scientific reports

Check for updates

OPEN

A comparative study and simple baseline for travel time prediction

Chuang-Chieh Lin[1,4], Ming-Chu Ho[2,4], Chih-Chieh Hung[2]✉ & Hui-Huang Hsu[3]

Accurate travel time prediction (TTP) is essential to freeway users, including drivers, administrators, and freight-related companies, for enabling them to plan trips effectively and mitigate traffic congestion. However, TTP is a complex challenge even for researchers due to the difficulty of capturing numerous and diverse factors such as driver behaviors, rush hours, special events, and traffic incidents, etc. A multitude of studies have proposed methods to address this issue, yet these approaches often involve multiple stages and steps, including data preprocessing, feature selection, data imputation, prediction model. The intricacy of these processes makes it difficult to pinpoint which steps or factors most significantly influence prediction accuracy. In this paper, we investigate the impact of various steps on TTP accuracy by examining existing methods. Beginning with the data pre-processing phase, we evaluate the effect of deep learning, interpolation, and max value imputation techniques on dealing with missing values. We also examine the influence of temporal features and weather conditions on the prediction accuracy. Furthermore, we compare five distinct hybrid models by assessing their strengths and limitations. To ensure our experiments align with real-world situations well, we conduct experiments using datasets from Taiwan and California. The experimental results reveal that the data-preprocessing phase, including feature editing, plays a pivotal role in TTP accuracy. Additionally, base models such as Long Short-Term Memory (LSTM) and eXtreme Gradient Boosting (XGBoost) outperform all hybrid models on real-world datasets. Based on these insights, we propose a baseline that fuses the complementary strengths of XGBoost and LSTM via a gating network. This approach dynamically allocates weights, guided by key statistical features, to each model, enabling the model to robustly adapt to both stable and volatile traffic conditions and achieve superior prediction accuracy compared to existing methods. By breaking down the TTP process and analyzing each component, this study provides insights into the factors which affect prediction accuracy most significantly, thereby offering guidance and foundation for developing more effective TTP methods in the future.

Accurate travel time prediction (TTP) benefits a wide range of freeway users, including drivers, traffic administrators, and freight-related companies. With accurate travel time prediction, drivers can plan their journeys effectively in advance and mitigate traffic congestion. Traffic administrators can use travel time predictions to better manage traffic flow and mitigate congestion during peak hours so as to improve social utility. Freight companies, on the other hand, can optimize their delivery schedules, ensuring timely deliveries and reducing operational costs. However, TTP still remains a challenging task for researchers due to the myriad factors which influence travel time, such as individual driver behaviors, rush hours, special events, and unforeseen incidents like car accidents, etc.

Numerous studies have proposed various methods to tackle the issue of travel time prediction. Generally speaking, these methods often encompass multiple steps, including data preprocessing, feature selection, data imputation, and the application of prediction models. Each step is suspected to play a crucial role in the overall accuracy of the prediction, yet so far, their individual contributions have not been evaluated. The intricacy of the processes involved in TTP makes it challenging to pinpoint which specific step has the most significant impact on prediction accuracy. As a result, in this paper, we aim to shed light on this intricate problem by discussing the influence of different steps on the accuracy of travel time prediction. By examining several existing works, we break down each method into components and analyze their impact on the overall predictive performance. Through this detailed examination, we can then further propose a simple and robust baseline for TTP that effectively fuses the important components for accurate travel time prediction, ultimately guiding future research towards more accurate and reliable TTP methodologies.

[1]Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung City 202301, Taiwan. [2]Department of Management Information Systems, National Chung Hsing University, Taichung City 402202, Taiwan. [3]Department of Computer Science and Information Engineering, Tamkang University, New Taipei City 251301, Taiwan. [4]Chuang-Chieh Lin and Ming-Chu Ho: These authors contributed equally to this work. ✉email: smalloshin@nchu.edu.tw

The workflow

Figure 1 summarized the workflow how this paper discusses the impact of each component to travel time prediction. A forecast method of TTP can be either *model-based* (i.e., *model-centric*) or *data-driven*. A model-based method aims to improve the machine learning model or algorithm *without* adjusting the data. By looking into the insight and characteristics of the model, one can achieve fixed complexity and make predictions efficiently and elegantly. Data-driven methods, in contrast, take advantage of big amount of data to make decisions. Scientists developing TTP with data-centric methods concentrate on data preprocessing, including feature engineering, data cleaning and value imputation etc.

In this work, we conjecture and presume that data-driven methods in terms of data processing and feature engineering play important roles toward the TTP. Then, after the "optimal way" of data preprocessing is determined, we conduct experiments to measure the performance of different machine learning models solving the TTP. That is, we take advantage of both model centric and data-driven methods to cope with TTP (the left side of Fig. 1), and at the same time, identify the crucial factors which make the TTP be accurately predictable.

Before we dive right in the comparison of different models, feature engineering is to be reckoned with. In terms of TTP, zero value imputation in travel time is just as important as missing value imputation (upper-left side of Fig. 1). Specifically, when there is no car passing by, both the travel speed and travel time will be recorded as zero by the sensor. However, travel speed should have a negative correlation to travel time. It is unreasonable to have zero value in travel time and travel speed at the same time. In order to investigate which imputation method is the best, we conduct several experiments on real-world datasets with common imputation methods.

After the optimal imputation method is determined, we extract temporal features as different ranges of time in terms of time slots (the middle of the upper-left side of Fig. 1). As different ranges of time reflects different temporal patterns toward travel time, we investigate whether and which of these features can improve model performance. Moreover, as Fig. 2 shows, we not only see distinct patterns of weekdays and weekends in a week (e.g., the pattern in weekdays seems to be more unsteady), but also the regular intraday patterns (i.e., rush hours and normal hours). Consequently, we conduct experiments to have a view toward the influence of temporal features on different models, then we conclude the best pairs of features for model training. In addition, weather effects and other feature-related approaches are also conducted and explained (bottom of the upper-left size of Fig. 1)).

Finally, after we select optimal features for model training, we proceed with model comparisons (right side of Fig. 1). We consider two base models and five hybrid models used in the experiments for the comparisons:

(1) (Base) a conventional machine learning model as a base model: XGBoost (Xtreme Gradient Boosting).
(2) (Base) a deep learning model as another base model: LSTM (Long Short-Term Memory).
(3) (Hybrid) a hybrid model (XGBoost plus GRU): linear regression from the outcome of XGBoost and GRU (gated recurrent unit)[1].

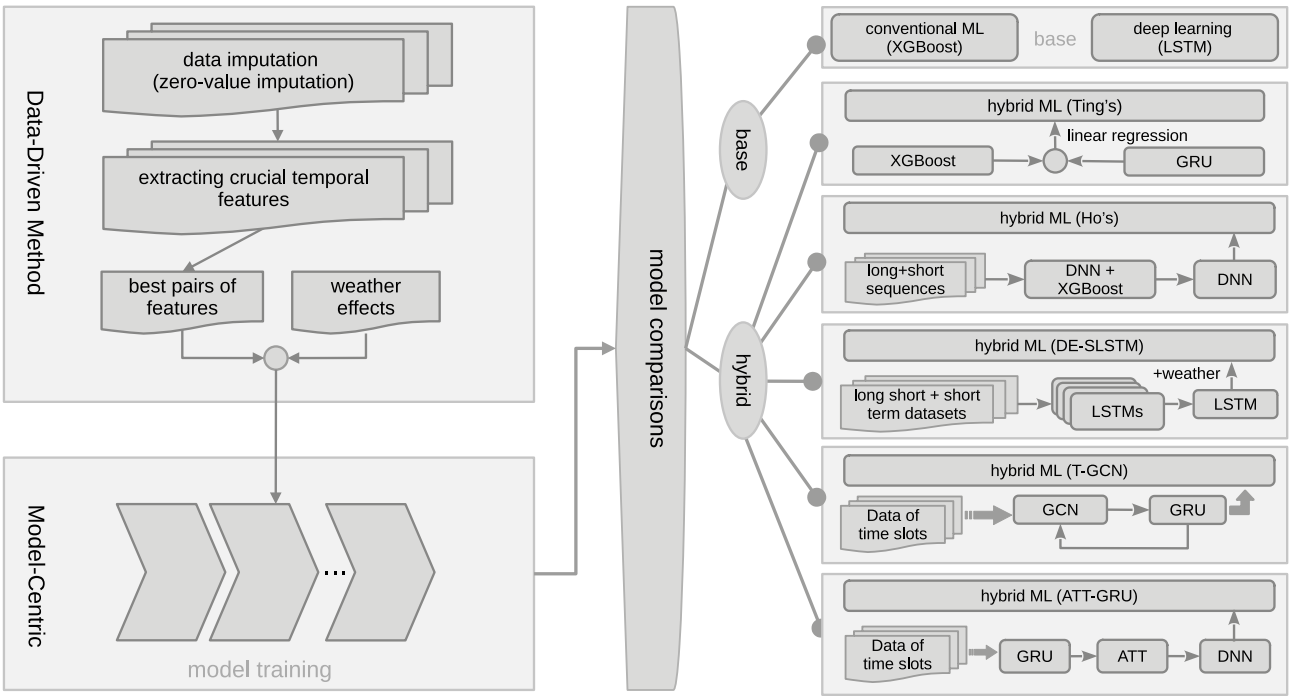

Fig. 1. Workflow demonstration of our contributions. XGBoost: extreme gradient boosting; LSTM: long short-term memory; DNN: deep neural network; GRU: gated recurrent unit; GCN: graph convolutional network; T-GCN: temporal GCN; ATT: self-attention mechanism.

