Validation	26.3405	26.0675	5.1056	4.1139	22.9362
	Test	29.5089	29.1954	5.4033	4.2506	23.0131
RNN	Validation	23.6822	23.2973	4.8267	3.9055	21.4250
	Test	23.9603	23.7765	4.8761	4.0431	20.8789
Stacked LSTM	Validation	23.5703	23.2025	4.8169	3.9023	21.0031
	Test	25.6610	25.4365	5.0435	4.1144	20.5473
Vanilla LSTM	Validation	42.0955	39.9834	6.3232	5.0423	27.3639
	Test	43.5302	43.6123	6.6040	5.2788	25.5910
FCN	Validation	25.6774	25.2317	5.0231	4.0225	21.5819
	Test	28.1411	27.8934	5.2814	4.1525	20.8245
GRU	Validation	23.6932	23.2377	4.8206	3.9244	21.7513
	Test	23.9825	23.7848	4.8770	4.0189	20.7770
Proposed MCDFN	Validation	23.3612	22.9416	4.7897	3.9065	21.3768
	Test	**23.7287**	**23.5738**	**4.8553**	**3.9991**	**20.1575**

On the other hand, the Vanilla LSTM model, despite being a fundamental architecture, struggled with the test data, yielding the highest test loss and relatively inferior performance in terms of MSE, RMSE, MAE, and MAPE. This may suggest limitations in capturing the temporal dependencies and patterns in the data, leading to suboptimal forecasting accuracy. The BiLSTM, RNN, and GRU models performed comparably, showcasing moderate performance in test evaluation metrics. While these models demonstrated the ability to capture temporal dependencies to some extent, they fell short of achieving the same level of accuracy as the proposed MCDFN architecture. The CNN model exhibited inferior performance compared to other DL architectures, particularly regarding RMSE and MAPE, indicating challenges in effectively leveraging spatial features for demand forecasting tasks. The FCN model, while offering simplicity in architecture, lagged behind in forecasting accuracy compared to more complex DL models, highlighting the importance of leveraging sequential and temporal information for accurate demand predictions (Table 3).

MCDFN achieves competitive performance with Theil's U statistic of 0.1181, demonstrating its robustness in predictive accuracy, as shown in Table 4. While it has a relatively higher parameter count (1,123,358) and longer training time (≈35,100 ms), it significantly outperforms all baseline models in inference efficiency with an inference time of just ≈32 ms. In contrast, BiLSTM and GRU exhibit slightly lower Theil's U values (0.1097 and 0.1082, respectively) but require higher inference times (≈122 ms and ≈474 ms). Stacked LSTM, despite its deeper architecture, shows the highest Theil's U statistic (0.1331) along with the longest training time (≈15,210 ms), indicating potential overfitting.

Table 3 Ablation study results for one or more modules of MCDFN. **Bold** indicates the best test results

Model	Loss	MSE	RMSE	MAE	MAPE (%)
w/o BiLSTM branch	24.5677	23.7164	5.1299	4.2332	21.0827
w/o CNN branch	24.6199	23.9214	5.5051	4.1633	21.6743
w/o BiGRU branch	24.4913	23.8372	5.0736	4.4519	21.6472
w/o Stacked LSTM branch	25.0792	24.8041	5.1571	4.1067	21.996
w/o BiLSTM and CNN branch	25.1304	24.0481	5.1470	4.148	21.4381
w/o CNN and BiGRU branch	24.5663	24.4179	5.3016	4.0947	21.6999
w/o BiGRU and Stacked LSTM branch	25.1313	24.3707	5.1925	4.0529	21.1524
w/o Stacked LSTM and BiLSTM branch	25.2962	24.5997	5.1558	4.3552	21.2836
w/o CNN and Stacked LSTM branch	24.5836	24.3847	5.3117	4.1446	21.2164
w/o BiLSTM and BIGRU branch	25.4159	24.4595	5.1338	4.0763	21.1825
Proposed MCDFN	**23.7287**	**23.5738**	**4.8553**	**3.9991**	**20.1575**

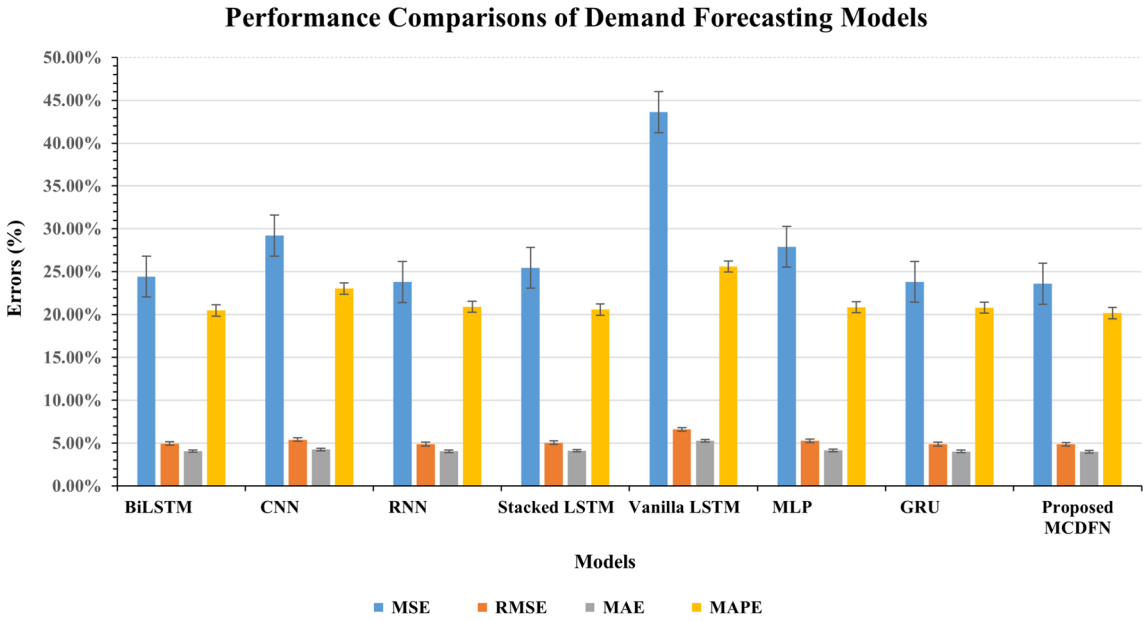


Fig. 10 Performance comparisons of the benchmarked demand forecasting models based on MSE, RMSE, MAE, and MAPE scores

