Leveraging Spatial-Temporal Graph Neural Networks for Multi-Store Sales Forecasting

Manish*
23b0354@iitb.ac.in
Department of Chemical Engineering, IIT Bombay
Mumbai, Maharashtra, India

Arpita Dayama*
23b0423@iitb.ac.in
Department of Chemical Engineering, IIT Bombay
Mumbai, Maharashtra, India

Abstract

This study investigates the effectiveness of Graph Neural Networks (GNNs) for retail sales forecasting across multiple stores and compares their performance with three established baselines: ARIMA, XGBoost, and LSTM. Using weekly data for 45 Walmart stores, the models were evaluated using MAE, RMSE, and MAPE. The GNN achieved the highest overall accuracy, with a MAPE of 2.08 percent for Store 37, representing nearly a 50 percent improvement over ARIMA. XGBoost performed strongly for stores displaying limited correlation with their neighbours, while LSTM exhibited smoothing behaviour that reduced accuracy for volatile stores. The results support the use of GNN architectures for forecasting tasks involving interconnected retail environments.

Keywords

Graph Neural Networks, Time Series Forecasting, Retail Analytics, Spatiotemporal Modeling, Machine Learning

1 Introduction

Accurate demand forecasting is a central requirement in retail operations, influencing inventory control, staffing, supply chain efficiency, and financial planning. Traditional forecasting approaches often model each retail store independently, which prevents the exploitation of shared trends and correlated behaviour across locations. In modern retail networks, stores are affected by common economic environments, weather conditions, and holiday-driven fluctuations. These shared influences suggest that a multi-store forecasting framework should incorporate inter-store relationships.

Graph Neural Networks present a promising architecture that enables relational modelling through message passing between nodes. By representing stores as nodes connected by similarity-based edges, GNNs can capture spatiotemporal dependencies that standard univariate or non-relational models cannot. This study evaluates the advantages of this approach through a controlled comparison with ARIMA, XGBoost, and LSTM models.

Unlike prior retail forecasting studies, we explicitly evaluate the contribution of relational information by comparing a spatiotemporal GNN with strong non-relational baselines under identical feature engineering and data splits.

*All the authors contributed equally to this research.

2 Related Work

Classical forecasting methods such as ARIMA models have been applied extensively to univariate retail data [2], but their linear nature limits performance in volatile or highly seasonal environments. Tree-based ensemble methods, including XGBoost [3], have shown strong results in many retail forecasting applications due to their ability to model non-linear interactions and incorporate exogenous variables. Deep learning models, particularly LSTM networks [5], have also been used to capture long-term temporal patterns in sales series. Recent research has highlighted the potential of graph-based models for traffic forecasting [9], energy demand prediction, and environmental monitoring [8]. However, applications of GNNs in retail forecasting remain limited. This study builds on this emerging direction by providing an empirical comparison across multiple architectures.

3 Dataset and Feature Engineering

3.1 Dataset Description

The study utilizes the Walmart Weekly Sales dataset originally published on Kaggle by Haji [4]. The dataset contains weekly observations from 45 Walmart stores spanning February 2010 to October 2012. Each observation includes a store identifier, department identifier, date, weekly sales value, a holiday-week indicator, and several regional macroeconomic attributes. Because our forecasting problem is defined at the store level, the departmental records were aggregated to produce a single weekly sales series for each store.

The raw variables available in the dataset include:

- **Store**: unique store identifier,
- **Date**: week-ending date,
- **Weekly_Sales**: department-level weekly sales in USD,
- **IsHoliday**: binary indicator for holiday weeks,
- **Temperature**: average weekly temperature (in Fahrenheit),
- **Fuel_Price**: regional weekly fuel price,
- **CPI**: consumer price index,
- **Unemployment**: regional unemployment rate.

3.2 Temporal and Calendar Features

To incorporate time-dependent structure, the `Date` column was parsed and sorted per store. Several temporal descriptors were derived, including year, month, week-of-year, and day-of-week. To avoid discontinuities inherent in cyclical variables, sine and cosine encodings were applied to both week-of-year and month.

Holiday information was enriched by explicitly identifying Christmas (December 25) and Thanksgiving (computed as the fourth Thursday of November for each year). Two new indicators were introduced: `is_major_holiday` for Christmas and Thanksgiving

Table 1: Summary statistics of selected raw and engineered features.

