# Mini-PatchTST: A Lightweight Patch-Based Time-Series Transformer for Metro Passenger Flow Forecasting

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Efficient urban transportation systems are essential for the quality of life in modern cities. Accurate public transport demand forecasting plays a key role in optimizing system operations and resource allocation. However, traditional approaches often struggle with the complex temporal patterns present in urban transit data. Recent advances in deep learning have demonstrated strong performance in time series forecasting. This study introduces Mini-PatchTST, a lightweight variant of PatchTST, designed for metro passenger demand forecasting with reduced complexity. To support this work, a novel station-level hourly metro passenger flow dataset was constructed using open data from the Istanbul Metropolitan Municipality. The proposed model is benchmarked against several strong forecasting methods, including deep architectures such as PatchTST and LSTM, as well as machine learning models such as Light Gradient Boosting Machine (LightGBM) and Extreme Gradient Boosting (XGBoost). Experiments on the Istanbul Metro Dataset show that Mini-PatchTST reduces RMSE by 5–12% against gradient boosting models and by 4% against PatchTST despite its smaller size. Its generalization ability is confirmed on the Hangzhou Metro dataset, where it reduces RMSE by 17–64% over LSTM baselines and outperforms the state-of-the-art TPA-LSTM with up to 22% MAE and over 14% RMSE improvements in transfer and intermediate station settings. The results demonstrate that Mini-PatchTST offers a robust, scalable, and efficient forecasting solution.

## 1 Introduction

Urban transportation systems play a vital role in the functioning of modern cities by providing mobility for millions of residents and visitors daily. In megacities such as Istanbul, where persistent challenges arise from high traffic congestion and dense population, optimizing metro services is essential to enhancing passenger experience and improving the overall efficiency of the transportation network. Accurate passenger count predictions are crucial for resource management, schedule planning, and operational efficiency.

Historically, public transport demand forecasting relied on traditional statistical methods, such as ARIMA, which offer interpretability but struggle with the complexity inherent in urban transit data (Ren et al., 2022). With the advancement of data collection and processing technologies, more sophisticated machine learning (ML) and deep learning (DL) techniques, including Long short-Term Memory (LSTM), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN), have emerged, providing improved accuracy due to their ability to model complex temporal and spatial dependencies. Recently, among the innovations in DL architectures, Transformer models such as Patch Time Series Transformer (PatchTST) (Nie et al., 2023a) have stood out for their superior performance in long-term time series forecasting. These architectures, leveraging attention mechanisms and a non-recurrent structure, have the potential to capture long-range dependencies better.

In this study, we introduce a lightweight forecasting model, Mini-PatchTST, tailored for metro passenger flow prediction. The model is developed as a simplified version of PatchTST, aiming to retain strong predictive performance while reducing computational complexity and improving practical usability. We also present the Istanbul Metro Dataset, a newly collected and structured metro passenger dataset from the Istanbul

Metropolitan Municipality that reflects real-world operational conditions and provides a strong basis for evaluating forecasting models. The main contributions of this study are threefold.

- We propose a lightweight transformer-based forecasting model, referred to as Mini-PatchTST, designed for metro passenger flow prediction. By adopting a patch-based temporal segmentation strategy, the proposed model captures both short-term variations and long-range temporal dependencies while maintaining a compact and computationally efficient architecture.
- We present the construction, preprocessing, and public release of the Istanbul Metro Dataset, a real-world passenger flow dataset collected from the open-source data platform of the Istanbul Metropolitan Municipality. Motivated by the well-documented scarcity of publicly available passenger demand datasets in the transit forecasting literature, where only a small fraction of studies share their data or code (Rocco di Torrepadula et al., 2024), the dataset is carefully structured and made publicly accessible to promote reproducibility, enable fair benchmarking, and serve as a valuable resource for further research on metro passenger demand forecasting.
- We conduct a comprehensive experimental evaluation in which Mini-PatchTST is systematically compared with traditional statistical approaches, machine learning models, and state-of-the-art deep learning methods across two real-world datasets, namely the Istanbul Metro Dataset and the Hangzhou Metro Dataset (Liu et al., 2022). The experimental results highlight Mini-PatchTST's ability to generalize across different cities and demonstrate its potential value for data-driven decision-making in urban transportation planning. The Mini-PatchTST implementation, including the model architecture and the Istanbul Metro Dataset, is publicly available at `hidden_link`.

## 2 Related Work

Early studies on public transport demand forecasting primarily relied on traditional statistical time series models due to their simplicity and interpretability. Among these, AutoRegressive Integrated Moving Average (ARIMA) and its seasonal extension (SARIMA) were widely applied to metro and railway passenger flow prediction. ARIMA-based models have been shown to perform well for short-term forecasting under relatively stable and stationary conditions, as demonstrated by Ren et al. (2022) using Hangzhou Metro's smart card data. To account for periodic demand patterns, SARIMA models explicitly incorporate seasonal components and have achieved satisfactory results in long-term railway demand forecasting, such as the study by Milenković et al. (2016) on the Serbian railway network. However, these parametric approaches are limited in their ability to model nonlinear dynamics and complex temporal variations commonly observed in large-scale urban transit systems, motivating the shift toward data-driven learning-based methods.

Deep Learning models have achieved significant breakthroughs in metro passenger flow forecasting, particularly due to their ability to capture complex temporal dependencies. Among these, LSTM networks have been widely adopted for sequence modeling. By incorporating gated mechanisms, LSTMs address the vanishing gradient problem of traditional RNNs and enable the learning of long-term temporal patterns. Numerous studies have demonstrated their effectiveness in public transportation demand forecasting. For example, Kang et al. (2020) and Siswanto et al. (2024) applied LSTM-based models to metro and bus demand prediction tasks, reporting substantial improvements in forecasting accuracy compared to conventional methods.