Table 4 Comparative analysis of model performance using Theil's U statistic, number of parameters, and computational efficiency

Model	Theil's U statistic	# Params	Training time (ms)	Inference time (ms)
BiLSTM	0.1097	657,438	≈7488	≈122
CNN	0.1158	191,006	≈2085	≈940
RNN	0.1051	133,022	≈5148	≈364
Stacked LSTM	0.1331	3,631,134	≈15210	≈124
Vanilla LSTM	0.1528	9,57,150	≈1911	≈74
FCN	0.1184	6145	≈7800	≈386
GRU	0.1082	290,334	≈5440	≈474
Proposed MCDFN	0.1181	1,123,358	≈35100	≈32

The proposed MCDFN architecture, integrating CNN, LSTM, and GRU layers, emerged as the top-performing model for demand forecasting, offering superior accuracy and robustness compared to other DL architectures evaluated in this study. The key results are summarized as follows:

- The MCDFN architecture outperformed other models in demand forecasting, demonstrating the lowest test loss and competitive scores across evaluation metrics.
- The Vanilla LSTM model exhibited the poorest performance, indicating limitations in capturing temporal dependencies for accurate demand predictions.
- Stacked LSTM performed better than Vanilla LSTM, suggesting additional LSTM layers improving forecasting accuracies by a greater margin.
- RNN emerged as the second-best performing model in terms of MSE, RMSE, and MAE, considering its architectural simplicity.
- Models incorporating a combination of CNN, LSTM, and GRU layers showed improved performance compared to individual architectures, emphasizing the importance of leveraging diverse NN structures for effective demand forecasting.

4.3 Ablation studies

The ablation study presented in Table 3 compares the performance of various model configurations, each missing one or more components (such as BiLSTM, CNN, BiGRU, and Stacked LSTM branches), against the fully configured proposed model, MCDFN. The proposed MCDFN model achieves the lowest values across all metrics, indicating superior performance.

By removing individual or paired branches, performance generally deteriorates across all metrics. For instance, removing the BiLSTM and CNN branches results in a notable increase in RMSE (5.15%) and MAPE (21.44%). Similarly, removing the Stacked LSTM and BiLSTM branches results in one of the highest loss values (25.30%), indicating that both these branches contribute significantly to the model's accuracy. This analysis highlights the importance of each component within the proposed MCDFN structure, with the complete architecture yielding the most accurate predictions.

4.4 Final output generation

The final output is predicted using the ensemble of models within the MCDFN framework, including CNN, GRU, and LSTM. For visualization, we plotted the predicted sales values alongside the actual sales values for a 30-day period. Figure 11 illustrates the normalized sales data over this period, which were individually input into the CNN, GRU, and LSTM channels of the MCDFN. The figure compares the forecasted sales produced by the MCDFN model against the ground truth sales values, highlighting the model's performance and accuracy in predicting future sales trends.

4.5 Statistical tests between predicted and ground truth values

We performed statistical tests to evaluate the performance of the implemented models using the mean t-statistic and mean p-value derived from the predicted values ($\hat{y}$) and the true values (y). The hypotheses for our tests are defined as follows:

- **Null Hypothesis** ($\mathbf{H_0}$): There is no significant difference between $\hat{y}$ and y.
- **Alternative Hypothesis** ($\mathbf{H_a}$): There is a significant difference between ($\hat{y}$) and y.

We set the significance level (α) at 5%. The mean t-statistic and mean p-value for each model were calculated as follows:

$$t = \frac{\bar{d}}{s/\sqrt{n}} \tag{26}$$

where $\bar{d}$ is the mean difference between $\hat{y}$ and y, s is the standard deviation of the differences, and n is the number of observations.