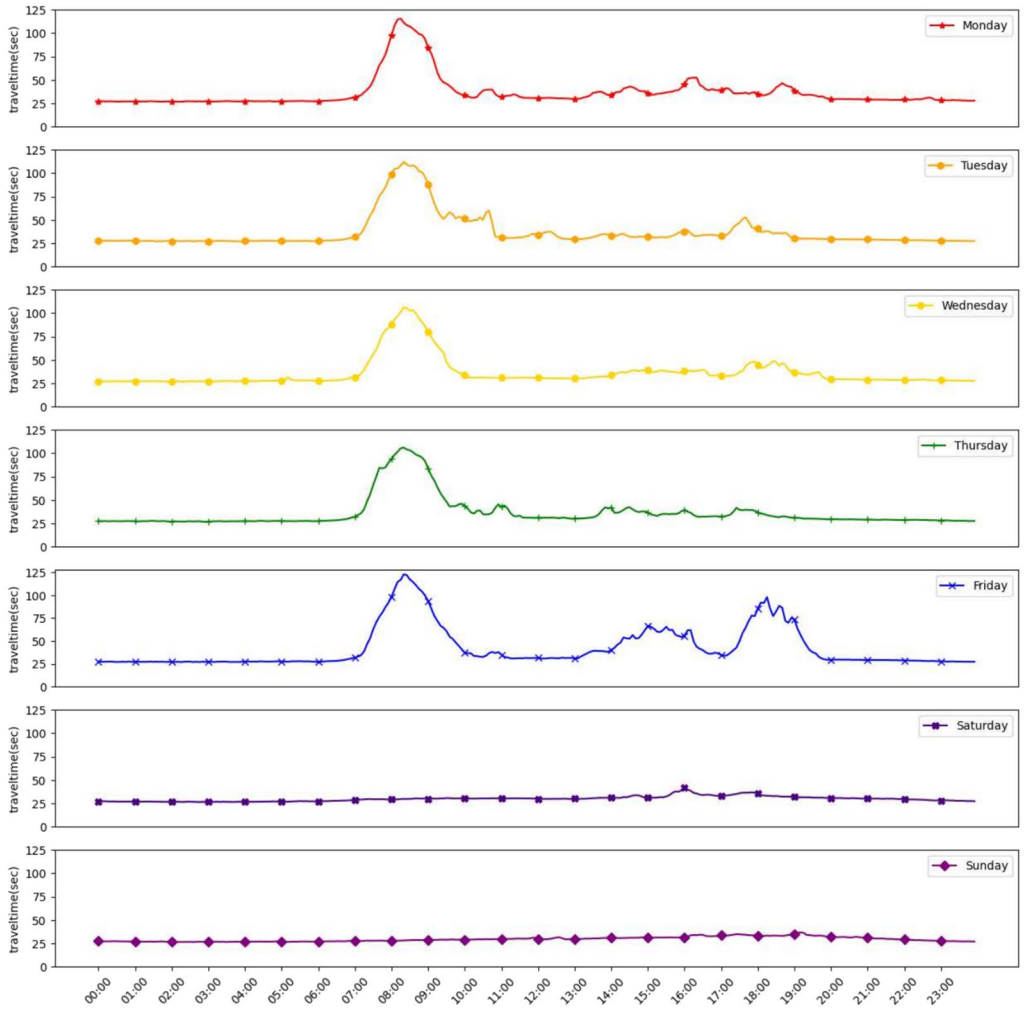

Fig. 2. Example of trend of travel time from Monday to Sunday (from January 15 th to January 21 st in 2018). The x-axis represents time-slots and the y-axis represents the travel time.

(4) (Hybrid) a hybrid model (DNN plus XGBoost): long and short sequences are fed into a XGBoost model and a DNN (deep neural network), respectively, and then the outcome is fed into the another DNN to get the output[16].
(5) (Hybrid) a hybrid model (stacked LSTMs): long short and short term datasets are fed into stacked LSTMs and the outcome together with the weather datasets are fed into a DNN to get the output[5].
(6) (Hybrid) a spatial and temporal hybrid model (GCN plus GRU): data of consecutive time slots are fed into a recurrent neural network which consists of a GCN (graph convolutional network) and a GRU[17].
(7) (Hybrid) a hybrid model combining self-attention and GRU (ATT-GRU): a sequence of historical segment travel time is fed into the attention-based GRU, and then predict the travel time at some time horizon[20].

We use real-world datasets to conduct our experiments, which include datasets from the Ministry of Transportation and Communications (MOTC) in Taiwan and the Caltrans Performance Measurement System (PeMs) District 7 in the state of California of the US.

Our contributions

We briefly summarize our contributions as follows.

- *Component-wise Analysis of Travel Time Prediction Methods.* The paper conducts a detailed analysis of various travel time prediction methods by breaking down each method into its constituent components. This approach helps to identify which specific steps, such as data preprocessing, feature selection, data imputation, and prediction models, significantly impact the accuracy of travel time predictions.
- *Evaluation of Data-driven and Model-centric Approaches.* The study leverages both data-driven and model-centric methods to tackle travel time prediction. By optimizing data preprocessing and feature engineering, and then evaluating the performance of different machine learning models, the paper provides a comprehen-

sive comparison of these approaches and identifies the crucial factors that contribute to accurate travel time predictions.
- *Experiments with Real-world Datasets.* The research utilizes real-world datasets from the Ministry of Transportation and Communications (MOTC) in Taiwan and the Caltrans Performance Measurement System (PeMs) District 7 in California. By conducting experiments with these datasets, the paper validates the proposed methods and demonstrates their practical applicability in improving travel time prediction accuracy in real-world scenarios.
- *Proposal of a simple and robust Baseline for TTP.* Based on our analysis, we propose a new dynamic fusion baseline that fuses XGBoost and LSTM via a gating network-offering a robust, adaptive prediction framework that outperforms existing models in diverse traffic scenarios.

The remaining sections of this paper are organized as follows. Sect."Related work" reviews literature related to TTP, including model constructions and methods of comparisons. Sect."Preliminary" illustrates the structure of our comparison and the configuration of the adopted models. Experimental results are presented in Sect."Imputation methods description". Finally, we conclude our work in Sect."Base and hybrid model frameworks" with discussions.

Related work

Categories of travel time prediction

Travel Time Prediction (TTP) can be divided into two primary categories: short-term TTP and long-term TTP. Short-term TTP focuses on predicting travel time within 5 to 30 minutes, and it is particularly useful for real-time decision-making, such as selecting the most efficient route to avoid congestion. On the other hand, long-term TTP refers to travel time predictions at least an hour in advance, which are valuable for planning purposes, such as optimizing delivery schedules. However, long-term TTP is more challenging due to the uncertainty of future conditions, making it harder to accurately predict travel time trends.

Both short-term and long-term TTP have distinct perspectives that aim to maximize the accuracy of travel time predictions. Short-term TTP is predominantly influenced by real-time traffic conditions, rather than periodic features such as the day of the week or time of day. For example, Ting et al.[1] developed a hybrid model using traffic volume, average travel time, and speed to improve short-term predictions. Qiao et al.[2] also used features like travel speed and weather conditions for training their model. Temporal features, such as month and travel time change, have also been integrated into some short-term TTP models[3]. Long-term TTP, being further removed from the current traffic conditions, relies more heavily on temporal patterns to enhance accuracy. The time gap between the current and target time introduces additional complexities in prediction. For instance, Chen et al.[4] and Chou et al.[5] utilized various temporal variables to better capture travel time trends. These studies demonstrate the importance of considering time-based attributes, as relying solely on historical travel trends is insufficient for accurate long-term predictions. In[6], Kandiri et al. proposed a two-stage feature selection method for TTP, which significantly reduced the number of features and computational costs, while still improved the prediction accuracy. Their data-driven approach partially motivated our work in the sense of jointly investigating data-driven method and model-centric method for TTP.

Travel time prediction models

Parametric models, such as the Autoregressive Integrated Moving Average (ARIMA), assume that travel time follows a regular trend. Billings and Yang[7] applied ARIMA for one-step-ahead predictions, while Qiao et al.[2] identified its limitations, particularly in handling peak traffic periods. Similarly, the Kalman Filter (KF) has been applied to future forecasting, treating travel time as a moving point for accurate predictions[8]. However, parametric models are constrained by fixed parameters and may suffer from underfitting in complex travel time patterns. This limitation has led to the adoption of non-parametric models, which are better suited to capturing higher-order dependencies.