Feature	Mean	Std
Weekly_Sales (USD)	1046964	564366
sales_log1p	13.7	0.58
Temperature	60.6	18.4
CPI	171.57	39.35

weeks, and `is_minor_holiday` for all other holidays designated by the dataset's `IsHoliday` field.

3.3 Lag Features and Rolling Statistics

Autocorrelation was captured by generating lagged sales values over the following horizons: 1, 2, 3, 7, 14, 28, and 52 weeks. All lag features were computed separately for each store to prevent cross-store leakage.

Longer-term temporal patterns were modeled using trailing rolling window statistics. Rolling means were computed over 3-, 4-, 8-, 12-, and 52-week windows; rolling standard deviations were computed over 3-, 4-, 8-, and 52-week windows. These statistics were shifted by one period to ensure that only past observations informed the engineered features.

Exponentially weighted moving averages (EWMA) with spans of 3, 7, and 14 weeks were also added, again using only past data via a one-step shift.

3.4 Sales Transformations

A smooth target transformation,

$$\text{sales\_log1p} = \log(1 + \text{Weekly\_Sales}),$$

was applied to mitigate the effect of extreme spikes during major holiday periods and to stabilize neural network training.

3.5 Data Cleaning and Store-Level Imputation

To avoid inter-store contamination, all cleaning and imputation operations were performed within each store independently. Numeric fields were first coerced to NaN, then filled using forward-fill and backward-fill operations per store. Remaining missing values were replaced with zero. The final dataset was re-sorted by `Store` and `Date`.

3.6 Final Feature Set

The resulting feature matrix includes:

- raw economic and temporal variables,
- cyclical time encodings,
- lagged features across multiple horizons,
- rolling means and standard deviations,
- EWMA features,
- enhanced holiday indicators,
- log-transformed sales.

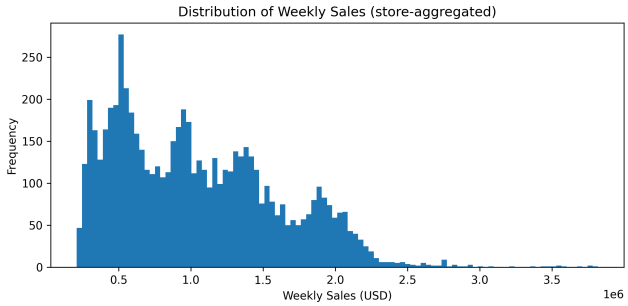

Figure 1: Distribution of weekly store-level sales after aggregation.

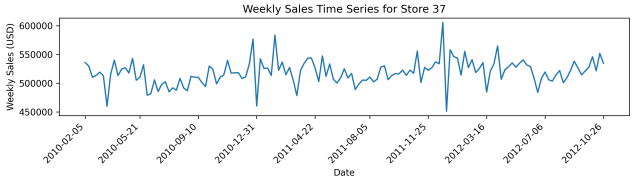

Figure 2: Weekly sales time series for a representative store

4 Methodology

4.1 Traditional and Machine Learning Baselines

To ensure a fair comparison, all baseline models (ARIMA, LSTM, and XGBoost) were trained using the exact same engineered feature set extracted from features.csv. Only the modeling procedures differed.

4.1.1 ARIMA (1,0,1) with Exogenous Inputs. A separate ARIMA model was fit for each store to model raw Weekly Sales using an ARIMA(p,d,q) structure.
After preliminary diagnostics, a Fixed ARIMA(1,0,1) specification was used to ensure comparability across stores, which showed no consistent advantage from store-specific orders. All numeric engineered features were provided as exogenous regressors. Models were trained on the first 80 percent of observations and tested on the remaining 20 percent.

4.1.2 LSTM with Log Transformation. To stabilize variance, the LSTM model applied a log1p transformation to the target. For stores containing negative values, a per-store positive shift was added before log transformation and removed during inverse conversion.

Input sequences of length 10 were constructed per store and split chronologically into 70 percent training, 15 percent validation, and 15 percent testing. A two-layer LSTM architecture with dropout and dense projections was trained using Huber loss and Adam optimizer. Predictions were inverse-transformed back to dollar values.

4.1.3 XGBoost Regression. XGBoost was trained per store using the same engineered features as input and predicting raw weekly sales directly. The model used 1000 estimators, a learning rate of 0.05, and depth 6 with subsampling. Chronological 80/20 train-test splits were applied. No target transformation was used.