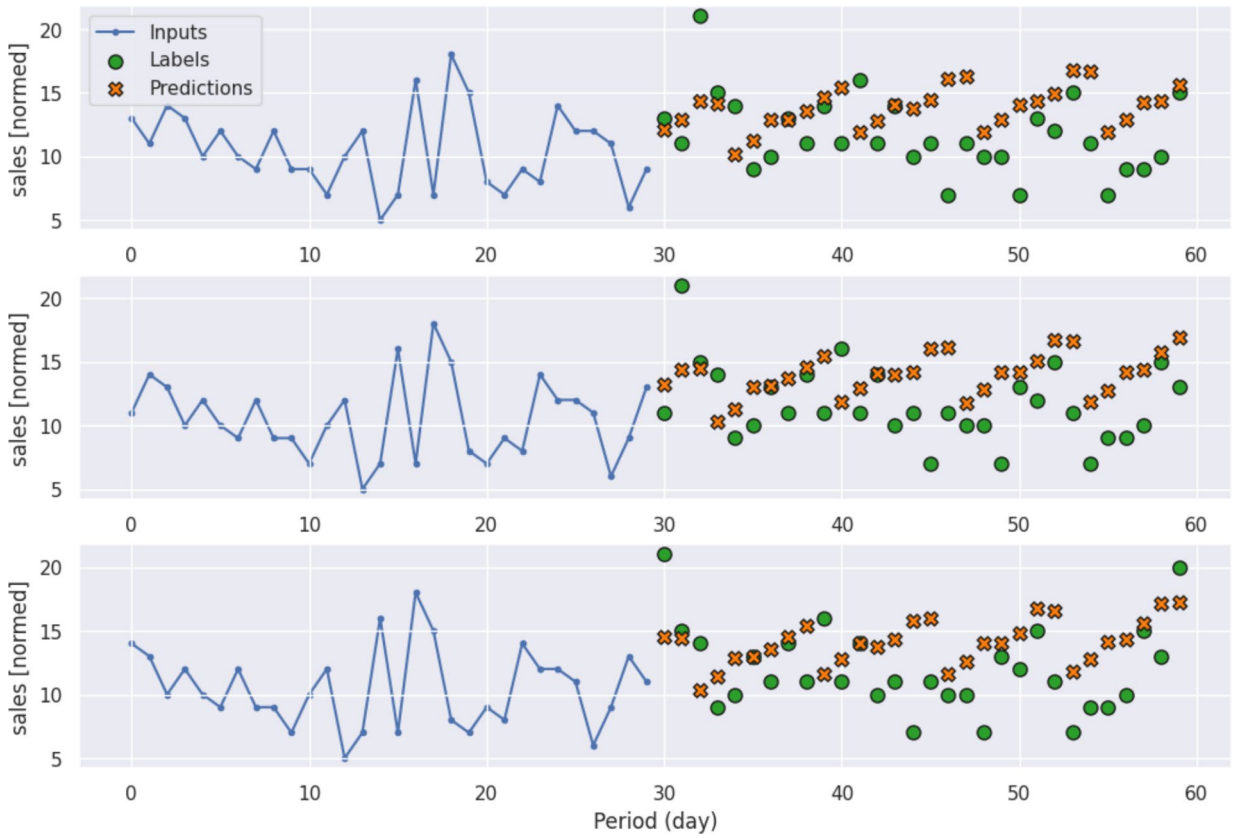


Fig. 11 Month ahead predicted and actual sales of the proposed MCDFN model

Table 5 Summary of the mean t-statistic and mean p-value for the benchmarked models

Model	Mean t-statistic	Mean p-value
BiLSTM	−1.4621117	0.20666657
CNN	−0.17138764	0.45867905
RNN	−0.6514868	0.52714866
Stacked LSTM	−1.6302485	0.23526402
Vanilla LSTM	−4.2031035	0.0075615253
FCN	−2.1991768	0.044341218
GRU	−0.6693428	0.524219
MCDFN (ours)	−1.2990302	0.20307085

Table 5 summarizes each model's mean t-statistic and mean p-value. The statistical results indicate that the Vanilla LSTM model shows a significant difference between $\hat{y}$ and y, with a p-value less than 5%, suggesting that the model's predictions are considerably inaccurate or not robust. Conversely, other models, including MCDFN, have higher p-values ($\geq$0.05), indicating no significant difference between $\hat{y}$ and y, thus suggesting better accuracy and robustness in their predictions. This highlights the importance of model selection in ensuring reliable and precise predictions in NN applications.

4.6 Statistical significance testing of model performance

In this section, we present a comprehensive statistical analysis of the performance of different models in comparison to the proposed MCDFN model. The statistical tests are performed to determine if there are significant differences in MSE, MAE, and MAPE between the MCDFN model and other baseline models. For each model comparison, the null hypothesis (H_0) and the alternative hypothesis (H_1) are formulated as follows:

- H_0: There is no significant difference in the performance metric (MSE, MAE, or MAPE) between the MCDFN and baseline models.
- H_1: There is a significant difference in the performance metric (MSE, MAE, or MAPE) between the MCDFN and baseline models.

The p-value is calculated using the t-distribution to determine the probability of observing the t-statistic under the null hypothesis. The significance level (α) is set to 0.05. Table 6 shows the t-values and p-values for the MSE, MAE, and MAPE comparisons between the MCDFN model and various baseline models. The p-values in Table 6 indicate that all comparisons between the MCDFN model and other baseline models are statistically significant at the $\alpha = 0.05$ level for all three metrics (MSE, MAE, and MAPE). This suggests that the performance of the MCDFN model is significantly different from the baseline models across all considered metrics. The MCDFN model consistently shows significant improvements over the other models regarding MSE, MAE, and MAPE. In particular, Vanilla LSTM and CNN models show the highest t-values, indicating large performance differences compared to the MCDFN model.

4.7 XAI interpretation

To address the "black box" nature of the MCDFN model, we employed two widely recognized XAI techniques: ShapTime and PFI. These methods provide insights into the model's decision-making process, enhancing transparency and trust in its predictions.

4.7.1 ShapTime

We utilized ShapTime with our proposed model and obtained the explanation outcomes. In this instance, we configured the number of super-times, denoted as n to 10, resulting in the following theoretical adjusted time series model:

$$y_t = \phi_{t_0} t_0 + \phi_{t_1} t_1 + \cdots + \phi_{t_9} t_9 + \omega_t \tag{27}$$

Where: - y_t: Predicted value at time t, - ϕ_{t_i}: SHAP value for super-time t_i, - ω_t: Residual error term.

Once the training phase concluded, we employed ShapTime to elucidate our model, attributing the forecasting outcomes to each super-time. Subsequently, we visualized these attributions through a heatmap. Our analysis revealed that t_1 holds paramount significance within our model, while t_6 carries minimal importance.

After evaluating the ShapTime explanations against predetermined criteria, we illustrated an evaluation example in Figure 12. Consistently, ShapTime's explanations demonstrate stability across various model types, a crucial factor for engendering trust among users across diverse industries. The potential fluctuation of explanation results during usage could provoke user skepticism or abandonment, underscoring the importance of this stability.

Theoretically, if we were to perturb the training data based on the explanation outcomes-for instance, substituting critical super-times with less essential ones-the model's predictive performance would be markedly deteriorating. In our example, this point is illustrated by replacing the crucial t_1 with the least influential t_6 and t_8 with t_3. This evaluation approach aligns with the conventional practice in the XAI domain, commonly called sensitivity analysis.