Machine learning and deep learning methods have become prominent in addressing the challenges of TTP. Ensemble learning techniques, which combine the outputs of weak learners, are effective for generating objective and accurate predictions. Yu et al.[9] introduced a random forest (RF) model based on near neighbors (RFNN) for bus travel time prediction. Random forests leverage bagging algorithms to overfit weak learners on specific datasets, improving overall prediction accuracy through weighted voting. Qiu and Fan[3] found that RF outperformed decision trees, XGBoost, and LSTM in their experiments. Gradient Boosting (GB) and its variants, such as Gradient Boosting Decision Tree (GBDT) and XGBoost, are also widely used for TTP. Chen et al.[4] applied the gradient boosting method for long-term TTP, incorporating a Fourier filter to denoise data, though the filter did not significantly improve predictions. XGBoost, known for its efficiency and high performance, has also been employed in freeway travel time prediction[10], and Ahmed et al.[11] demonstrated its superiority over other models like LightGBM in multiple real-world datasets. Except LSTM, other deep learning approaches, such as MLP[12,13] (multilayer perceptron), CNN[14], deep-stacked auto-encoder[15], have been addressed for TTP. Hybrid models have gained traction in recent years, aiming to capture diverse patterns from data to enhance TTP accuracy. Ting et al.[1] combined GRU and XGBoost via linear regression, while Ho et al.[16] designed a two-phase hybrid model integrating DNN and XGBoost. Chou et al.[5] developed a hybrid model using stacked LSTMs for long-term TTP, incorporating weather effects, though the significance of these effects remains unclear. Zhao et al.[17] proposed the Temporal Graph Convolutional Network (T-GCN), which simultaneously captures temporal and spatial dependencies using a combination of Graph Convolutional Networks (GCNs) and GRU, demonstrating promising results. As RNNs, such as LSTM and GRU, basically assign the same weights to the hidden states, the attention mechanism[18], which is widely used in the transformers for their ability of flexibly reweighting the network weights, has been investigated for the TTP[19,20]. Recently, Chughtai et al.[20]proposed the ATT-GRU

model which added an attention layer after the GRU (cf[19]. the LSTM was used instead) in order to deal with the relationship between distinct positions in the travel time sequence.

Deep learning techniques for time series forecasting

The development of novel deep learning architectures has significantly enhanced the accuracy and efficiency of time series forecasting models[21]. introduced the N-BEATS architecture, a feedforward neural network with a basis expansion mechanism that achieved state-of-the-art performance across several benchmarks. Similarly[22], proposed the Informer, a transformer-based model optimized for long-sequence forecasting, which utilizes a sparsity mechanism to improve computational efficiency. Transformers have also been enhanced for interpretability and multi-horizon forecasting, as demonstrated by[23], who developed the Temporal Fusion Transformer (TFT). Further advancements include the Autoformer[24], which introduced a decomposition-based approach to extract seasonal and trend components for long-term forecasting. Hybrid models, combining statistical techniques and deep learning, have been increasingly explored[25]. developed the ES-RNN, which integrates exponential smoothing with RNNs, achieving first place in the M4 competition. Probabilistic methods, such as DeepAR[26], have also gained traction by providing uncertainty estimates for forecasts.

Recent studies have emphasized enhancing generalizability across diverse scenarios. For instance[27], incorporated volatility modeling with GARCH and signal decomposition for non-stationary time series[28]. reviewed graph neural networks for spatiotemporal forecasting, particularly in traffic applications, highlighting their ability to model spatial dependencies effectively. Several surveys have provided systematic overviews of advancements in deep learning for time series forecasting[29]. outlined foundational methods and key challenges, while[30] highlighted emerging trends, such as foundation models and their potential for transfer learning in related forecasting problems.

More recent works have further pushed the envelope of forecasting methodologies. For example, iTransformer[31] inverts the standard embedding strategy by treating individual time points as distinct variate tokens, thereby capturing richer multivariate correlations, while Non-Stationary Transformers[32] employ de-stationary attention mechanisms to handle evolving data distributions more effectively. On the graph-based front, FourierGNN[33] rethinks multivariate time series forecasting from a pure graph perspective by representing series as hypervariate graphs and applying Fourier space operations to capture unified spatiotemporal dynamics. The comprehensive survey on GNNs for time series[34] further categorizes these approaches in forecasting, classification, imputation, and anomaly detection. In addition, practical applications in transportation have been explored: Wang et al.[35] introduced a deep neural network model for travel time estimation, and Derrow-Pinion et al.[36] demonstrated effective ETA prediction using graph neural networks in Google Maps. Most recently, ForecastGrapher[37] has been proposed to recast multivariate forecasting as a node regression task on graphs, thereby bridging spatial and temporal modeling.

Deep learning has revolutionized time series forecasting, offering unparalleled capabilities in capturing complex patterns and enhancing forecasting accuracy. From novel architectures to hybrid and application-specific models, the advancements summarized in this review demonstrate the field's rapid evolution. Nevertheless, challenges such as improving interpretability, addressing data scarcity, and achieving robust generalization remain open for future research.

Preliminary

In this section, we lay the groundwork for addressing the Travel Time Prediction (TTP) problem by defining the core problem and outlining the key features and models involved. The focus is on identifying important predictive variables and optimal modeling strategies. By setting a future time period as the target, we aim to predict travel time based on historical data captured in a sliding time window. This approach helps us capture temporal patterns and trends in travel time, which are critical to developing accurate prediction models.

Problem definition

To address the Travel Time Prediction (TTP) problem, we aim to identify the most significant features and optimal models. During the data preprocessing phase, we define a future period, denoted as t^* (e.g., one hour later), as the target prediction time, and the travel time at t^* as T_{t^*}. Assuming the current time period is t, the feature vector associated with t is x_t. Each time period represents a 5-minute interval, and the target period is set as $t^* = t + 12$, with the goal of predicting T_{t+12}. The feature vector x_t consists of distinct features, $(A_t, B_t, C_t, \ldots, N_t)$. To capture travel time trends, we employ a sliding time window, i.e., $(x_{t-\ell}, x_{t-\ell+1}, \ldots, x_t)$, where ℓ is the number of time slots used to identify hidden temporal patterns over an unknown period. The objective function is formulated as:

$$f\left(\bigcup_{i=0}^{\ell}(A_{t-\ell+i}, B_{t-\ell+i}, C_{t-\ell+i}, \ldots, N_{t-\ell+i})\right) = T_{t+12},$$

where f represents the model and ℓ refers to the number of time slots related to the hidden temporal pattern within the TTP.

Data preparation

This study made use of two datasets from different countries from January to July in 2018, capturing the consistency of TTP. We consider a 5-minute interval as the time horizon unit for both datasets. The Taiwan dataset was collected (see Fig. 3) from Freeway Bureau, MOTC in Taiwan (https://tisvcloud.freeway.gov.tw/history/TDCS), where the general information we had captured includes the traffic flow, average travel time,

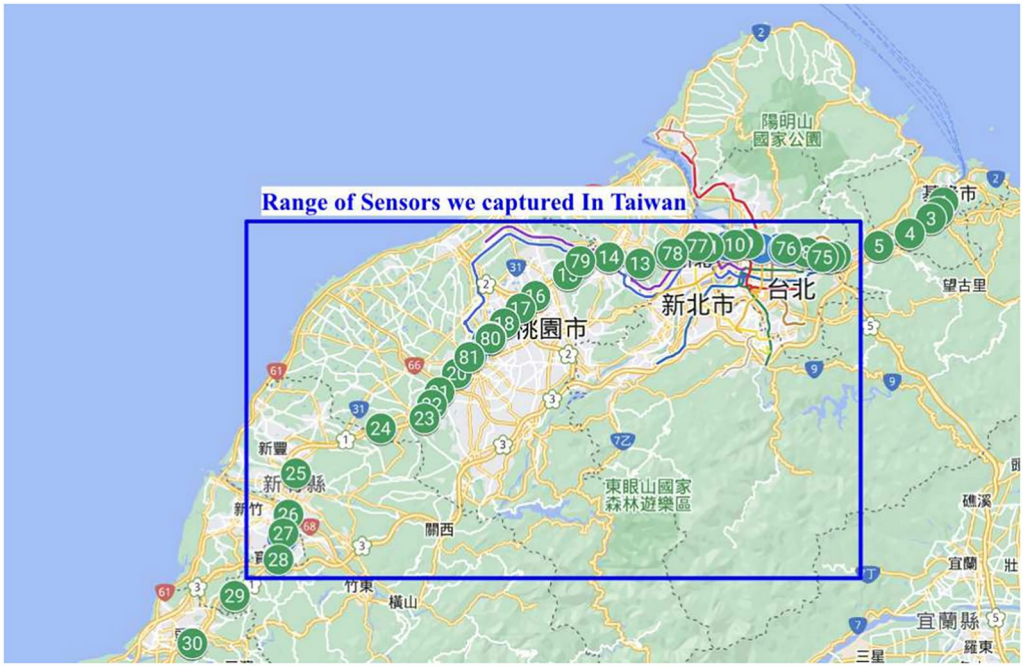

Fig. 3. The 69 routes in Taiwan dataset (Image © Google). Each green circle with number indicates a route between an upstream position and a downstream position.