Several studies have further validated the applicability of LSTM architectures using real-world metro datasets. Hu (2023) employed an LSTM model for short-term passenger flow prediction at metro stations using 10-minute smart card data from the Hangzhou metro system collected between January 1 and January 20, 2019. Model parameters were optimized using RMSE, Mean Absolute Percentage Error (MAPE), and $R^2$ (R-squared) metrics. Building upon this work, Wei et al. (2023) utilized the same Hangzhou Metro Dataset with an extended time span (January 1–26, 2019) and proposed a hybrid TPA-LSTM model. By integrating a temporal pattern attention mechanism, TPA-LSTM aimed to enhance the model's ability to capture inter-temporal dependencies across multiple time steps, thereby addressing limitations of conventional LSTM architectures.

Empirical results reported in Wei et al. (2023) demonstrate the quantitative advantages of the TPA-LSTM model over the conventional LSTM across multiple experimental settings. Compared with LSTM, TPA-LSTM reduces MAPE by approximately 1.8–2.7% points at ordinary stations (e.g., Linping), by about 1.9% points under weekday and weekend scenarios, and by around 2.7% points at transfer stations (e.g., Jinjiang). In terms of absolute error metrics, TPA-LSTM achieves average reductions of 3–4% in MAE and 4–6% in RMSE, while consistently improving the coefficient of determination $R^2$ by 0.02–0.04. These results indicate that incorporating the temporal pattern attention mechanism enables TPA-LSTM to more effectively capture complex inter-temporal dependencies, thereby leading to more accurate and robust passenger flow predictions across diverse station types and temporal conditions.

More recently, attention-based models have gained increasing attention in time series forecasting. The Transformer architecture, originally developed for natural language processing (Vaswani et al., 2023), has been successfully adapted for temporal prediction tasks through self-attention mechanisms. Unlike recurrent models, Transformers dynamically assign weights to past observations, enabling more effective modeling of long-range dependencies. Studies by Zuo et al. (2023) and Wang et al. (2023) have demonstrated that Transformer-based models outperform recurrent architectures in metro passenger flow prediction, particularly when incorporating exogenous variables such as weather conditions and calendar effects.

New time series forecasting architectures, such as PatchTST, have been proposed to address long-horizon time series forecasting by combining Transformer models with a patching strategy that segments temporal data into meaningful local blocks. PatchTST has demonstrated strong performance across a wide range of domains, including electricity load forecasting (Lu et al., 2025), wind speed forecasting (Ma et al., 2025), and financial time series forecasting (Noguer i Alonso & Pereira Franklin, 2024). However, PatchTST has not previously been explored for metro passenger demand forecasting, where data exhibit strong periodicity, abrupt demand fluctuations, and station-specific heterogeneity.

Metro passenger flow prediction presents unique challenges that align well with the design principles of PatchTST. In particular, the presence of strong daily and weekly seasonality, short-term demand spikes during peak hours, and varying station-level demand patterns make patch-based temporal modeling especially suitable. On the other hand, the original PatchTST architecture is designed for large-scale multivariate forecasting tasks and may create unnecessary computational overhead in short-horizon, station-level prediction settings. These observations motivate the adaptation of PatchTST to the public transportation domain and the development of a simplified, task-specific architecture. In this study, we address this gap by proposing Mini-PatchTST, an optimized and lightweight version of PatchTST tailored specifically for short-term metro passenger flow forecasting.

## 3 Methodology

### 3.1 Data Collection and Preprocessing

#### 3.1.1 The Istanbul Metro Dataset

The Istanbul Metro Dataset was constructed using raw level data obtained from the Istanbul Metropolitan Municipality (IMM) open data platform. However, an initial exploratory analysis showed that only a limited temporal subset of the data was reliable and suitable for modeling. In particular, data outside July 2024 exhibited inconsistencies and missing records. As a result, the analysis focused exclusively on the period July 1–31, 2024, during which data quality was consistent across stations and time.

The source data include passenger usage records for multiple transport modes, such as metro, tram, funicular, bus, and ferry services. Although travel direction is inherently determined when passengers use buses and ferries, this information is not explicitly available in the released data. As a result, the absence of directional information makes it difficult to accurately model station level passenger flows. In contrast, metro systems inherently operate without requiring passengers to specify a travel direction at entry, making the absence of directional information non-problematic. For this reason, only metro-related records were used in the construction of the dataset. After filtering, the Istanbul Metro Dataset covered 265 stations across 10 metro lines, 3 tram lines, and 3 funicular lines.

The publicly released data provide passenger counts with several categorical breakdowns, including card type (full fare, discounted, and free passes) and trip type (transfer versus non-transfer). For this study, these records were aggregated into hourly passenger entry counts at the station level, resulting in a univariate time series per station. The final dataset therefore reflects typical urban travel behavior. The training period was defined as July 1–24, 2024, with 151,193 observations, whereas the test period covered July 25–31, 2024, with 44,880 observations. The forecasting task aims to predict passenger demand for the next three hourly time steps.

Table 1 presents descriptive statistics for each metro line and provides an overview of passenger demand patterns used in the forecasting analysis. The number of stations reflects the physical size of each line, while the average and maximum hourly passenger counts describe typical demand levels and peak congestion, with larger lines such as Marmaray and T1 carrying higher passenger volumes. For all lines, the most crowded hour occurs at 18:00, corresponding to the evening peak period and indicating that demand is strongly driven by commuting behavior. The most crowded stations are generally major transfer points or central locations, such as Yenikapı, Şişli, and Eminönü. Overall, these statistics highlight differences between lines in terms of scale, peak intensity, and spatial concentration of demand, supporting the design of accurate demand forecasting models.