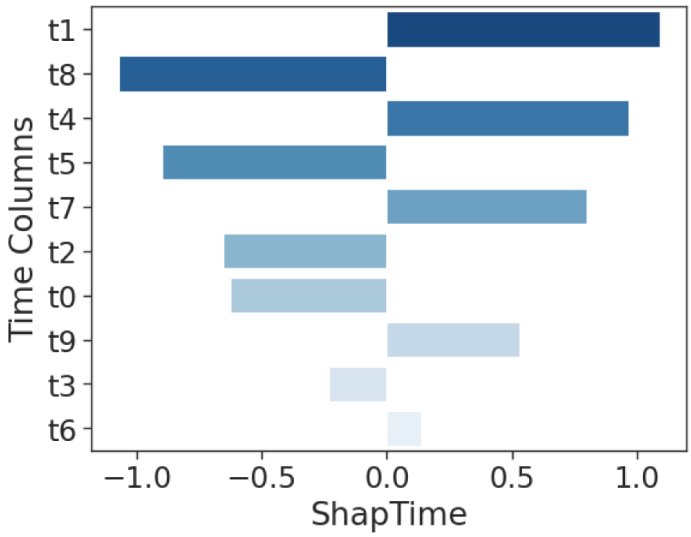
Figure 12a displays the ShapTime barplot, illustrating the average SHAP values for each feature in the MCDFN model. The barplot highlights the overall impact of each feature on the model's predictions, with longer bars indicating higher importance and contribution to the MCDFN model's output. Figure 12b shows the ShapTime heatmap, providing a detailed view of the SHAP values for each feature across different time steps. This heatmap enables the visualization of how the contribution of each feature varies over time, offering insights into temporal patterns and feature importance dynamics in MCDFN.

4.7.2 Permutation feature importance

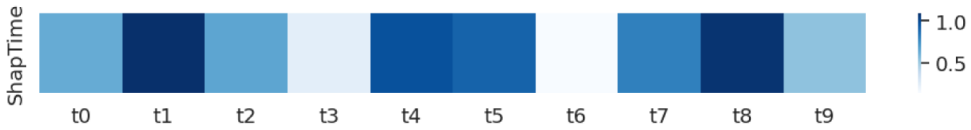
The permutation feature importance (PFI) scores indicate the degree to which each feature influences the predictive performance of MCDFN. Features with higher importance scores contribute more significantly to the accuracy of the demand forecasts generated by the model. The PFI score for a feature f is calculated as follows:

Table 6 10 fold cross-validated paired t-test between MCDFN and the other models

Model	t-value (MSE)	p-value (MSE)	t-value (MAE)	p-value (MAE)	t-value (MAPE)	p-value (MAPE)
BiLSTM	35.27276	5.85E-11	10.49892	2.38E-06	38.9985	2.38E-11
CNN	102.333	4.12E-15	28.71858	3.67E-10	263.1605	8.41E-19
RNN	11.2214	1.36E-06	7.404645	4.08E-05	49.9998	2.57E-12
Stacked LSTM	82.47896	2.87E-14	12.22952	6.55E-07	30.64947	2.05E-10
Vanilla LSTM	1024.278	4.10E-24	100.5583	4.83E-15	505.497	2.36E-21
FCN	153.3214	1.09E-16	25.72245	9.77E-10	71.03343	1.10E-13
GRU	6.030728	0.000195	8.243713	1.74E-05	50.1296	2.51E-12



(a) ShapTime barplot of feature importance in descending order of their contribution to the MCDFN's predictions.



(b) ShapTime heatmap showing the variation of feature contribution over time in MCDFN.