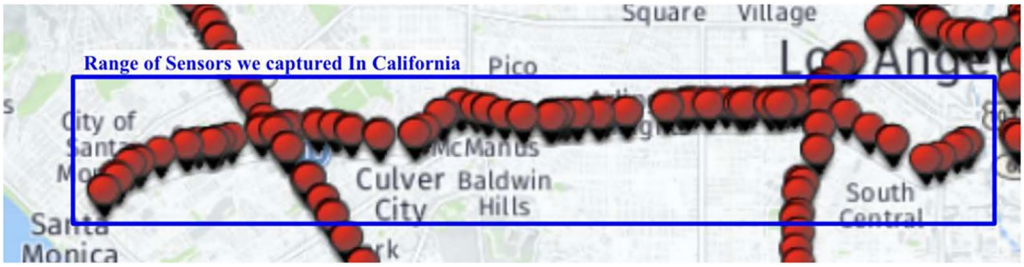

Fig. 4. The 81 routes in California dataset (https://pems.dot.ca.gov/). Each red circle indicates a route between an upstream position and a downstream position.

average travel speed, and traffic flow between the sensor at the start point and the one at the end point. The segment between two sensors defines a route segment. In our study, we conducted experiments on the freeway from Taipei to Hsinchu, including 69 routes. Additionally, we collected the California dataset (see Fig. 4) from PeMS, state of California (https://pems.dot.ca.gov/). In the California dataset, we fetched general information from respective sensors, which includes average traffic flow and average travel speed. In order to extract the travel time, we divided the distance of a route by the average travel speed. Totally 81 routes (two sensors formed a route, illustrated in Fig. 5) were selected in District 7, Los Angeles County. In the following experiments, these two datasets were used to build different models, and then we will compare their performances.

Imputation methods description

In this section, we present various methods to handle missing or zero values in our dataset, which are considered noisy due to the inverse relationship between speed and travel time. Zero values in the dataset would be regarded as noisy, for speed is inversely proportional to travel time. From the view of the sensor, that no car passing by is treated as no travel time data and no travel speed data as well. In this way, they were recorded as zero values. Therefore, we adopted several methods to impute zero values. The effectiveness of the imputation will also be investigated.

Chou's imputation (interpolation)

In[5], Chou et al. first filtered outliers and then used interpolation to fill the missing values among continuous data. If a missing value still existed, they adopted the data at the same time point a week ago or later for the imputation. Finally, the average value was used to impute the remaining missing values.

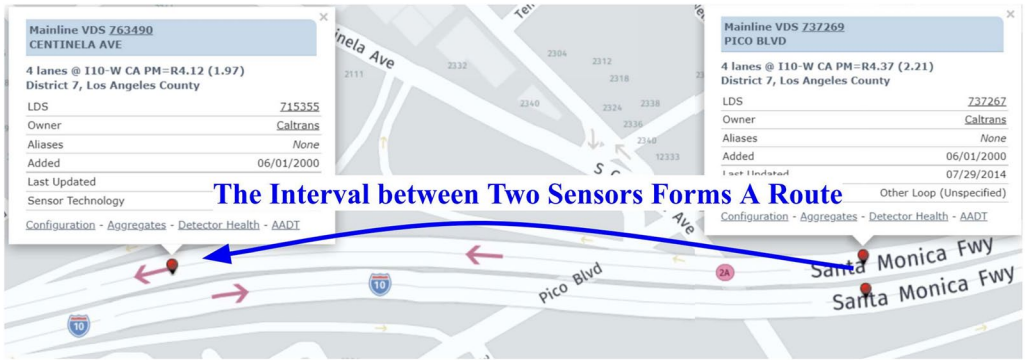

Fig. 5. Detail information about two sensors forming a route.

Max imputation

When there is no car passing by, the route is clear and drivers can maximize their driving speed till speed limit. In this way, we captured the speed limit from every route and imputed zero value at the speed column. Afterwards, we average the travel time whenever their speed reaches the speed limit.

Denoising AutoEncoder (DAE)

Deep learning methods, such as AutoEncoders, have been adopted to address the issue of missing value imputation[38,16]. Inside the AutoEncoder network, the information is compressed at a bottleneck so as to capture meaningful and succinct representation of the data. In[38], DAE was used for imputation. It added some noise to data in order to make the model be more robust. Assume that each data point x represented as $\{a, b, c\}$, where a, b, c symbolizes the features. Then, one feature is marked randomly, e.g., $\{a, b, \text{mask}\}$ for which c is randomly chosen to be masked. Next, let $\tilde{x}$ be the randomly masked input, then feed it into DAE and get $\hat{x} = \{a, b, \hat{c}\}$ as the predicted outcome. Finally, apply the mean-squared error (MSE) function to calculate loss between x and $\hat{x}$ which is used to update the model.

Base and hybrid model frameworks

In this section, we describe the core models used in our approach to travel time prediction. We begin by outlining the base models, including XGBoost and LSTM, which are widely used for their effectiveness in handling structured data and sequential patterns, respectively. Following this, we introduce several hybrid models that combine the strengths of various techniques to improve prediction accuracy. Each model's architecture and functionality are explained in detail, providing a comprehensive understanding of their role in solving the travel time prediction problem.

Base model description

In order to represent the influence of different methods of feature engineering, we proposed ensemble learning model XGBoost and deep learning model LSTM to measure the effectiveness.

XGBoost ensemble earning model

XGBoost is famous for its features of parallel computing, optimal performance and large-amount adjustable parameters. Through the loss of previous models, it constructed the next weak model to minimize its loss. To illustrate the algorithm of XGBoost, we first define the object: $\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$. In this way, we can realize that the outcome is summed by the functions f_k's (i.e., *K decision trees as weak learners*). After that, the objective function of XGBoost will be

$$\text{obj} = \sum_{i=1}^{n} \text{loss}\ (y_i, \hat{y}_i) + \sum_{k=1}^{K} R(f_k). \tag{1}$$

The objective function (Eq. (1)) can be divided into two parts: the loss function part and the regularization term. In the loss function part, its default loss function is the squared error: $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, yet we can specify specific loss functions to address different issues. In the complexity control part, it controls how many samples a node can contain, depths of a tree and number of nodes in a tree. In addition, XGBoost exploited the Taylor's expansion up to the second order to approximate and simplify the objective function.

The second-order Taylor's approximation can be formulated as Eq. (2).

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)(\Delta x)^2. \tag{2}$$

Then we let $f(x_i) = \text{ loss } (y_i, \hat{y_i}^{k-1})$. Therefore, $f(x_i + \Delta x_i) = \text{ loss } (y_i, \hat{y_i}^{k-1} + f_k(x_i))$. In this function, $\hat{y_i}^{k-1}$ represents the sum of predictions by the previous $k-1$ trees on x, and $f_k(x_i)$ represents the prediction (i.e., Δx) of the current k-th tree. In this way, we can transform the objective function in the form:

$$\text{obj}_k = \sum_{i=1}^{n} \text{ loss } (y_i, \hat{y_i}^{k-1} + f_k(x_i)) + \sum_{k=1}^{K} R(f_k). \tag{3}$$

The objective function (Eq. (3)) can be substituted by Taylor's expansion as $\sum_{i=1}^{n}(\text{ loss } (y_i, \hat{y}_i^{k-1}) + \partial_{\hat{y}_i^{k-1}} \text{ loss } (y_i, \hat{y}_i^{k-1}) f_k(x_i) + \frac{1}{2}\partial^2_{\hat{y}_i^{k-1}} \text{ loss } (y_i, \hat{y}_i^{k-1}) f_k^2(x_i)) + R(f_k)$. The first term $\text{ loss } (y_i, \hat{y}_i^{k-1})$ is a known term since it was calculated by previous trees. Therefore, it can be omitted from the objective. Then, we realize that the first derivative and second derivative of the loss functions are regarded as residuals so as to offer XGBoost information about the result of previous trees. Finally, by adding the predictions from all the trees constructed by the XGBoost algorithm, it hopefully provides more accurate results.

Long short-term memory (LSTM) model

The LSTM is proposed to resolve issues in the traditional RNN. Despite the fact that an RNN can store former output as memory and influence next input by loading the memories from a hidden layer, RNN is unable to memorize long-term memory after feeding several inputs. Instead, the LSTM with three distinct gates aims to keep important information in the memory for the model. Figure 6 illustrates the structure of the LSTM.

1. Z_t(from the output of the previous layer) goes through the `tanh` activation function.
2. The input gate controls the degree of influence from Z_t by using `sigmoid` activation function which outputs a value in [0, 1].
3. After going through the input gate, the forget gate uses the `sigmoid` function to control whether to clean the memory cell or not. Then, it sums the value from the previous layer and updates the memory cell.
4. The output gate controls the degree of output value by multiplying the value from the output gate (in [0, 1]) and the value from the previous layer.

In this way, LSTM can not only recognize the importance of long-term memory by controlling distinct gates, but also capture the pattern of adjacent inputs. Due to the promising characteristics of the LSTM, we adopted it as one of the base models.