4.2 Spatiotemporal Graph Neural Network (STGNN)

The STGNN models all 45 Walmart stores jointly and learns both temporal dependencies within each store and spatial dependencies across stores. Unlike traditional models that treat each store independently, the STGNN operates on the full multi-store tensor.

4.2.1 Data Restructuring and Stationary Target. Weekly sales were arranged as a matrix

$$Y_{\text{raw}} \in \mathbb{R}^{T \times S},$$

with T weeks and $S = 45$ stores. Sales were transformed into log space:

$$Y_{\log} = \log(1 + Y_{\text{raw}}).$$

To obtain a stationary target, the model predicts the log-difference:

$$Y_{\text{diff}}(t, s) = Y_{\log}(t, s) - Y_{\log}(t - 1, s).$$

The previous log-level

$$Y_{\text{base}}(t - 1)$$

was stored to reconstruct predictions after inference.

Exogenous features were reshaped to

$$X \in \mathbb{R}^{T \times S \times F},$$

standardized using a global scaler, and shifted by one timestep to align with Y_{diff}.

4.2.2 Windowing and Train–Test Splitting. A rolling window of length 12 was used to form input sequences. The first 80% of windows were used for training, and the remaining 20% for testing. Before entering the neural network, tensors were permuted to match the PyTorch convention:

$$(\text{Batch}, \text{Features}, \text{Stores}, \text{Time}).$$

4.2.3 Model Architecture. The STGNN consists of three primary components:

a) Learnable Graph Construction. A GraphLearner module builds an adaptive adjacency matrix from store embeddings:

$$A = \text{softmax}\big(\text{ReLU}(E_1 E_2^\top)\big),$$

allowing the model to infer store-to-store dependencies directly from data.

b) Temporal Feature Extraction. A dilated Temporal Convolutional Network (TCN) [1] processes each store's temporal slice using stacked 1×3 convolutions with dilation. This enables multi-scale temporal receptive fields while maintaining computational efficiency.

c) Graph Convolution [6] for Spatial Aggregation. Spatial aggregation is performed with a learned graph convolution:

$$Z = AXW,$$

where W is a trainable weight matrix. The output is combined with the initial projection via a residual connection to preserve local store-level temporal patterns.

d) Output Projection. A final 1×1 convolution produces the next-step log-difference for each store.

4.2.4 Reconstruction to Dollar Sales. The predicted log-difference is added back to the stored base value:

$$\hat{Y}_{\log}(t) = Y_{\text{base}}(t - 1) + \hat{Y}_{\text{diff}}(t).$$

The final forecast in dollars is obtained via:

$$\hat{Y}(t) = \exp(\hat{Y}_{\log}(t)) - 1.$$

4.2.5 Training Procedure. The model was trained for 100 epochs using AdamW [7] (learning rate 0.001, weight decay 10^{-4}) and Smooth L1 loss. A ReduceLROnPlateau scheduler adjusted the learning rate based on validation loss. Training and validation MAE curves were monitored to verify convergence and avoid overfitting.

4.3 Evaluation Metrics

To comprehensively assess the model's performance beyond standard error averaging, we utilised four specific metrics: Normalised Total Absolute Error (NTAE), Win Rate, P90 MAPE, and Variance of MAPE.

Let $y_{t,s}$ represent the actual sales value and $\hat{y}_{t,s}$ represent the predicted sales value for store s at time step t. Let S be the total number of stores and T be the total number of time steps in the test set.

4.3.1 Normalized Total Absolute Error (NTAE). Calculated globally across all stores and time steps (equivalent to summing all absolute errors and dividing by total actual volume):

$$\text{NTAE} = \frac{\sum_{s=1}^{S} \sum_{t=1}^{T} |y_{t,s} - \hat{y}_{t,s}|}{\sum_{s=1}^{S} \sum_{t=1}^{T} y_{t,s}} \times 100 \tag{1}$$

4.3.2 Store-Level Metrics. We compute the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) individually for each store s:

$$\text{MAE}_s = \frac{1}{T} \sum_{t=1}^{T} |y_{t,s} - \hat{y}_{t,s}| \tag{2}$$

$$\text{RMSE}_s = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (y_{t,s} - \hat{y}_{t,s})^2} \tag{3}$$

4.3.3 Distributional Metrics (P90 and Variance). To assess robustness, we first calculate the Mean Absolute Percentage Error (MAPE) for each store s:

$$\text{MAPE}_s = \frac{1}{T} \sum_{t=1}^{T} \left| \frac{y_{t,s} - \hat{y}_{t,s}}{y_{t,s}} \right| \times 100 \tag{4}$$