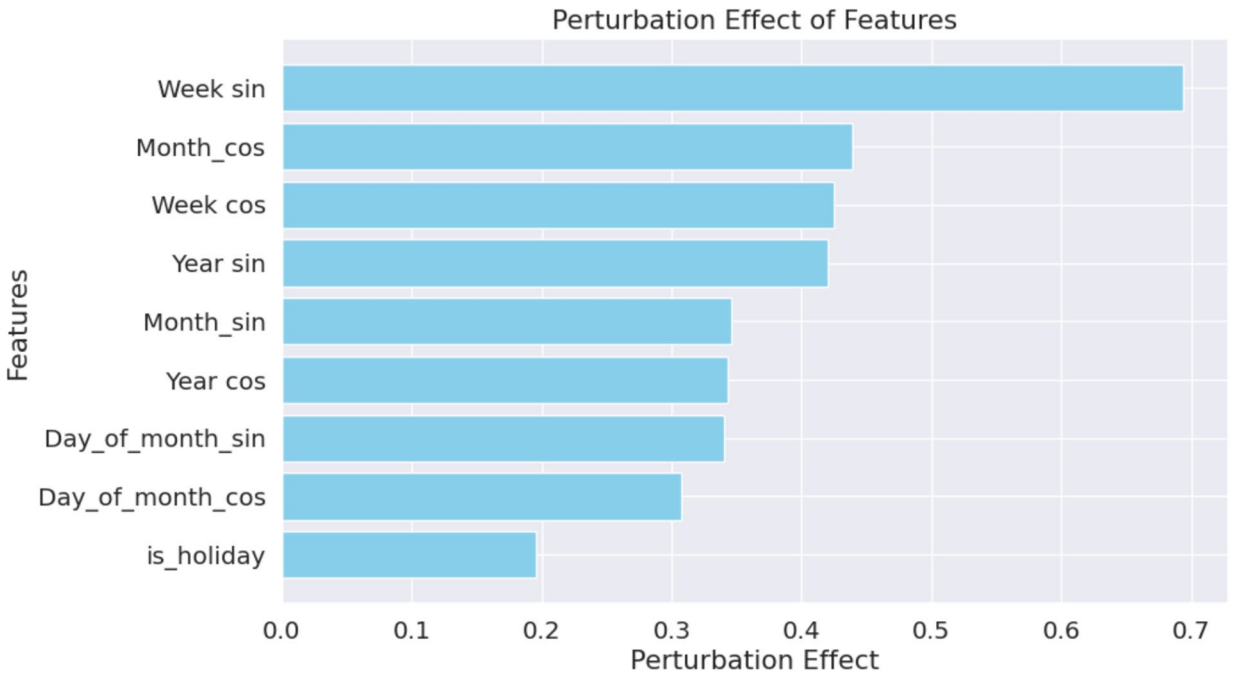


Fig. 12 Evaluation of Shap-Time explanation results on the proposed MCDFN

Fig. 13 Permutation feature importance analysis on the proposed MCDFN

$$\text{PFI}(f) = \frac{\text{Performance}_{\text{original}} - \text{Performance}_{\text{permuted}}}{\text{Performance}_{\text{original}}} \quad (28)$$

where, $\text{Performance}_{\text{original}}$: Model performance metric (e.g., MSE, MAE) on the original dataset, $\text{Performance}_{\text{permuted}}$: Model performance metric after permuting the values of feature f.

Figure 13 presents the PFI analysis for the MCDFN model. The results reveal that cyclic temporal features such as the "Week_sin" feature emerge as the most influential predictor, followed by "Month_cos" and "Week_cos," indicating the pronounced impact of weekly and monthly cyclical patterns on demand fluctuations. Moreover, the temporal features related to year and month exhibit substantial importance, underscoring the seasonal trends inherent in demand dynamics.

Interestingly, the presence of holidays, as captured by the binary feature "is_holiday", also holds notable importance, albeit to a lesser extent than the cyclic temporal features. This suggests that holidays exert a discernible influence on demand patterns, albeit not as pronounced as the cyclic temporal factors.

5 Discussion

This research underscores two pivotal ideas: firstly, the development of MCDFN for integrating multiple data modalities in demand forecasting, and secondly, the significant performance improvements achieved by employing this advanced DL architecture. Our findings reveal the economic value of utilizing MCDFN, showcasing substantial enhancements in SC efficiency and aligning with previous studies that highlight the practicality of these methods for SC planners. This research supports the increasing focus on AI to bolster organizational data analytics capabilities [59]. By combining statistical and DL-based approaches, our analysis mitigates the excessive variance often seen in DL methods and the bias inherent in traditional statistical methods [37].

The MCDFN can seamlessly integrate into modern SC management platforms to enhance demand forecasting capabilities. Many enterprises rely on Enterprise Resource Planning (ERP) systems, inventory management tools, and decision support systems to optimize SC operations. MCDFN can be deployed as a plug-and-play predictive module that integrates with widely used ERP solutions such as SAP, Oracle SCM, and Microsoft Dynamics, as well as inventory management tools like NetSuite and Fishbowl Inventory. This integration can be achieved through API-based deployment, allowing MCDFN to retrieve historical sales, real-time transactions, and inventory data to enhance forecasting. Alternatively, MCDFN can be incorporated into a data pipeline, enabling a seamless flow from data ingestion to forecasting to automated stock adjustments. Forecast outputs can be automatically fed into inventory management modules to adjust stock levels dynamically, reducing manual intervention in procurement decisions. MCDFN's explainability features, such as ShapTime and Permutation Feature Importance, also allow SC managers to interpret key demand drivers, making inventory restocking decisions more transparent.

Through extensive benchmarking and panel data analysis from a retail scenario, we have shown that MCDFN outperforms traditional time series and linear models by capturing complex relationships within the data. This finding is consistent with existing research indicating the superior performance of DL methods in homogeneous datasets [15, 22]. Additionally, our research addresses inconsistent evidence in AI-based demand forecasting by providing robust empirical validation of its effectiveness in homogeneous datasets [37]. This study also responds to calls for testing hybrid models in demand forecasting and evaluating their impact on performance outcomes. We assessed three performance metrics-one financial and two non-financial. Our results indicate that the hybrid DL-based MCDFN approach significantly improves inventory performance, which is crucial in capital-intensive industries such as retail. Enhanced inventory efficiency, facilitated by accurate demand prediction, translates into better return on assets and profitability. Similar to findings in the retail industry for perishable goods [24], our research demonstrates significant performance improvements when DL methods are employed over traditional approaches.

Demand forecasting in this study was performed at an aggregate product family level, capturing broader demand patterns before refining insights for operational use. Our results confirm the importance of macro-economic factors in aggregate demand prediction, with the MCDFN model effectively capturing and incorporating complex, non-linear relationships among multiple predictive variables. Improved forecasting accuracy offers significant economic benefits, including reduced stockouts and overstocking, lower inventory holding costs, reduced obsolescence risk, and improved working capital efficiency. These benefits extend to better production scheduling, optimized logistics, and reduced emergency procurement costs, all of which contribute to lower total SC costs and improved service levels. Accurate aggregate forecasts can also be translated into more precise SKU-level predictions, potentially reducing inefficiencies and the bullwhip effect prevalent in non-seasonal demand industries [13].

Effective demand forecasting significantly enhances SC efficiency, but fully eliminating the bullwhip effect requires a system-wide perspective and improved collaboration among

SC participants [23]. While SC integration remains challenging, leveraging big data analytics (BDA) capabilities can yield favorable outcomes, particularly for upstream firms with limited visibility into end-consumer demand. Improved demand visibility also fosters collaboration, encouraging joint planning and information sharing, which is critical to long-term SC resilience. Despite adopting sophisticated analytical methods, many firms continue to rely on judgmental adjustments in demand forecasting [11, 39, 48]. Our findings confirm the ongoing importance of expert judgment in refining model outputs. As highlighted by Sanders and Manrodt, judgmental adjustments remain common even among practitioners using model-based forecasts, underscoring the enduring value of domain expertise alongside advanced analytics [48].