Table 1: Descriptive statistics of metro lines in the Istanbul Metro Dataset

| Line | Number of stations | Average hourly passengers | Maximum hourly passengers | Most crowded station |
|---|---|---|---|---|
| F1 | 2 | 764 | 1,849 | KABATAŞ |
| F4 | 2 | 181 | 544 | ASİYAN |
| M1 | 23 | 13,983 | 33,238 | YENİKAPI |
| M2 | 16 | 17,938 | 54,779 | ŞİŞLİ |
| M3 | 19 | 5,836 | 17,677 | MAHMUTBEY |
| M4 | 23 | 12,009 | 32,350 | KADIKÖY |
| M5 | 20 | 12,075 | 34,762 | ÜSKÜDAR |
| M6 | 4 | 602 | 2,039 | LEVENT |
| M7 | 17 | 8,990 | 25,363 | MECİDİYEKÖY |
| M8 | 13 | 2,923 | 9,588 | DUDULLU |
| M9 | 13 | 2,619 | 8,125 | M9 YENİBOSNA |
| MARMARAY | 43 | 26,090 | 62,778 | YENİKAPI |
| T1 | 31 | 16,328 | 32,205 | EMİNÖNÜ |
| T4 | 22 | 7,609 | 19,331 | KİPTAŞ VENEZİA |
| T5 | 14 | 2,809 | 7,754 | EMİNÖNÜ |
| TUNEL | 2 | 561 | 1,213 | KARAKÖY |

#### 3.1.2 The Hangzhou Metro Dataset

In addition to the Istanbul Metro Dataset, we utilized the Hangzhou Metro Dataset (Liu et al., 2022) which is based on anonymized smart card swipe records collected from the Hangzhou urban rail transit system. It contains detailed transaction information, including timestamp, station and line identifiers, and entry or exit status. The raw data cover January 1–20, 2019 (and up to January 26, 2019 in some studies), during which Hangzhou operated three metro lines with a total of 81 stations. Passenger flows are commonly aggregated into 10-minute intervals, resulting in 144 time steps per station per day for inbound and outbound flows. After basic cleaning to remove invalid transactions and off-hours records, the dataset provides a realistic snapshot of daily and peak-hour travel patterns in a large urban metro network (Ren et al., 2022). This dataset has been used under different experimental settings in prior studies. Hu (2023) analyzed passenger flow data collected between January 1 and January 20, 2019. Their study focused on two high-demand stations—Longxiang Bridge Station and East Railway Station—selected from a total of 81 stations to ensure sufficient training samples. The data were aggregated at 10-minute intervals, yielding approximately 2,736 time steps per station over 19 valid days. January 1, a public holiday characterized by abnormal demand patterns, was excluded from the analysis to evaluate model robustness. Data from January 2 to January 19 were used for training, whereas January 20 served as the test period. In this study, the same station selection and experimental settings were adopted to enable a direct and fair evaluation of Mini-PatchTST under comparable conditions. Similarly, Wei et al. (2023) utilized automated fare collection data from the Hangzhou metro system spanning January 1–26, 2019, and focused on three stations: Linping, Jinjiang, and Xinfeng. To ensure consistency and fairness in comparison, Mini-PatchTST was evaluated using the same

data selection, preprocessing procedures, station selection, and training–testing split as described in their experimental protocol.

### 3.2 Mini-PatchTST: A Lightweight Transformer Architecture for Forecasting

The original PatchTST model is a Transformer-based forecasting architecture designed for long multivariate time series by operating on temporal patches rather than individual time steps. Given a historical input sequence, PatchTST divides the historical time series into fixed-length temporal patches. Each patch is mapped into a latent representation and then processed by a Transformer encoder. By working at the patch level rather than at individual time steps, the model shortens the effective input sequence, improves computational efficiency, and is able to capture both short-term patterns and longer-term temporal dependencies.

Mini-PatchTST is a compact and task-focused adaptation of the PatchTST architecture designed for short-term passenger flow forecasting in urban rail systems. Mini-PatchTST relies on historical multivariate time series data and does not incorporate external variables or future information. Temporal and spatial identifiers, including hour of day, day of week, and station identity, are derived directly from past observations and transformed into numerical inputs using ordinal encoding. In this study, ordinal encoding is implemented by assigning a unique integer value to each categorical level. Formally, for a categorical variable $c \in \mathcal{C}$, the encoding defines a mapping

$$\phi(c) = k, \quad k \in \{0, 1, \ldots, |\mathcal{C}| - 1\}, \tag{1}$$

where each category is represented by a distinct numerical identifier.

These encoded temporal and spatial features are treated as numerical inputs and combined with the target variable to form the multivariate input sequence. By adopting ordinal encoding instead of learned embedding layers, the model maintains a simple and computationally efficient structure while still capturing station-specific characteristics and time-dependent patterns present in the data. All input features are constructed solely from historical information, ensuring that no future or target-related signals are used during training or inference.

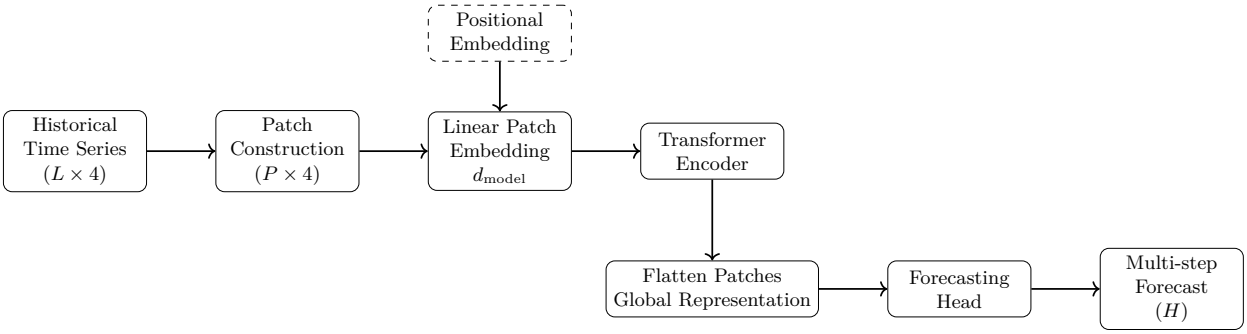



Figure 1: Overview of the proposed Mini-PatchTST architecture. The historical multivariate time series of length $L$ with four input features is divided into non-overlapping temporal patches of length $P$. Each patch is linearly projected into a latent embedding space of dimension $d_{\text{model}}$ and combined with positional embeddings before being processed by a Transformer encoder. The encoder outputs are flattened to form a global representation, which is passed to a forecasting head to produce a multi-step passenger flow forecast over a horizon of length $H$.