We apply XGBoost and LSTM as the prediction models to compute the TTP accuracies so as to measure the performance of the methods used in the feature engineering and data preprocessing phase (e.g., data imputation and temporal feature extraction).

Hybrid model frameworks

In the model comparison phase, we adopt five different hybrid models depicted as follows. We would investigate which model performs the best.

Ting's GRU-XGBoost hybrid model

Ting's hybrid model has two phases[1]. In the first phase, it used data in the form of sliding windows as input going through GRU and XGBoost respectively. In the second phase, Linear regression is constructed to capture and leverage the comprehensive information from the predictions made by GRU and XGBoost. The captured features include One-station Traffic Volume, Two-Station Traffic Volume, Average Travel Time, and Average Traffic Speed. They used the Denoising AutoEncoder[38] to impute the zero values in Average Travel Time and Average Traffic Speed. Figure 7 illustrates the structure of Ting's Hybrid Model.

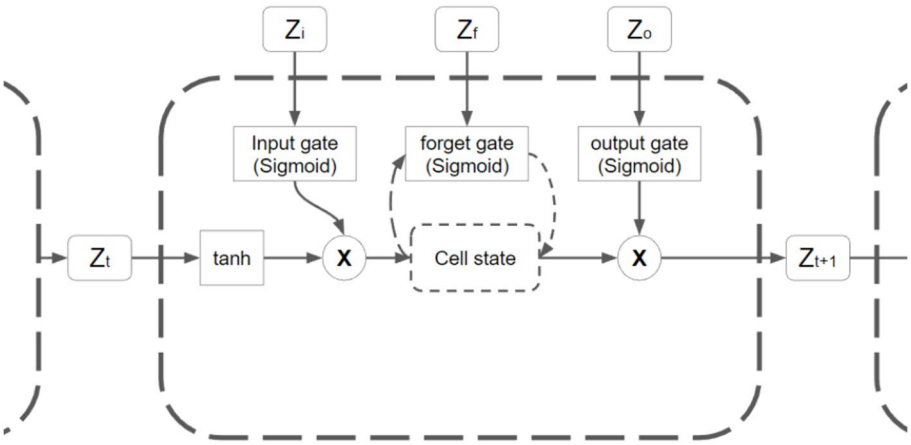

Fig. 6. Structure of the LSTM.

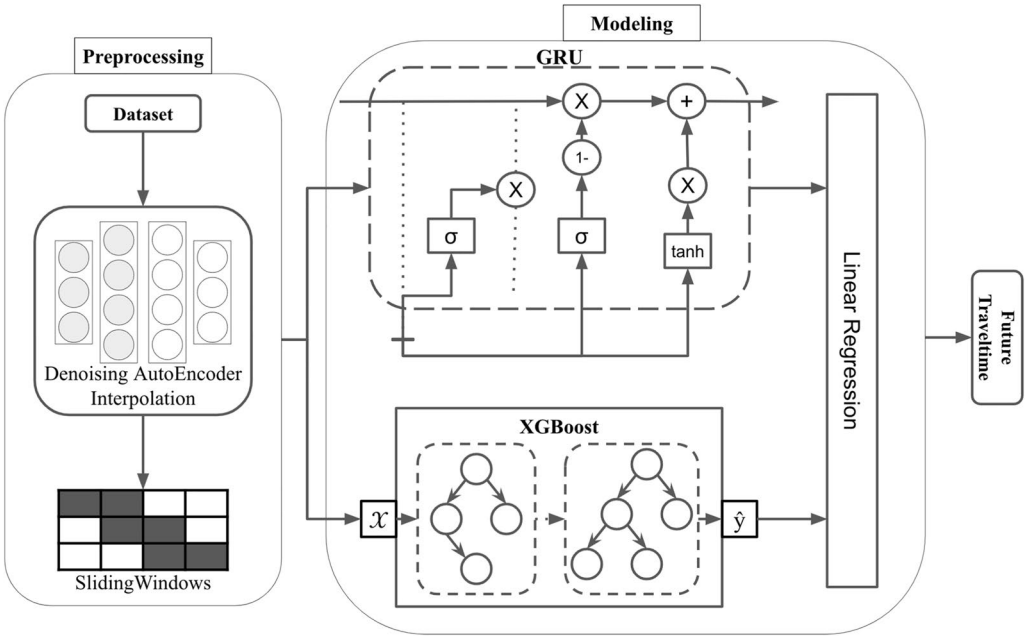

Fig. 7. Structure of Ting's Hybrid Model[1].

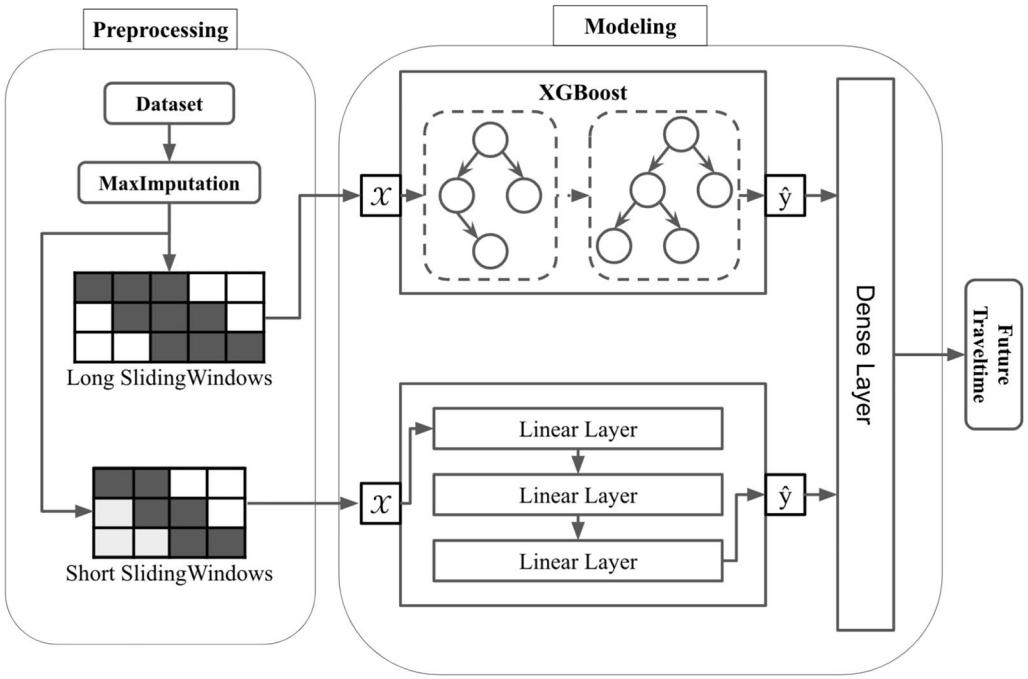

Fig. 8. Structure of Ho's Hybrid Model[16].

Ho's DNN-XGBoost hybrid model

This section explores Ho's hybrid model[16], which operates in two distinct phases. In the first phase, both a deep neural network and an XGBoost model are developed. Long and short sequences of input data are then fed into these models separately to capture different patterns within the dataset. The results from these two models are processed by a second deep neural network. Features such as average travel time, average traffic speed, day of the week, and hour of the day are utilized. When imputing missing values, the speed limit is used as a proxy for both average traffic speed and travel time on the freeway. Figure 8 provides a visual representation of Ho's Hybrid Model.

Deep ensemble stacked LSTM (DE-SLSTM)
The structure of DE-SLSTM (i.e., Deep Ensemble Stacked Long Short Term Memory) is made up of two predictors, a non-peak predictor and a peak predictor, to capture information at the moment of a peak period[5]. They stacked different LSTMs in each predictor by three kinds of dataset: long short-term dataset, containing the information of last week, short-term dataset, containing the information of last hour and long-term dataset, containing the information of the historical dataset. The weather dataset will be addressed before the model outputs. After calculating the proportion of peak periods and non-peak periods, the prediction will be made in proportion to their importance. Average travel speed and average travel time from MOTC in Taiwan, and generated temporal attributes such as month, day of week, holiday, time slot and peak, were extracted. Interpolation is applied to impute missing data. Figure 9 illustrates the structure of DE-SLSTM.

Temporal graph convolutional network (T-GCN)
A T-GCN (i.e., temporal graph convolutional network) is a hybrid model consisting of two parts: GCN and GRU. GCN (i.e., the graph convolutional network) captured the topological structure of the road network and GRU captured the time trend of travel time[17]. First, GCN is used to make every route as a node and aggregate the information from neighboring nodes. Then, every node containing information from their surrounding nodes in a specific length of time steps will be fed into GRU so as to get the trend of travel time. After the GRU phase, the prediction of travel time will be made. Average Travel Speed is used as the feature and interpolation is used to impute missing values. Figure 10 illustrates the structure of the T-GCN.