It is challenging to directly compare the performance of our proposed system with existing studies due to the absence of research incorporating a similar combination of deep learning techniques, decision integration strategies, and diverse learning algorithms in the demand forecasting process. Another limitation is the lack of real-world datasets like ours, making it difficult to benchmark against state-of-the-art studies. While this study presents the results of our proposed system, we also report findings from several related research efforts within the demand forecasting domain. A greedy aggregation-decomposition method was used to solve an intermittent demand forecasting problem for a fashion retailer in Singapore. The approach combined aggregated and detailed forecasts, improving accuracy compared to standard methods. The study achieved a 5.9% MAPE using a small dataset, showing that the method works well for handling irregular demand [33]. A deep learning-based demand forecasting model was developed and tested using real-world retail data from the SOK Market in Turkey. The study combined deep learning, support vector regression, and multiple time-series models with a novel decision integration strategy. The approach improved forecast accuracy compared to single models. The proposed system achieved a MAPE of 24.7%, significantly reducing prediction errors. The research highlights the benefits of integrating different forecasting techniques for better accuracy [29]. Both studies' datasets are not publicly available due to confidentiality agreements, making it difficult to validate the MCDFN model on different industries. Without access to diverse real-world datasets, generalizing the model's effectiveness beyond the original domain remains challenging.

6 Conclusions

This research introduced the MCDFN, a novel hybrid DL architecture that integrates multiple data modalities for enhanced demand forecasting accuracy. By leveraging the strengths of CNN, LSTM, and GRU, MCDFN captures both spatial and temporal patterns in the data, resulting in superior forecasting performance compared to existing models. Our extensive benchmarking against seven different DL models demonstrated the superiority of MCDFN in terms of accuracy and robustness across various evaluation metrics, including RMSE, MSE, MAPE, and MAE.

We validated the robustness and reliability of MCDFN through a 10-fold cross-validated statistical paired t-test for each benchmarked model's evaluation metrics, showing that MCDFN significantly outperforms other models at a 5% significance level. Additionally, we performed a paired t-test for each model using a p-value of 5%, demonstrating no significant difference between the actual and predicted values of MCDFN. To address the "black box" nature of MCDFN, we employed XAI techniques such as ShapTime and PFI, providing insights into the model's decision-making process.

Despite the promising results, our study has several limitations. First, the computational complexity of MCDFN is higher than traditional models, which may limit its applicability in real-time forecasting scenarios or environments with limited computational resources. Second, the performance of MCDFN heavily relies on the quality and quantity of the input data, making it sensitive to noisy or incomplete datasets. Third, while XAI techniques provide some level of interpretability, the model's complexity can still challenge stakeholders to understand the underlying decision-making processes fully. Fourth, our study considered a single product family and a retail shop, potentially introducing selective biases due to geographical positions and other demographic variables.

Future research could address these limitations by exploring several avenues. Conducting a sensitivity analysis of window size to assess its impact on model performance could help fine-tune MCDFN for different forecasting scenarios and improve its adaptability. Implementing advanced data augmentation techniques and robust preprocessing pipelines would enhance the model's ability to handle noisy or incomplete datasets, ensuring more reliable and stable predictions. Since most relevant studies' datasets are confidential and not publicly available, making it difficult to validate MCDFN across different industries, future work should focus on obtaining diverse datasets or using synthetic data to evaluate generalizability. Combining MCDFN with other state-of-the-art techniques, such as attention mechanisms and transformers, could improve its predictive capabilities and interoperability. Exploring the integration of MCDFN with optimization techniques like Genetic Algorithms, Particle Swarm Optimization, or Mixed-Integer Programming can enhance SC decision-making.

Additionally, reinforcement learning could be applied to dynamically adjust demand forecasts in response to real-time changes, improving inventory control and order fulfillment. Comparing performance with generalized DL forecasting methods developed by industry experts, such as N-BEATS (ElementAI) and DeepAR (Amazon), can

benchmark MCDFN's effectiveness. Investigating the scalability of MCDFN in distributed computing environments and deploying it in real-time systems can demonstrate its practical applicability. Extending MCDFN to other domains, such as healthcare or finance, can validate its versatility and generalizability across different types of time series data. Identifying auxiliary variables that affect product demand and studying their influence on forecasting performance, along with integrating external factors like macroeconomic indicators, weather patterns, and social media trends, could further enhance demand forecasting accuracy.

Author Contributions Md Abrar Jahin: Writing — Original draft preparation, Conceptualization, Methodology, Software, Formal analysis, and Visualization. Asef Shahriar: Writing — Original draft preparation, Conceptualization, Methodology, Software, and Data curation. Md Al Amin: Supervision, Writing — Review & Editing.