An overview of the Mini-PatchTST architecture is presented in Figure 1, highlighting the main components of the model, including patch-based input construction, Transformer-based temporal modeling through self-attention, and the final forecasting head. Together, these components form a compact yet expressive forecasting framework that is well suited for short-term passenger flow prediction in urban rail transit systems. The following subsections describe the Mini-PatchTST architecture in detail.

#### 3.2.1 Input Features and Representation

Given a forecasting time $t$, the model operates solely on historical information from the look-back window of length $L$. Each past time step $\tau \in \{t-L, \ldots, t-1\}$ is represented by a four-dimensional feature vector:

$$\mathbf{x}_\tau = \left[y_\tau^{\text{scaled}},\ h_\tau^{\text{enc}},\ d_\tau^{\text{enc}},\ s_\tau^{\text{enc}}\right], \tag{2}$$

where $y_\tau^{\text{scaled}}$ denotes the normalized passenger flow observation at time $\tau$, and $h_\tau^{\text{enc}}$, $d_\tau^{\text{enc}}$, and $s_\tau^{\text{enc}}$ correspond to the encoded representations of hour of day, day of week, and station identifier, respectively.

The complete model input is therefore given by the multivariate time series

$$\mathbf{X}_t = [\mathbf{x}_{t-L}, \mathbf{x}_{t-L+1}, \ldots, \mathbf{x}_{t-1}] \in \mathbb{R}^{L\times 4}, \tag{3}$$

which is subsequently partitioned into non-overlapping temporal patches before being processed by the transformer encoder.

The temporal variables and station identifiers are categorical by nature. To enable their integration into the neural forecasting framework, each category is mapped to a unique numerical identifier through ordinal encoding. All input features are then scaled to comparable numerical ranges to improve optimization stability during training.

#### 3.2.2 Patch Construction and Embedding

Let

$$\mathbf{X} \in \mathbb{R}^{B\times L\times F} \tag{4}$$

denote the input tensor, where $B$ is the batch size ($B = 32$ in this study), $L$ is the historical input length, and $F = 4$ is the number of features per time step.

The historical sequence is partitioned into non-overlapping temporal patches of fixed length $P$, resulting in

$$N = \frac{L}{P} \tag{5}$$

patches per sample. In this study, for the Istanbul Metro Dataset (hourly intervals), $P = 24$ is used, corresponding to one full day of observations. For the Hangzhou Metro Dataset (10-minute intervals), $P = 18$ is applied, corresponding to a 3-hour window. Each patch

$$\mathbf{X}_i \in \mathbb{R}^{P\times F}, \quad i = 1, \ldots, N, \tag{6}$$

captures short-term temporal dynamics across all input features.

Each patch is flattened into a vector

$$\mathbf{z}_i = \text{vec}(\mathbf{X}_i) \in \mathbb{R}^{P\cdot F}, \tag{7}$$

where $\mathbf{z}_i$ represents a raw, unstructured description of the $i$-th temporal patch obtained by concatenating its feature values across time. This vector contains local temporal information but does not yet encode higher-level abstractions suitable for sequence modeling.

To obtain a compact and learnable representation, each flattened patch vector is projected into a latent embedding space of dimension $d_{\text{model}}$ via a linear transformation:

$$\mathbf{e}_i = \mathbf{W}_p\mathbf{z}_i + \mathbf{b}_p, \quad \mathbf{e}_i \in \mathbb{R}^{d_{\text{model}}}, \tag{8}$$

where $\mathbf{W}_p \in \mathbb{R}^{d_{\text{model}}\times(P\cdot F)}$ and $\mathbf{b}_p \in \mathbb{R}^{d_{\text{model}}}$ are learnable parameters that linearly project each flattened temporal patch into the model's embedding space. The resulting embedding $\mathbf{e}_i$ provides a dense representation of the patch that captures interactions among temporal and feature dimensions, enabling efficient processing by the Transformer encoder. $\mathbf{p}_i$ is a learnable positional embedding vector that encodes the temporal position of the $i$-th patch and is added to its embedding to preserve ordering information:

$$\tilde{\mathbf{e}}_i = \mathbf{e}_i + \mathbf{p}_i. \tag{9}$$

The resulting embedded patch sequence

$$\tilde{\mathbf{E}} \in \mathbb{R}^{B \times N \times d_{\text{model}}} \tag{10}$$

serves as the input to the Transformer encoder.

#### 3.2.3 Transformer Encoder

The embedded patch sequence

$$\tilde{\mathbf{E}} = [\tilde{\mathbf{e}}_1, \ldots, \tilde{\mathbf{e}}_N] \in \mathbb{R}^{B \times N \times d_{\text{model}}} \tag{11}$$

is processed by a stack of $L_e$ Transformer encoder layers. Here, $B$ denotes the batch size, $N$ is the number of temporal patches within the historical window, and $d_{\text{model}}$ is the embedding dimension. Through self-attention, each patch can directly interact with all others, allowing the model to capture both short-term temporal patterns and longer-range dependencies.

Within an encoder layer, the input representations $\mathbf{E}$ are first projected into queries, keys, and values:

$$\mathbf{Q} = \mathbf{E}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{E}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{E}\mathbf{W}_V, \tag{12}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ are learned parameters and $d_k$ is the dimensionality of each attention head.

Self-attention is computed using scaled dot-product attention:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{13}$$

where $\text{Attn}(\cdot)$ softly aggregates information from all patches based on their pairwise relevance.

To capture different temporal relationships simultaneously, attention is performed in parallel using $H$ heads:

$$\text{MHA}(\mathbf{E}) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_H\right)\mathbf{W}_O, \tag{14}$$

where each head applies $\text{Attn}(\cdot)$ in a different learned subspace. The learnable output projection $\mathbf{W}_O \in \mathbb{R}^{(Hd_k) \times d_{\text{model}}}$ linearly combines and reprojects the concatenated head outputs back to the model dimension, enabling information mixing across heads and ensuring dimensional compatibility.