Attention-based gated recurrent unit (ATT-GRU)
Chughtai et al.[20] proposed the ATT-GRU (attention-based gated recurrent unit) model, which is also a hybrid model consisting of two parts: a simple attention mechanism and a GRU. This model can leverage the benefit of GRU for the short-term period TTP task, and moreover, re-weight the weights in the network by a simple self-attention mechanism. See Fig. 11 for the sketch of such a hybrid model. Similar approach by replacing GRU by LSTM was also proposed[19], though GRU is more efficient in terms of faster training process with fewer parameters than LSTM.

Experiment

Evaluation metric

We adopted MAE (i.e., Eq. (4)) and RMSE (i.e., Eq. (5)) to evaluate the performances. MAE measured the average absolute difference between the predicted value (i.e., $\hat{y}_i$) and the actual value (i.e., y_i). That is,

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_i - \hat{y}_i|, \tag{4}$$

where n denotes the number of data points. RMSE computes the square root of the average differences between the predicted value and the actual value. Because of the effect of square root, large differences would lead to larger residuals than MAE did.

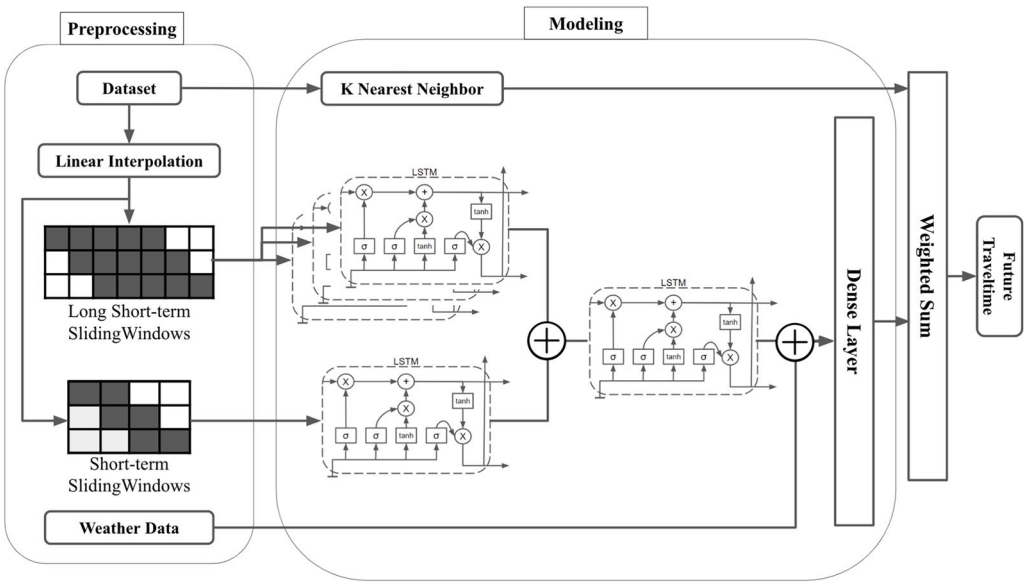

Fig. 9. Structure of DE-SLSTM[5].

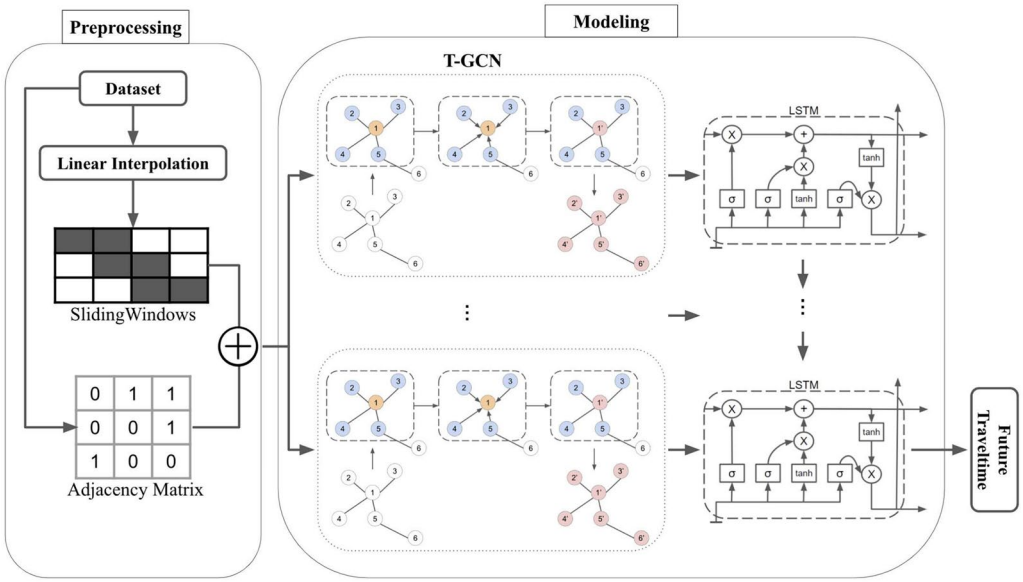

Fig. 10. Structure of T-GCN[17].

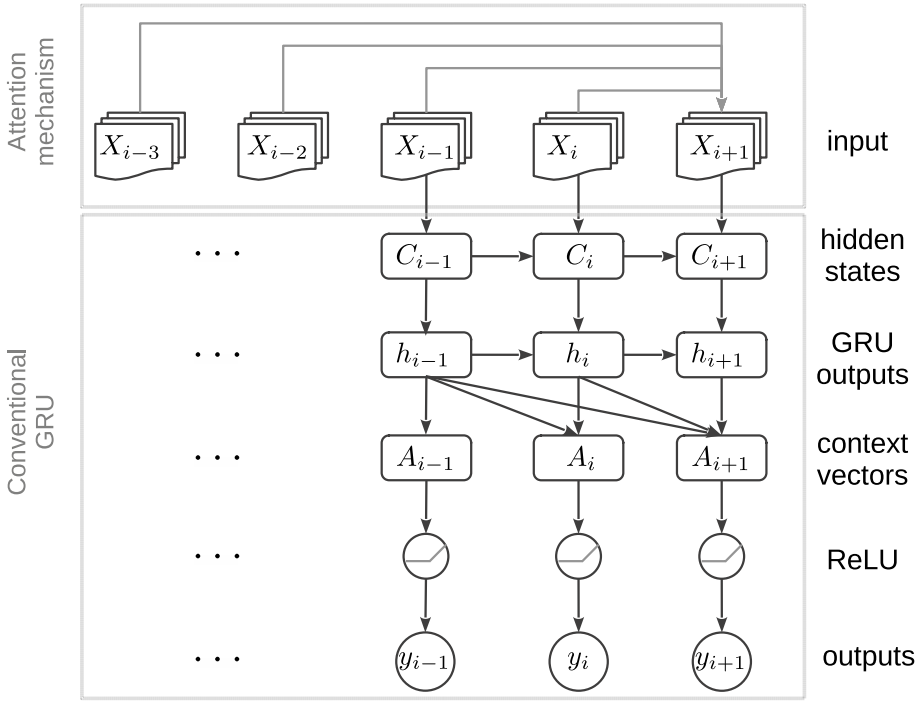

Fig. 11. Structure of ATT-GRU[20].

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_i - \hat{y}_i)^2}. \tag{5}$$

We adopted both of them in the data preprocessing phase and the model comparison phase. In addition, since we adopted several routes to measure the performances of models, we measure MAE and RMSE in mean and median in order to find out objective outcome.

Taiwan	Max Imputation	Chou's Imputation	DAE Imputation
Mean MAE	16.766	**16.762**	16.965
Median MAE	14.183	**13.551**	14.244
Mean RMSE	47.422	**47.168**	47.646
Median RMSE	**37.751**	38.907	37.748
California	**Max Imputation**	**Chou's Imputation**	**DAE Imputation**
Mean MAE	**4.143**	**4.143**	4.150
Median MAE	3.152	3.152	3.152
Mean RMSE	8.244	**8.237**	8.250
Median RMSE	**6.318**	**6.318**	6.365

Table 1. TTP performance by XGBoost using different missing value imputation methods.

Taiwan	Max Imputation	Chou Imputation	DAE Imputation
Mean MAE	**16.940**	17.024	17.184
Median MAE	**12.767**	13.749	13.631
Mean RMSE	**45.034**	45.243	45.152
Median RMSE	**35.005**	35.835	35.027
California	**Max Imputation**	**Chou Imputation**	**DAE Imputation**
Mean MAE	**4.420**	4.462	4.453
Median MAE	**3.290**	3.450	3.452
Mean RMSE	**8.453**	8.468	8.461
Median RMSE	6.608	**6.573**	6.654

Table 2. TTP performance by the LSTM using different missing value imputation methods.

Data preprocessing phase

In this phase, we adopted XGBoost and LSTM to measure the performances of different methods in data preprocessing. We intend to use these two models to generate a global view toward this phase.

Analysis of missing value imputation

In this paragraph, we compare the performance of TTP using XGBoost and the LSTM with different missing value imputation methods, and we would like to know which missing value imputation method leads to the best prediction results. We adopted Chou's imputation (Interpolation), Max Imputation and DAE. In DAE, we used default parameters to tune the model. From Table 1 and 2, we can see that Chou et al.'s imputation leads to the smallest MAE and RMSE by XGBoost and LSTM, yet Max Imputation leads to comparable performance. Note that, as shown in Table 1 and 2, using DAE with default parameters for the missing value imputation leads both XGBoost and the LSTM to the worst performance.