Funding Open Access funding provided by the Qatar National Library. The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability The historical sales dataset used in this research has been made publicly available on Mendeley Data. Due to the proprietary nature of the dataset and confidentiality agreements with the retailer, the raw sales data cannot be publicly shared. However, the processed data that is anonymized and aggregated to ensure the retailer's confidentiality, along with any supplementary materials that support the findings of this study, can be accessed via the following link: https://data.mendeley.com/datasets/xwmbk7n3c8.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abdollahzadeh B, Khodadadi N, Barshandeh S, Trojovský P, Gharehchopogh FS, El-kenawy E-SM, Abualigah L, Mirjalili S (2024) Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning. Clust Comput 27(4):5235–5283
2. Abolghasemi M, Beh E, Tarr G, Gerlach R (2020) Demand forecasting in supply chain: the impact of demand volatility in the presence of promotion. Comput Indus Eng 142:106380
3. Amarasinghe K, Marino DL, Manic M (2017) Deep neural networks for energy load forecasting. In 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), p 1483–1488
4. Antunes A, Andrade-Campos A, Sardinha-Lourenço A, Oliveira MS (2018) Short-term water demand forecasting using machine learning techniques. J Hydroinf 20(6):1343–1366
5. Arık SO, Pfister T (2021) Interpretable Deep Learning for Time Series Forecasting
6. Awwad M (2018) Big Data Analytics in Supply Chain: A Literature Review. In 3rd North American International Conference on Industrial Engineering and Operations Management, IEOM Society
7. Badr H, Ahmed W (2023) A comprehensive analysis of demand prediction models in supply chain management. Am J Ind Bus Manag 13(12):1353–1376
8. Bandara K, Shi P, Bergmeir C, Hewamalage H, Tran Q, Seaman B (2019) Sales Demand Forecast in E-commerce Using a Long Short-Term Memory Neural Network Methodology. In: Gedeon T, Wong KW, Lee M (eds) Neural Information Processing. Lecture Notes in Computer Science. Cham. Springer International Publishing, pp 462–474
9. Botchkarev A (2019) Performance metrics (Error Measures) in machine learning regression, forecasting and prognostics: properties and typology. IJIKM 14:045–076 arXiv:1809.03006
10. Bouktif S, Fiaz A, Ouni A, Serhani MA (2018) Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with Machine Learning Approaches†. Energies 11(7):1636 (**Number: 7 Publisher: Multidisciplinary Digital Publishing Institute**)
11. Boulaksil Y, Franses PH (2009) Experts' stated behavior. Interfaces 39(2):168–171 (**Publisher: INFORMS**)
12. Böse J-H, Flunkert V, Gasthaus J, Januschowski T, Lange D, Salinas D, Schelter S, Seeger M, Wang Y (2017) Probabilistic demand forecasting at scale. Proc. VLDB Endow. 10(12):1694–1705
13. Cachon GP, Randall T, Schmidt GM (2007) In search of the bullwhip effect. Manufact Service Operat Manage 9(4):457–479 (**Publisher: INFORMS**)
14. Cai M, Pipattanasomporn M, Rahman S (2019) Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. Appl Energy 236:1078–1088
15. Carbonneau R, Laframboise K, Vahidov R (2008) Application of machine learning techniques for supply chain demand forecasting. Eur J Oper Res 184(3):1140–1154
16. Carbonneau R, Vahidov R, Laframboise K (2007) Machine learning-based demand forecasting in supply chains. IJIIT 3(4):40–57 (**Publisher: IGI Global**)
17. Chien C-F, Lin Y-S, Lin S-K (2020) Deep reinforcement learning for selecting demand forecast models to empower Industry 3.5 and an empirical study for a semiconductor component distributor. Int J Prod Res 58(9):2784–2804. https://doi.org/10.1080/00207543.2020.1733125. (**Publisher: Taylor & Francis**)
18. Cox A, Sanderson J, Watson G (2001) supply chains and power regimes: toward an analytic framework for managing extended networks of buyer and supplier relationships. J Supply Chain Manag 37(1):28–35. https://doi.org/10.1111/j.1745-493X.2001.tb00097.x
19. El-kenawy E-SM, Khodadadi N, Mirjalili S, Abdelhamid AA, Eid MM, Ibrahim A (2024) Greylag goose optimization: nature-inspired optimization algorithm. Expert Syst Appl 238:122147
20. El-Kenawy E-SM, Rizk FH, Zaki AM, Mohamed ME, Ibrahim A, Abdelhamid AA, Khodadadi N, Almetwally EM, Eid MM (2024) Football optimization algorithm (FbOA): a novel metaheuristic inspired by team strategy dynamics. J Artif Intell Metaheuristics 8(1):21–38 (**Publisher: American Scientific Publishing Group (ASPG)**)