The output of the attention block is added to the input and normalized:

$$\mathbf{E}' = \text{LayerNorm}\left(\mathbf{E} + \text{MHA}(\mathbf{E})\right), \tag{15}$$

which stabilizes training and preserves information through a residual connection.

Next, a position-wise feed-forward network is applied independently to each patch:

$$\text{FFN}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{16}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ are learnable weight matrices that project the input to a higher-dimensional hidden space and back to the model dimension, respectively, and $\mathbf{b}_1$ and $\mathbf{b}_2$ are corresponding bias terms. The function $\sigma(\cdot)$ denotes a non-linear activation, enabling the FFN to refine each patch representation independently.

Finally, a second residual connection and normalization produce the output of the encoder layer:

$$\mathbf{E}_{\text{out}} = \text{LayerNorm}\left(\mathbf{E}' + \text{FFN}(\mathbf{E}')\right). \tag{17}$$

Since Mini-PatchTST is designed for short-horizon forecasting with a compact architecture, only a small number of encoder layers $L_e$ is used, ensuring an effective balance between model expressiveness and computational efficiency.

#### 3.2.4 Global Representation and Forecasting Head

The output of the Transformer encoder is a tensor

$$\mathbf{H} \in \mathbb{R}^{B \times N \times d_{\text{model}}}, \tag{18}$$

where $B$ is the batch size, $N$ is the number of patches, and $d_{\text{model}}$ is the embedding dimension. Each element of $\mathbf{H}$ represents a context-aware patch embedding that already incorporates information from all other patches in the historical window.

To obtain a single representation of the full input sequence, the encoder output is flattened along the patch dimension:

$$\mathbf{h} \in \mathbb{R}^{B \times (N \cdot d_{\text{model}})}. \tag{19}$$

This operation simply concatenates all patch embeddings into one global vector per sample, allowing the model to summarize the entire historical context.

The global representation $\mathbf{h}$ is then passed through a fully connected layer that projects it into a lower-dimensional latent space. This layer acts as a compression step, producing a compact summary vector that captures the overall temporal dynamics of the input sequence. A final linear layer maps this summary directly to the forecasting horizon:

$$\hat{\mathbf{y}} \in \mathbb{R}^{H}, \tag{20}$$

where $\hat{\mathbf{y}}$ denotes the predicted future values and $H$ is the number of time steps to be forecast.

With this design, passenger flow forecasting is formulated as a sequence-to-vector regression task, where the full historical sequence is jointly used to predict all future time steps in a single forward pass.

#### 3.2.5 Differences from the PatchTST Architecture

Mini-PatchTST uses a feature-joint patch representation. Instead of treating each variable separately, all input features within a temporal patch are embedded together into a single latent vector. This allows each patch to directly capture interactions between different types of information, such as temporal indicators, spatial context, and categorical attributes, at the input level.

Mini-PatchTST is also deliberately designed as a lightweight version of PatchTST. While the original architecture typically relies on deeper Transformer encoders and larger embedding dimensions to model long-range dependencies, Mini-PatchTST reduces the overall model size by using fewer encoder layers, a smaller latent dimension, and a simplified prediction head. These choices are well aligned with the short-horizon forecasting setting considered in this study, where efficiency and robustness are prioritized over maximum model complexity. Table 2 summarizes the architectural differences between the original PatchTST model and the proposed Mini-PatchTST.

Overall, Mini-PatchTST retains the core idea of patch-based temporal modeling from PatchTST, while simplifying the architecture to better suit short-term forecasting tasks and computationally constrained environments.

## 4 Experiments

This section presents the experimental setup and evaluation results of the proposed Mini-PatchTST model. Experiments are conducted on two metro passenger flow datasets of two megacities with different roles: the Istanbul Metro Dataset, which is collected and prepared in this study and serves as the primary dataset for model comparison; and the Hangzhou Metro Dataset, a widely used public benchmark employed to assess the generalization performance of the proposed approach.

### 4.1 Comparison of Models on the Istanbul Metro Dataset

The models evaluated on the Istanbul Metro Dataset in this study were grouped into three main categories: traditional statistical models, machine learning models, and deep learning models. The traditional statistical

Table 2: Architectural comparison between the original PatchTST and the proposed Mini-PatchTST

| **Component** | **PatchTST** | **Mini-PatchTST** |
|---|---|---|
| Model objective | Long-horizon forecasting | Short-horizon, efficient forecasting |
| Input handling | Channel-independent patching | Multivariate patch fusion |
| Patch embedding | $patch_len \rightarrow d_{model}$ | $(patch_len \times 4) \rightarrow d_{model}$ |
| Embedding dim. ($d_{model}$) | 64 | 64 |
| Positional encoding | Learnable | Learnable |
| Transformer layers | 6[a] | 2[b] |
| Attention heads | 32 | 4 |
| Transformer type | Encoder-only | Encoder-only |
| Residual/connector layers | Implicit (Transformer blocks) | Explicit connector layer |
| Output aggregation | Flatten + linear head | Flatten + projection head |
| Final prediction head | $d_{model} \rightarrow H$ | $d_{model} \rightarrow H$ |
| Memory footprint | High | Low |

[a] The number of encoder layers in PatchTST is a tunable hyperparameter; the value of 6 reflects the configuration used in our experimental evaluation on the Istanbul Metro Dataset, consistent with the setting reported in the original study (Nie et al., 2023a).

[b] The value of 2 reflects the default configuration used for the Istanbul Metro Dataset. For the Hangzhou Metro Dataset, the encoder depth is treated as a dataset-specific hyperparameter and tuned over {2, 3, 4}, following the same convention as the original PatchTST architecture.

approaches included the naive, seasonal naive, and window average methods. The machine learning models comprised the LightGBM and XGBoost algorithms. The deep learning category included LSTM networks, the original PatchTST model, and the proposed Mini-PatchTST architecture, which uses two Transformer encoder layers.