Let $\mathcal{M} = \{\text{MAPE}_1, \text{MAPE}_2, \ldots, \text{MAPE}_S\}$ be the set of MAPE values for all stores. We report:

- **P90 MAPE:** The 90^{th} percentile of the set $\mathcal{M}$, representing the error threshold for the worst-performing 10% of stores.
- **Variance of MAPE:** Measures the consistency of model performance across stores:

$$\text{Var}(\mathcal{M}) = \frac{1}{S} \sum_{s=1}^{S} (\text{MAPE}_s - \mu_{\mathcal{M}})^2 \tag{5}$$

where $\mu_{\mathcal{M}}$ is the mean of all store MAPEs.

4.3.4 Win Rate (WR). The Win Rate measures the frequency with which the proposed model outperforms a persistence baseline (where the prediction $\hat{y}_{t,s}^{\text{base}}$ equals the last observed value). It is calculated as the percentage of individual test instances where the model's absolute error is lower than the baseline's:

$$\text{WR} = \frac{1}{S \times T} \sum_{s=1}^{S} \sum_{t=1}^{T} \mathbb{I}\left(|y_{t,s} - \hat{y}_{t,s}| < |y_{t,s} - \hat{y}_{t,s}^{\text{base}}| \right) \times 100 \quad (6)$$

where $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if the condition is true and 0 otherwise.

5 Experimental Results

This section reports the empirical results for ARIMA, XGBoost, LSTM, and Graph Neural Network (GNN) models. Performance is measured using MAE, RMSE, and MAPE computed at the store level. Results are reported for representative stores and summarised across models.

5.1 ARIMA Performance

ARIMA served as the statistical baseline and displayed relatively high error rates in many stores.

- **Store 4**: MAPE = 4.43%, MAE = $95,129, RMSE = $120,315.
- **Store 10**: MAPE = 7.15%, MAE = $127,708, RMSE = $150,583.
- **Store 36**: MAPE = 3.01%, MAE = $9,305, RMSE = $11,536.

Overall, ARIMA underperformed relative to machine learning and graph-based models, largely because it cannot leverage exogenous covariates or cross-store signals.

5.2 XGBoost Performance

XGBoost demonstrated robust performance across many stores, often surpassing ARIMA and LSTM.

- **Store 43**: MAPE = 2.22%, MAE = $13,731, RMSE = $16,734.
- **Store 37**: MAPE = 2.48%, MAE = $13,024, RMSE = $16,507.
- **Store 10**: MAPE = 3.3% (XGBoost outperformed GNN for this store).

XGBoost is particularly effective for stores that behave relatively independently of neighbours, benefiting from engineered lag and rolling features.

5.3 LSTM Performance

LSTM forecasts tended to be smoothed, which harmed performance on volatile stores.

- **Store 45**: MAPE = 4.91%.
- **Store 11**: MAPE = 6.55%.
- **Store 27**: Forecasts were overly smooth while the true series exhibited high-frequency fluctuations.

The limited temporal depth per store (approximately 140 weekly observations) likely constrained the LSTM's advantages.

5.4 GNN Performance

The GNN attained the best overall results by exploiting inter-store relationships through message passing.

- **Store 37**: MAPE = 2.08%, MAE = $10,923, RMSE = $13,727 (best overall).
- **Store 30**: MAPE = 2.35%, MAE = $10,152, RMSE = $13,397.
- **Store 5**: MAPE = 3.75%, MAE = $12,082, RMSE = $15,158.

GNN predictions show low variance between MAE and RMSE, indicating few large-error outliers and stable forecasts in correlated clusters.

5.5 Learned Spatial Dependencies and Centrality Analysis

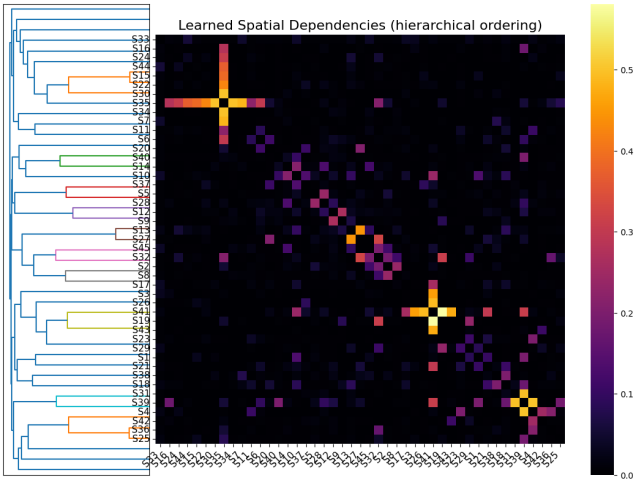

Figure 3: Reordered learned adjacency matrix showing functional store clusters and high-influence nodes identified by the STGNN