21. El-Sayed E, Eid MM, Abualigah L (2024) Machine learning in public health forecasting and monitoring the zika virus. Metaheuristic Optimiz Rev 1(2):01–11
22. Ferreira KJ, Lee BHA, Simchi-Levi D (2016) Analytics for an online retailer: demand forecasting and price optimization. Manufact Service Operat Manage 18(1):69–88 (**Publisher: INFORMS**)
23. Gaur V, Giloni A, Seshadri S (2005) Information sharing in a supply chain under ARMA demand. Manage Sci 51(6):961–969 (**Publisher: INFORMS**)
24. Huber J, Stuckenschmidt H (2020) Daily retail demand forecasting using machine learning with emphasis on calendric special days. Int J Forecast 36(4):1420–1438
25. Jahin MA, Shahriar A (2024) Supply Chain Demand Forecasting Dataset of Bangladeshi Retailer. Publisher, Mendeley Data
26. Jahin MA, Shovon MSH, Islam MS, Shin J, Mridha MF, Okuyama Y (2023) QAmplifyNet: pushing the boundaries of supply chain backorder prediction using interpretable hybrid quantum-classical neural network. Sci Rep 13(1):18246 (**Number: 1 Publisher: Nature Publishing Group**)
27. Jahin MA, Shovon MSH, Shin J, Ridoy IA, Mridha MF (2024) Big Data-Supply Chain Management Framework for Forecasting: Data Preprocessing and Machine Learning Techniques. Arch Comput Methods Eng
28. Ke J, Zheng H, Yang H, Chen XM (2017) Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. Trans Res Part C: Emerging Technol 85:591–608
29. Kilimci ZH, Akyuz AO, Uysal M, Akyokus S, Uysal MO, Atak Bulbul B, Ekmis MA (2019) An improved demand forecasting model using deep learning approach and proposed decision integration strategy for supply chain. Complexity 2019(1):9067367
30. Kingma DP, Ba J (2017) Adam: A Method for Stochastic Optimization. arXiv:abs/1412.6980
31. Kogler C, Schimpfhuber S, Eichberger C, Rauch P (2021) Benchmarking procurement cost saving strategies for wood supply chains. Forests 12(8):1086
32. Law R, Li G, Fong DKC, Han X (2019) Tourism demand forecasting: a deep learning approach. Ann Tour Res 75:410–423
33. Li C, Lim A (2018) A greedy aggregation-decomposition method for intermittent demand forecasting in fashion retailing. Eur J Oper Res 269(3):860–869
34. Liao S, Zhou L, Di X, Yuan B, Xiong J (2018) Large-scale short-term urban taxi demand forecasting using deep learning. In: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pages 428–433
35. Lim B, Arık SO, Loeff N, Pfister T (2021) Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int J Forecast 37(4):1748–1764
36. Ma X, Li M, Tong J, Feng X (2023) Deep learning combinatorial models for intelligent supply chain demand forecasting. Biomimetics 8(3):312
37. Makridakis S, Hyndman RJ, Petropoulos F (2020) Forecasting in social settings: the state of the art. Int J Forecast 36(1):15–28
38. Mohsen BM (2023) Impact of artificial intelligence on supply chain management performance. J Serv Sci Manag 16(1):44–58
39. Moritz B, Siemsen E, Kremer M (2014) Judgmental forecasting: cognitive reflection and decision speed. Prod Oper Manag 23(7):1146–1160 (**Publisher: SAGE Publications**)
40. Piccialli F, Giampaolo F, Prezioso E, Camacho D, Acampora G (2021) Artificial intelligence and healthcare: forecasting of medical bookings through multi-source time-series fusion. Inform Fusion 74:1–16
41. Premkumar GP (2000) Interorganization systems and supply chain management: an information processing perspective. Inf Syst Manag 17(3):56–69. https://doi.org/10.1201/1078/43192.17.3.20000601/31241.8. (**Publisher: Taylor & Francis**)
42. Premkumar GP (2001) Interorganizational Systems and Supply Chain Management: An Information Processing Perspective. Auerbach Publications, In Making Supply Chain Management Work
43. Punia S, Nikolopoulos K, Singh SP, Madaan JK, Litsiou K (2020) Deep learning with long short-term memory networks and random forests for demand forecasting in multi-channel retail. Int J Prod Res 58(16):4964–4979
44. Punia S, Singh SP, Madaan JK (2020) A cross-temporal hierarchical framework and deep learning for supply chain forecasting. Comput Indus Eng 149:106796
45. Qiu X, Ren Y, Suganthan PN, Amaratunga GAJ (2017) Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. Appl Soft Comput 54:246–255
46. Sagaert YR, Aghezzaf E-H, Kourentzes N, Desmet B (2018) Temporal big data for tactical sales forecasting in the tire industry. Interfaces 48(2):121–129
47. Salehzadeh R, Tabaeeian RA, Esteki F (2020) Exploring the consequences of judgmental and quantitative forecasting on firms' competitive performance in supply chains. Benchmark Int J 27(5):1717–1737
48. Sanders NR, Manrodt KB (2003) Forecasting software in practice: use, satisfaction, and performance. Interfaces 33(5):90–93 (**Publisher: INFORMS**)
49. Seyedan M, Mafakheri F (2020) Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities. J Big Data 7(1):53
50. Shi H, Xu M, Li R (2018) Deep learning for household load forecasting-a novel pooling deep RNN. IEEE Trans Smart Grid 9(5):5271–5280
51. Srivastava N, Salakhutdinov RR, Hinton GE (2013) Modeling documents with deep boltzmann machines
52. Tang Z, Ge Y (2022) CNN model optimization and intelligent balance model for material demand forecast. Int J Syst Assurance Eng Manage 13(3):978–986
53. Terrada L, Khaili ME, Ouajji H (2022) Demand Forecasting Model using Deep Learning Methods for Supply Chain Management 4.0. International Journal of Advanced Computer Science and Applications
54. Thomopoulos NT (2015) Demand Forecasting for Inventory Control. In: Thomopoulos NT (ed) Demand Forecasting for Inventory Control. Springer International Publishing, Cham, pp 1–10
55. Torres JF, Fernández AM, Troncoso A, Martínez-Álvarez F (2017) Deep Learning-Based Approach for Time Series Forecasting with Application to Electricity Load. In: Ferrández Vicente JM, Álvarez Sánchez JR, de la Paz López F, Toledo Moreo J, Adeli H (eds) Biomedical Applications Based on Natural and Artificial Computing. Lecture Notes in Computer Science. Cham. Springer International Publishing, pp 203–212
56. Wang G, Gunasekaran A, Ngai EWT, Papadopoulos T (2016) Big data analytics in logistics and supply chain management: certain investigations for research and applications. Int J Prod Econ 176:98–110
57. Wang K, Kumar V, Zeng X, Koehl L, Tao X, Chen Y (2019) Development of a textile coding tag for the traceability in textile supply chain by using pattern recognition and robust deep learning. Int J Comput Intell Syst 12(2):713–722
58. Watson G (2001) Subregimes of Power and integrated supply chain management. J Supply Chain Manag 37(1):36–41. https://doi.org/10.1111/j.1745-493X.2001.tb00098.x
59. Wen R, Torkkola K, Narayanaswamy B, Madeka D (2018) A Multi-Horizon Quantile Recurrent Forecaster. arXiv:abs/1711.11053
60. Wu B, Wang L, Wang S, Zeng Y-R (2021) Forecasting the U.S. oil markets based on social media information during the COVID-19 pandemic. Energy 226:120403

61. Yasutomi AY, Enoki H (2020) Localization of inspection device along belt conveyors with multiple branches using deep neural networks. IEEE Robotics Auto Lett 5(2):2921–2928
62. Zhang Y, Li G, Muskat B, Law R (2021) Tourism demand forecasting: a decomposed deep learning approach. J Travel Res 60(5):981–997

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.