Model performance was assessed using two widely adopted error metrics: MAE and RMSE. MAE measures the average magnitude of the absolute differences between predicted and observed values, whereas RMSE assigns a higher penalty to larger errors, making it more sensitive to outliers.

### 4.2 Evaluation of Mini-PatchTST on the Hangzhou Metro Dataset

For the Hangzhou Metro Dataset, the training and evaluation procedure was designed to remain consistent with the protocols presented in previous studies (Hu, 2023; Wei et al., 2023), in which passenger flow forecasting is conducted for selected stations and directions.

Since the Hangzhou experiments are designed to replicate the protocols of Hu (2023) and Wei et al. (2023), the competing results are taken directly from those studies rather than re-implemented. This approach ensures the comparison reflects the original experimental conditions as closely as possible, at the cost of a fully unified evaluation environment. This is noted as a limitation of the comparison.

In Hu (2023), the model is trained using data from January 2 to January 19, and forecasts are generated for January 20. In that study, the dataset comprises observations from two monitoring stations, namely Longxiang Bridge and East Railway. January 19 was used as the validation set for hyperparameter tuning, whereas January 20 was reserved for testing.

In Wei et al. (2023), the experiments focus on the Jinjiang and Xinfeng stations, with the training period covering January 1 to January 24 and the test period corresponding to January 25–26. To remain consistent with this setting, January 23–24 were used as validation days for hyperparameter selection in this research.

We tuned the number of Transformer encoder layers (2, 3, and 4) and training epochs (90, 100, 110, 130, and 150), selecting the best configuration based on validation performance.

For the scenario in Hu (2023), the optimal Mini-PatchTST settings were 130 epochs with three encoder layers for Longxiang Bridge Station Inbound and East Railway Station Outbound, and 110 epochs with two and four encoder layers for East Railway Station Inbound and Longxiang Bridge Station Outbound, respectively. For the scenario in Wei et al. (2023), the best configurations were 110 epochs with four encoder

Table 3: Comparison of RMSE, MAE, MAPE and training duration on the Istanbul Metro Dataset

| Model | RMSE | MAE | MAPE (%) | Training Duration (s) |
|---|---|---|---|---|
| Naive | 485.27 | 223.33 | 262.74 | 2 |
| Seasonal Naive | 240.04 | 80.55 | 35.85 | 2 |
| Window Average | 527.39 | 292.75 | 714.39 | 3 |
| LGBMRegressor (Ke et al., 2017) | 91.95 | 35.19 | 15.90 | 126 |
| XGBRegressor (Chen & Guestrin, 2016) | 84.16 | 33.32 | 15.24 | 117 |
| LSTM (Hochreiter & Schmidhuber, 1997) | 75.43 | 32.62 | 27.54 | 18018 |
| PatchTST (Nie et al., 2023b) | 84.23 | 39.79 | 33.89 | 7039 |
| Mini-PatchTST | 79.71 | 35.80 | 23.42 | 4794 |
| Mini-PatchTST (trained per station) | 81.65 | 38.50 | 31.04 | 5017 |

layers for Xinfeng Station and two encoder layers for Jinjiang Station. These settings were used for the final test evaluation.

## 5 Results and Discussion

This section presents the performance of the proposed Mini-PatchTST model in comparison with baseline models on the Istanbul Metro Dataset, as well as the results of scenario-based comparisons on the Hangzhou Metro Dataset.

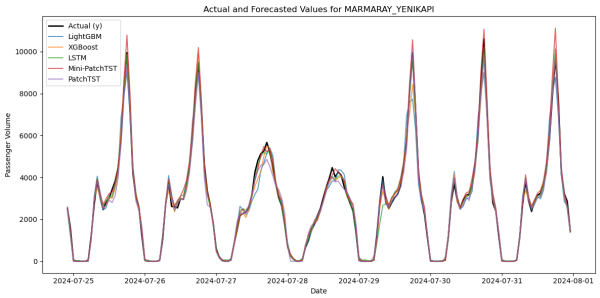
### 5.1 Performance on the Istanbul Metro Dataset

Table 3 displays the RMSE and MAE values for all models on the Istanbul Metro Dataset (25-31 July as the test period).

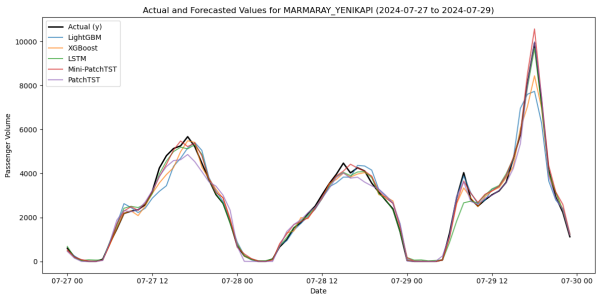
From the results, clear performance differences can be observed across model categories. Traditional statistical models (Naive, Seasonal Naive, and Window Average) exhibit the highest error values across all evaluation metrics (RMSE, and MAE), indicating limited capacity to model the complex and non-linear temporal patterns present in metro passenger flow data. In contrast, Mini-PatchTST substantially outperforms these baselines on all metrics, demonstrating the effectiveness of deep learning architectures for this task.

When compared with machine learning models (LGBMRegressor and XGBRegressor), Mini-PatchTST achieves a lower RMSE, indicating superior overall error reduction. Specifically, its RMSE of 79.71 is lower than that of the best-performing machine learning model, XGBRegressor (84.16). However, Mini-PatchTST exhibits a slightly higher MAE compared to XGBRegressor, which may be attributed to its focus on capturing global temporal dependencies rather than minimizing point-wise absolute errors, leading to occasional larger local deviations. In terms of training duration, machine learning models (LGBMRegressor: 126s, XGBRegressor: 117s) train significantly faster than deep learning models, owing to their lower architectural complexity.