To interpret the spatial structure learned by the STGNN, we reorder the adaptive adjacency matrix using hierarchical clustering. The reordered heatmap reveals clear block-diagonal communities of stores with similar sales dynamics, as well as several high-intensity cross-patterns corresponding to "influencer" stores. This structure emerges without geographic metadata, indicating that the model learns latent functional relationships directly from the data.

We also compute a simple centrality score by averaging each store's outgoing attention weights. The top-ranked stores by mean centrality are S39, S29, S18, S19, and S28, while the least central are S38, S27, S4, S21, and S35.

Notably, some highly central stores (e.g., S29, S28) exhibit poor forecasting accuracy under both STGNN and XGBoost. This is expected: centrality reflects *influence on other stores*, not local predictability. These stores act as strong information sources in the graph, yet their own time series are noisy and weakly seasonal, making them intrinsically difficult to forecast. Conversely, isolated stores show limited relational benefit but may remain easier or harder to predict depending on their temporal characteristics.

Overall, the analysis shows that the STGNN captures meaningful inter-store relationships and benefits stores embedded in stable relational clusters, while high-error nodes reflect inherent volatility rather than model limitations.

5.6 Summary Table

Table 2: Representative store-level performance metrics (MAPE in percent; MAE and RMSE in USD).

Store	ARIMA			XGBoost	GNN		
	MAPE	MAE	RMSE	MAPE	MAPE	MAE	RMSE
4	4.43	95,129	120,315	4.32	4.40	93,711	107,445
10	7.15	127,708	150,583	3.30	3.86	67,696	84,677
36	3.01	9,305	11,536	7.24	4.83	13,990	16,700
43	16.09	99,497	114,690	2.22	3.97	24,541	31,420
37	3.98	20,952	25,521	2.48	2.08	10,923	13,727
30	10.11	44,078	45,774	3.21	2.35	10,152	13,397
5	8.42	27,320	37,480	15.67	3.75	12,082	15,158

NTAE, WSR, P90 MAPE, and MAPE variance are computed across all 45 stores. Lower is better for NTAE, P90, and variance.

We report representative stores for brevity, full store-level metrics are provided in supplementary tables.