Analysis of sliding windows

Sliding windows sequenced the data in order to capture time trends in a specific period (see Fig. 12). Every time slot is in the unit of five minutes. We compare six sizes of time sliding windows: 1 time slot (the current time slot), 6 time slots (from current time slot to 30 minutes before), 12 time slots (from current time slot to one hour before), 24 time slots, 36 time slots and 48 time slots.

Figure 13 shows the comparison of each size of time sliding windows and each diagram adopted different evaluation metrics in the Taiwan dataset. We find out the local minimum when the size of time sliding windows is 24 in most of the diagrams. Thus, we realize that in the Taiwan dataset, when we capture the data from two hours before, the model is able to predict travel time most accurately. Note that more model training time is required when a larger sliding window is applied and more features are involved. We will explain the relationship between size of time sliding windows and time consuming later.

Figure 14 presents the comparison among different evaluation metrics in the California dataset. We observe that when the size of sliding windows increases, the predicted outcome made by XGBoost and the LSTM leads to smaller MAE and RMSE, which is different from the results on the Taiwan dataset. However, it takes more time to train the models for larger time sliding windows due to larger data volume and dimension, hence there is a trade-off at choosing the time sliding window size.

Therefore, we illustrated the relationship among average evaluation metrics (i.e., $(\mathrm{RMSE} + \mathrm{MAE})/2$), size of time sliding windows and training time. In Fig. 15, the red line represents average MAE and RMSE, and the green line represents training time for each size of time sliding windows. In the diagram, we observe that when the size of sliding windows increases to 24, training time will increase significantly. And as the diagram we compare in the Taiwan dataset, when the size of time sliding windows is 24, it attains the optimal results. Hence, we adopted sliding windows of 24 time slots as the optimal size of time sliding windows.

Fig. 12. Schema of time sliding windows.

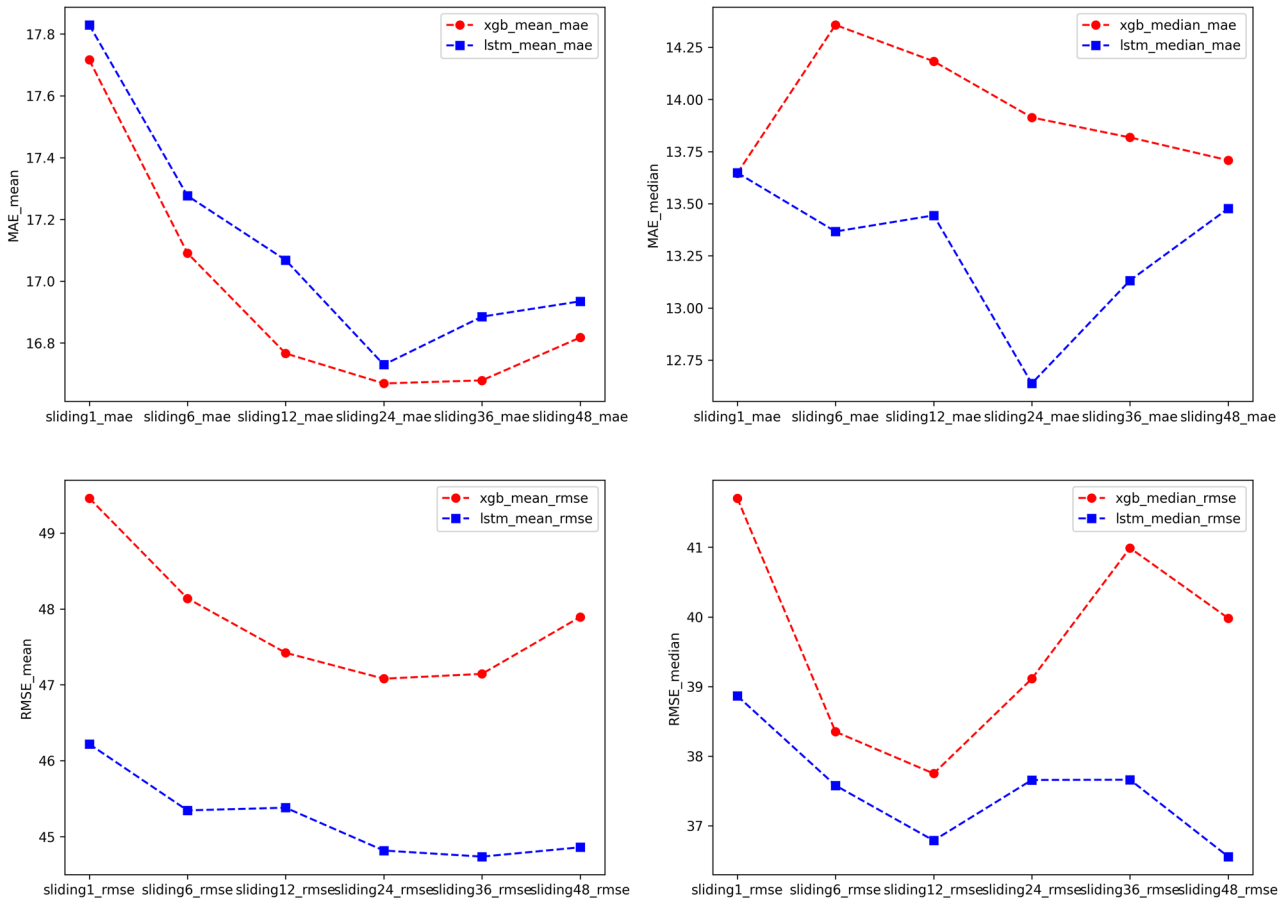

Fig. 13. TTP performance of different sliding window sizes on the Taiwan dataset.

Analysis of weather feature (Taiwan dataset)
There are many features related to weather, including precipitation, blowing rate, wind direction and temperature, etc. However, adding features would increase the model complexity which could lead to overfitting of the model. Therefore, the information provided by extra features shall surpass the effects from increasing complexity. Hence, it is interesting to know whether weather features are important enough to improve model performance. We adopted the Taiwan dataset as an illustrating example. Since most weather stations are not at the same location as the sensors on the freeway, we identify the nearest weather station of the *smallest Euclidean distance* for each sensor. The weather stations (see Fig. 16) are orange balloons and the stations on the freeway are green balloons numbered from 1 to 69.

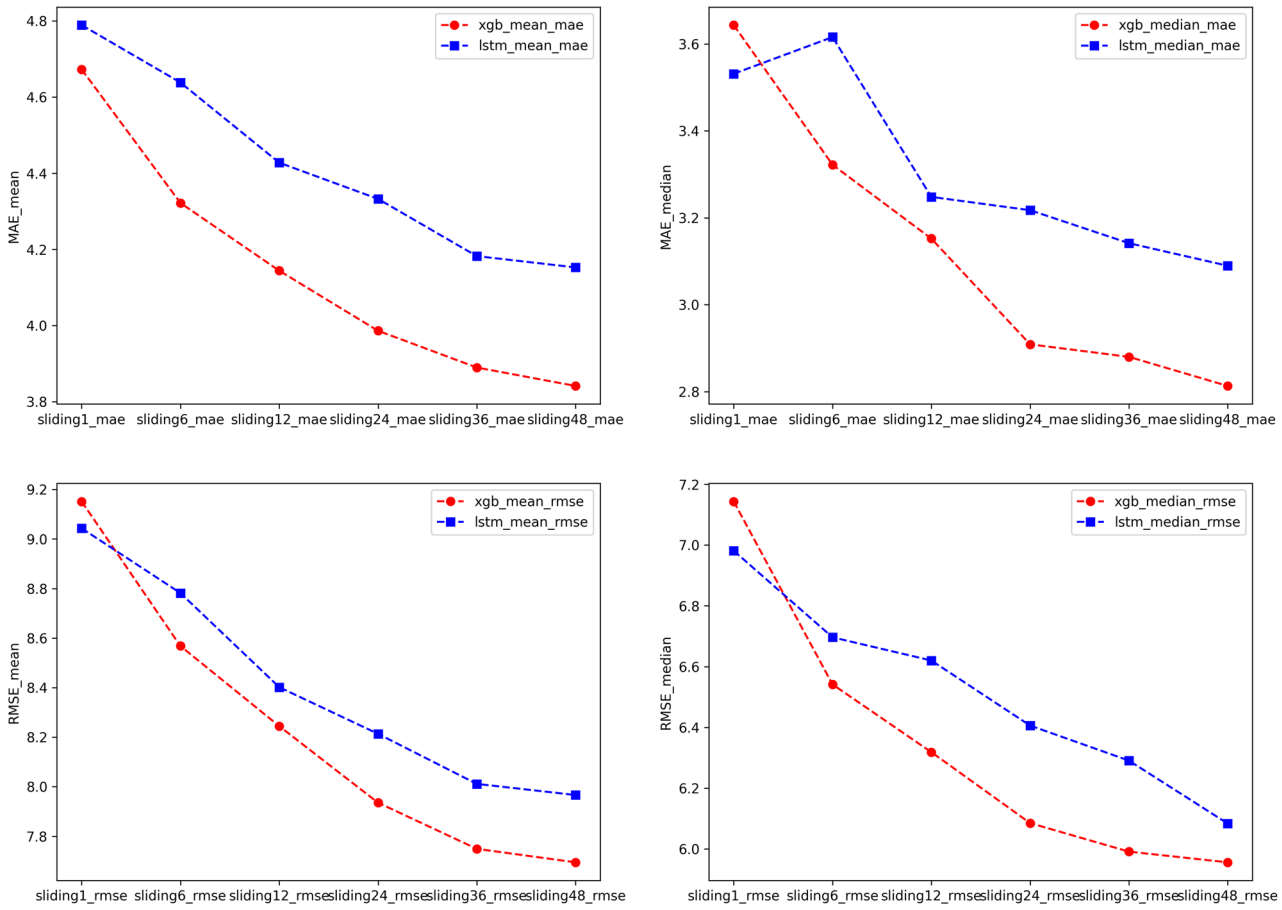

Fig. 14. Sliding windows comparison in the California dataset.