Among the deep learning models, LSTM achieves the lowest RMSE (75.43), followed by Mini-PatchTST (79.71) and PatchTST (84.23). The strong performance of LSTM is consistent with prior studies that identify LSTM as a competitive and reliable architecture for time series forecasting, particularly due to its ability to capture long-term temporal dependencies through gated memory mechanisms (Hochreiter & Schmidhuber, 1997). However, LSTM requires the longest training duration among all deep learning models (18,018s), which presents a significant practical limitation for large-scale deployment. Mini-PatchTST, on the other hand, achieves comparable accuracy with a substantially reduced training time of 4,794s, outperforming both LSTM and PatchTST (7,039s) in computational efficiency. While LSTM attains the best absolute accuracy on the Istanbul Metro Dataset, Mini-PatchTST surpasses the original PatchTST and delivers comparable performance with a more compact and efficient architecture, making it a practical alternative for large-scale urban metro traffic prediction.




(a) Predicted and actual hourly passenger flow at the Marmaray–Yenikapı station.




(b) Predicted and actual hourly passenger flow at the Marmaray–Yenikapı station between July 27 and 29, highlighting the days with the largest performance differences among models.

Figure 2: Comparison of predicted and actual passenger flow at the Marmaray–Yenikapı station.

As shown in Figure 2a, all models generally forecast passenger demand well, successfully capturing seasonality and peak periods. The performance differences between the models are mainly driven by relatively small details. This can be observed more explicitly in Figure 2b. For instance, travel behavior changes during weekends, and LightGBM and PatchTST fail to adequately capture the distinct demand pattern observed on Saturday, July 27. In addition, LightGBM and XGBoost underestimates demand during peak hours on Monday, July 29. LSTM does not capture the morning demand spike on Monday whereas Mini-PatchTST captures such patterns effectively. However, Mini-PatchTST tends to slightly overestimate demand during weekday evening peak hours. These experimental results demonstrate the effectiveness of Mini-PatchTST in capturing routine demand patterns compared to the other models.

### 5.2 Comparison on the Hangzhou Metro Dataset

To assess the model's generalization ability and its performance relative to the existing literature, results under the simulation scenarios for Mini-PatchTST on the Hangzhou Metro Dataset, following the experimental settings of Hu (2023) and Wei et al. (2023), are presented in Table 4 and Table 5, respectively.

Hu (2023) achieved an LSTM RMSE of 82.61 for Longxiang Bridge Station Inbound, 106.02 for Longxiang Bridge Station Outbound, 228.51 for East Railway Station Inbound, and 123.83 for East Railway Station Outbound flow on the Hangzhou Metro Dataset. The simulated scenario results for the Mini-PatchTST architecture (RMSE 68.73, 59.77, 83.13, and 90.88, respectively) show lower error rates across all scenarios compared to the basic LSTM reference model. This suggests that the proposed architecture has superior predictive capability compared to a standard LSTM, particularly in handling both inbound and outbound flows at different stations.

Table 4: Comparison of the LSTM reference model (Hu, 2023) and Mini-PatchTST on the Hangzhou Metro Dataset (10-min interval)

| **Scenario** | **Model** | **RMSE** |
|---|---|---|
| Longxiang Bridge Station Inbound | LSTM | 82.61 |
| Longxiang Bridge Station Inbound | Mini-PatchTST | **68.73** |
| Longxiang Bridge Station Outbound | LSTM | 106.02 |
| Longxiang Bridge Station Outbound | Mini-PatchTST | **59.77** |
| East Railway Station Inbound | LSTM | 228.51 |
| East Railway Station Inbound | Mini-PatchTST | **83.13** |
| East Railway Station Outbound | LSTM | 123.83 |
| East Railway Station Outbound | Mini-PatchTST | **90.88** |

Wei et al. (2023) proposed the TPA-LSTM model to overcome some limitations of standard LSTM networks, which can be sensitive to computational cost and training settings. TPA-LSTM combines LSTM with a

temporal pattern attention (TPA) mechanism. This allows the model to focus on LSTM outputs from several past time steps as well as the current time step at the same time. By combining information from these different time steps, TPA-LSTM can produce more accurate predictions and better capture long-term dependencies and periodic patterns in metro passenger flow data.

However, the comparison results reported in Table 5 indicate that Mini-PatchTST achieves generally better or competitive performance compared to TPA-LSTM across different station types in the Hangzhou Metro Dataset. In the Jinjiang (Transfer Station) scenario, Mini-PatchTST clearly outperforms TPA-LSTM, achieving substantially lower MAE (26.55 vs. 34.37) and RMSE (41.31 vs. 48.47). For the Xinfeng (Intermediate Station) scenario, Mini-PatchTST attains a lower MAE (8.90 vs. 9.69) while achieving a competitive RMSE (14.03 vs. 13.14).

Table 5: Comparison of the TPA-LSTM model (Wei et al., 2023) and Mini-PatchTST on the Hangzhou Metro Dataset (10-min interval)

| **Station Type** | **Metric** | **TPA-LSTM** | **Mini-PatchTST** |
|---|---|---|---|
| Xinfeng (Intermediate Station) | MAE | 9.69 | **8.90** |
| Xinfeng (Intermediate Station) | RMSE | **13.14** | 14.03 |
| Jinjiang (Transfer Station) | MAE | 34.37 | **26.55** |
| Jinjiang (Transfer Station) | RMSE | 48.47 | **41.31** |

These results indicate that the Transformer-based Mini-PatchTST can effectively capture complex temporal patterns without relying on recurrent structures. By using patch-based representations and self-attention, the model efficiently learns long-range dependencies. As a result, Mini-PatchTST shows better performance than TPA-LSTM on most metrics. In particular, it achieves noticeably lower errors at the high-demand Jinjiang transfer station. Although TPA-LSTM attains a slightly lower RMSE at the intermediate Xinfeng station, Mini-PatchTST still provides lower MAE overall. These results indicate that Mini-PatchTST generally outperforms TPA-LSTM while remaining a simpler modeling approach.