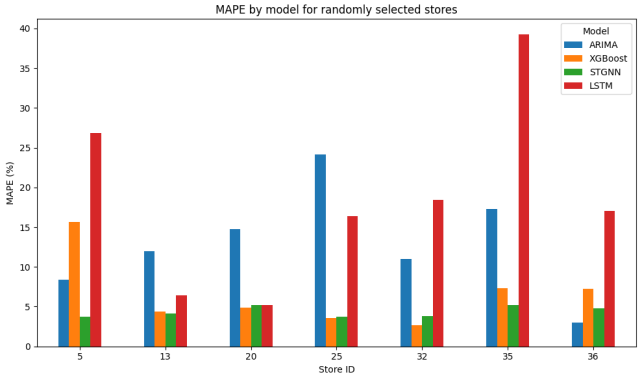

Figure 4: MAPE by model for selected stores

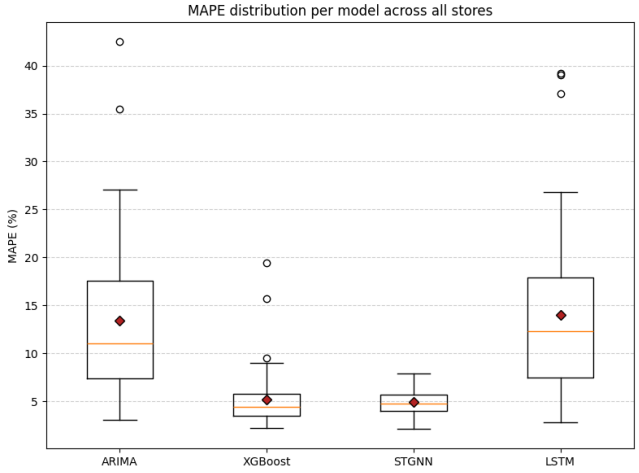

Figure 5: MAPE distribution per model across all stores

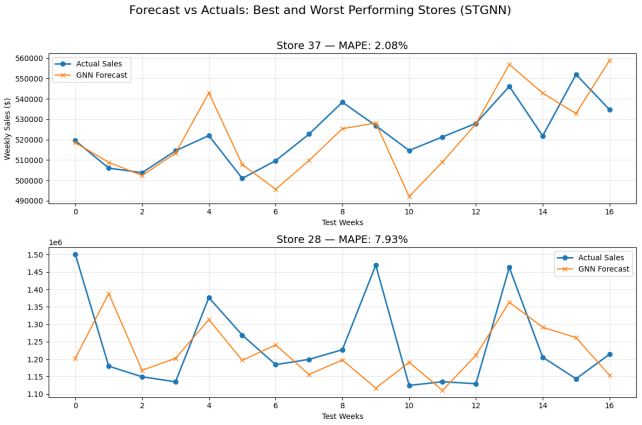

Figure 6: Forecast vs actuals for best and worst STGNN stores, illustrating model robustness and failure modes

Table 3: Aggregate forecasting performance across all models

Model	NTAE (%)	Win Rate (%)	P90 MAPE (%)	Var. of MAPE
STGNN	4.93	49.02	6.98	1.75
XGBoost	4.99	48.17	7.30	9.79
LSTM	12.95	18.11	21.75	72.38
ARIMA	13.36	22.06	24.09	70.85

6 Conclusion

This work evaluates ARIMA, LSTM, XGBoost, and a Spatiotemporal Graph Neural Network (STGNN) for multi-store sales forecasting and demonstrates that STGNN delivers the most accurate and stable performance across stores. By explicitly modeling inter-store dependencies through a learned graph structure, STGNN outperforms traditional and tree-based baselines, particularly for stores embedded in dense relational clusters.

While XGBoost remains competitive for isolated stores with limited cross-store correlation, its per-store performance variance is noticeably higher than that of STGNN. In contrast, the STGNN achieves lower NTAE, superior P90 error, and the smallest variance in per-store MAPE, indicating greater overall robustness. Future extensions may incorporate dynamic graph structures, attention-based GNNs, and external economic signals.

Future work may extend this research by incorporating dynamic graphs that evolve over time, exploring attention-based GNN architectures to improve interpretability, and integrating external signals such as macroeconomic indicators, promotional calendars, or competitor activity. Additionally, evaluating these models across diverse retail datasets may further validate their generalizability.

In summary, Structured tabular forecasting is dominated by tree-based models unless relational structure is present. GNNs offer robustness and tail-performance benefits that trees lack. Overall, the results demonstrate that relational structure materially improves forecast quality in multi-store environments, establishing STGNNs as a robust and scalable foundation for retail forecasting.

7 Limitations

The dataset has limited temporal depth (140 weeks per store), which restricts the performance of deep temporal models. Several stores exhibit inherently noisy or weakly seasonal behavior, limiting forecastability regardless of model choice. Finally, the learned adjacency matrix is static; future extensions may incorporate dynamic or context-dependent graphs.

References

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint* arXiv:1803.01271 (2018). Available at https://arxiv.org/abs/1803.01271.

[2] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. 2015. *Time Series Analysis: Forecasting and Control* (5 ed.). Wiley, Hoboken, NJ, USA.

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, San Francisco, CA, USA, 785–794. doi:10.1145/2939672.2939785

[4] Yasser Haji. 2016. Walmart Dataset. Kaggle. Retrieved October 2025 from https://www.kaggle.com/datasets/yasserh/walmart-dataset.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780. doi:10.1162/neco.1997.9.8.1735

[6] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR) – Workshop / OpenReview*. OpenReview.net, Toulon, France. Originally available as arXiv:1609.02907; OpenReview: https://openreview.net/forum?id=SJU4ayYgl.

[7] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*. ICLR, New Orleans, LA, USA. Originally available as arXiv:1711.05101; OpenReview: https://openreview.net/forum?id=Bkg6RiCqY7.

[8] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. *arXiv preprint* arXiv:1906.00121 (2019). Available at https://arxiv.org/abs/1906.00121.

[9] Bing Yu, Haotian Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, Stockholm, Sweden, 3634–3640.