## 6 Conclusions

In this study, we introduced Mini-PatchTST, a Transformer-based deep learning architecture for forecasting hourly passenger flow in metro networks. The proposed model was evaluated using real-world metro data from two major metropolitan cities, Istanbul and Hangzhou, demonstrating its effectiveness across different urban contexts. Additionally, this study presented a newly constructed dataset, the Istanbul Metro Dataset, for passenger flow forecasting based on the real-world data provided by the Istanbul Metropolitan Municipality. We believe this new dataset can serve as a valuable resource for future research.

The experimental results indicate that the proposed Mini-PatchTST model achieves substantially higher forecasting accuracy than traditional statistical and machine learning-based approaches, including Naive, Seasonal Naive, Window Average, LightGBM, and XGBoost. Among deep learning models, Mini-PatchTST outperforms PatchTST while demonstrating competitive performance with LSTM.

Comparisons with existing literature on the Hangzhou Metro Dataset further confirm the effectiveness of Mini-PatchTST, which achieves superior or highly competitive error metrics relative to prior studies (Hu, 2023; Wei et al., 2023). In particular, under the Transfer Station scenario at Jinjiang Station, Mini-PatchTST clearly surpasses the state-of-the-art TPA-LSTM model by yielding lower MAE and RMSE. Overall, the results demonstrate that Transformer-based architectures, particularly the lightweight Mini-PatchTST, provide strong generalization and forecasting performance for urban rail transit demand prediction.

Future work may improve Mini-PatchTST by incorporating exogenous variables, such as weather and local events, and by evaluating its performance on medium- and long-term forecasting horizons. A fully controlled benchmark across all competing methods on the Hangzhou dataset also remains for future investigation, as this study follows the experimental protocols of prior work to enable direct comparison. Beyond metro passenger forecasting, Mini-PatchTST may be applied to other time-series tasks, including traffic flow and energy demand forecasting.

## References


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

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

Jiahan Hu. Prediction of short-term passenger flow of subway based on LSTM model. *Theoretical and Natural Science*, 13(1):237–244, 2023. doi: 10.54254/2753-8818/13/20240853. URL https://doi.org/10.54254/2753-8818/13/20240853.

Liya Kang, Hongjie Liu, Ming Chai, and Jidong Lv. A lstm-based passenger volume forecasting method for urban railway systems. In Jianhua Qian, Honghai Liu, Jiangtao Cao, and Dalin Zhou (eds.), *ICRRI*, pp. 368–380, Singapore, 2020. Springer Singapore. ISBN 978-981-33-4929-2.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Lingbo Liu, Jingwen Chen, Hefeng Wu, Jiajie Zhen, Guanbin Li, and Liang Lin. Physical-virtual collaboration modeling for intra- and inter-station metro ridership prediction. *IEEE Trans. Intell. Transp. Syst.*, 23(4):3377–3391, 2022. doi: 10.1109/TITS.2020.3036057.

Kuan Lu, Menghao Huo, Yuxiao Li, Qiang Zhu, and Zhenrui Chen. Ct-patchtst: Channel-time patch time-series transformer for long-term renewable energy forecasting, 2025. URL https://arxiv.org/abs/2501.08620.

Changwen Ma, Chu Zhang, Junhao Yao, Xinyu Zhang, Muhammad Shahzad Nazir, and Tian Peng. Enhancement of wind speed forecasting using optimized decomposition technique, entropy-based reconstruction, and evolutionary patchtst. *Energy Convers. Manag.*, 333:119819, 2025. ISSN 0196-8904. doi: https://doi.org/10.1016/j.enconman.2025.119819. URL https://www.sciencedirect.com/science/article/pii/S0196890425003425.

M. Milenković, L. Švadlenka, V. Melichar, N. Bojović, and Z. Avramović. Sarima modelling approach for railway passenger flow forecasting. *Transport*, pp. 1–8, 2016. doi: 10.3846/16484142.2016.1139623.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023a. URL https://arxiv.org/abs/2211.14730.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, 2023b. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Miquel Noguer i Alonso and Rodolfo Pereira Franklin. Large language models for financial time series forecasting. *SSRN Electronic Journal*, October 2024. doi: 10.2139/ssrn.4988022. URL https://ssrn.com/abstract=4988022. Available at SSRN.

Zehong Ren, H. Gu, and Zehua Ren. Predicting passenger flow in hangzhou metro stations—using a time series arima model based on desensitized data. In *Proceedings of the 7th ICCSIE*, pp. 465–470, 2022. doi: 10.1145/3558819.3565127.

Franca Rocco di Torrepadula, Enea Vincenzo Napolitano, Sergio Di Martino, and Nicola Mazzocca. Machine learning for public transportation demand prediction: A systematic literature review. *Engineering Applications of Artificial Intelligence*, 137:109166, 2024. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2024.109166. URL https://www.sciencedirect.com/science/article/pii/S0952197624013241.

J. Siswanto, D. Manongga, I. Sembiring, and S. Wijono. Deep learning based lstm model for predicting the number of passengers for public transport bus operators. *Jurnal Online Informatika*, 9:18–28, 2024. doi: 10.15575/join.v9i1.1245.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL `https://arxiv.org/abs/1706.03762`.

Z. Wang, L. Chen, and W. Mao. Inbound passenger flow prediction at subway stations based on lbcnnm-tft. In *Proceedings of the 4th CSAE*, 2023. doi: 10.1145/3627915.3629593.

L. Wei, D. Guo, Z. Chen, J. Yang, and T. Feng. Forecasting short-term passenger flow of subway stations based on the temporal pattern attention mechanism and the long short-term memory network. *ISPRS Int. J. Geo-Inf.*, 12(25), 2023.

T. Zuo, X. Jing, G. Liu, Z. Sui, and S. Tang. Metro passenger flow prediction model based on wt transformer algorithm. In *China Automation Congress (CAC)*, 2023. doi: 10.1109/CAC59555.2023.10451767.