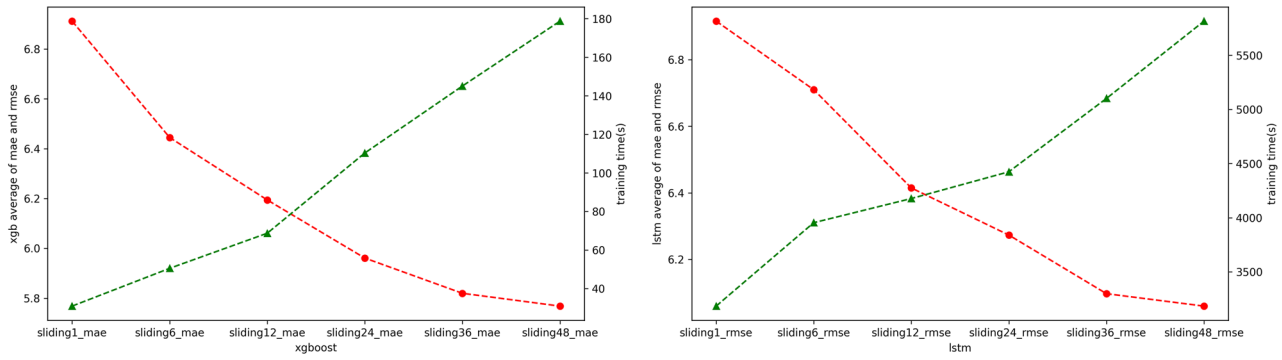

Fig. 15. Relationship among average evaluation metrics, size of time sliding windows and training time.

Figure 17 shows TTP predicted by XGBoost with and without weather features. The x-axis represents the number of routes and the y-axis represents evaluation metrics. We can see that the red line is mostly above the blue line (65 and 60 routes without weather features have lower MAE and RMSE respectively), which means that the prediction made by adding weather features is worse than the prediction without weather features.

Figure 18 shows the result of TTP by LSTM. Compared to the prediction made by XGBoost, adding weather features also deteriorates model performance but not as much as that by XGBoost. Adding weather features deteriorates 38 routes and 45 routes in terms of larger MAE and RMSE respectively. From the result of this experiment, we conclude with the following conjectures:

(a) The weather effect cannot improve the precision of the model significantly in the Taiwan dataset. We conjecture that the reason behind could be due to the climate in Taiwan, in which snowing and extreme climate do not exist in such a subtropical region. We conjecture that, since only extreme weather, which is nearly

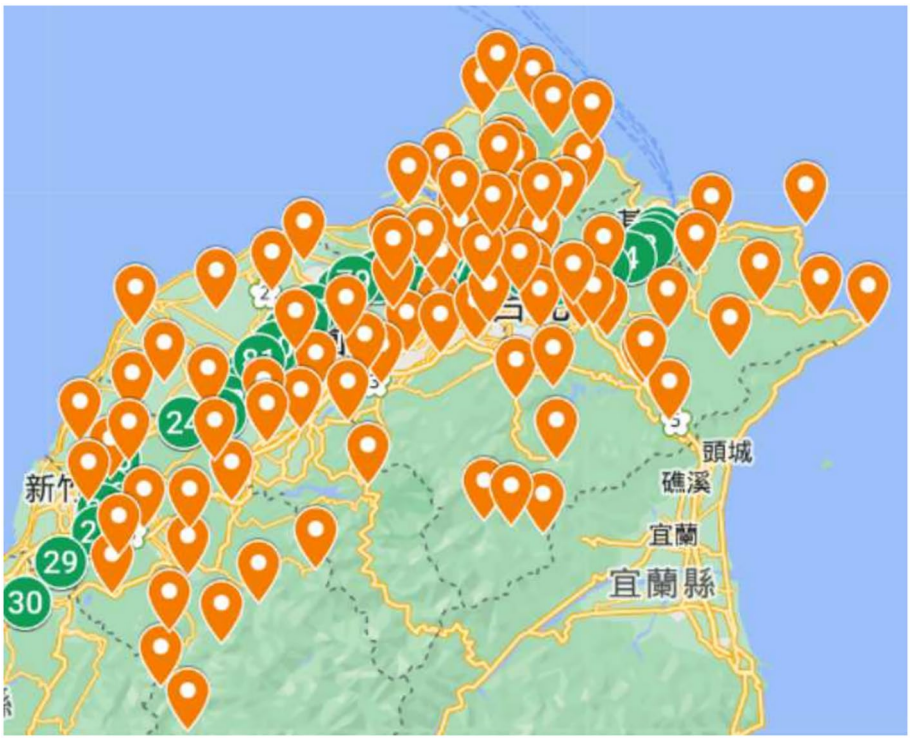

Fig. 16. The weather stations at the north of in Taiwan (Image © Google).

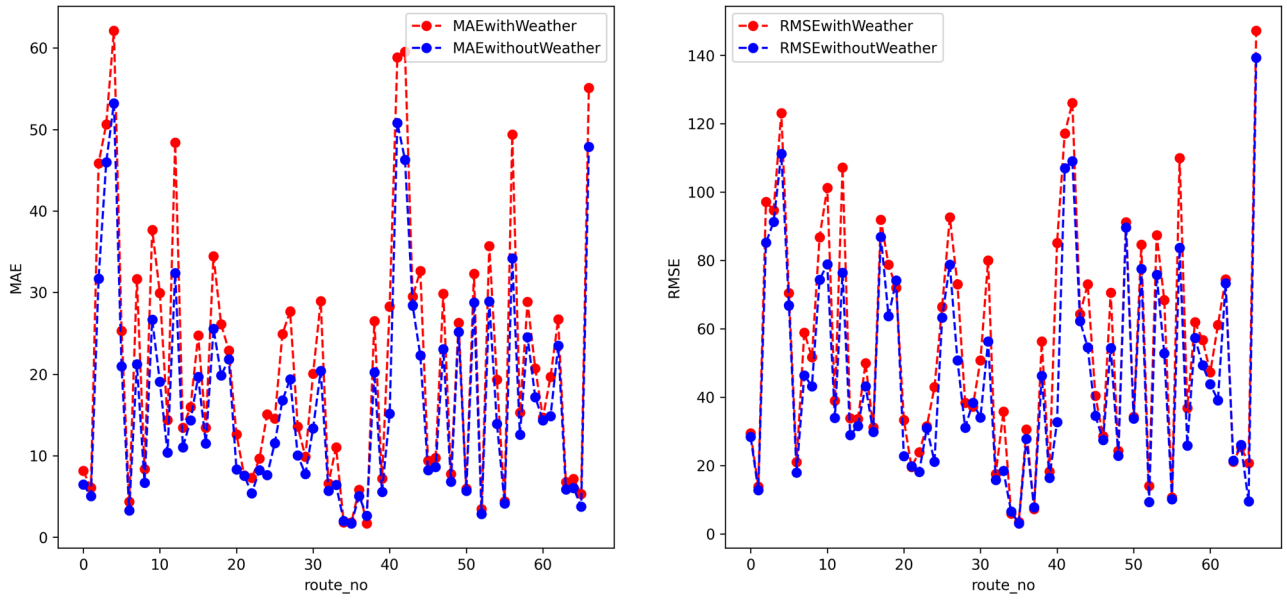

Fig. 17. Weather feature comparison by XGBoost.

impossible in Taiwan, can significantly affect the travel time, the feature of weather effect appears to be noisy and hence it becomes more difficult for the model to fit the dataset well. On the other hand, it might mislead the model's understanding of the dataset and the model may overfit the weather data.

(b) The LSTM, however, is more capable of avoiding the influence of noisy data than XGBoost, because adding weather features in the LSTM model still had more than 20 routes performing as good as that without adding weather features. Therefore, the LSTM has better noise-immunity with adding noisy features in the Taiwan dataset.

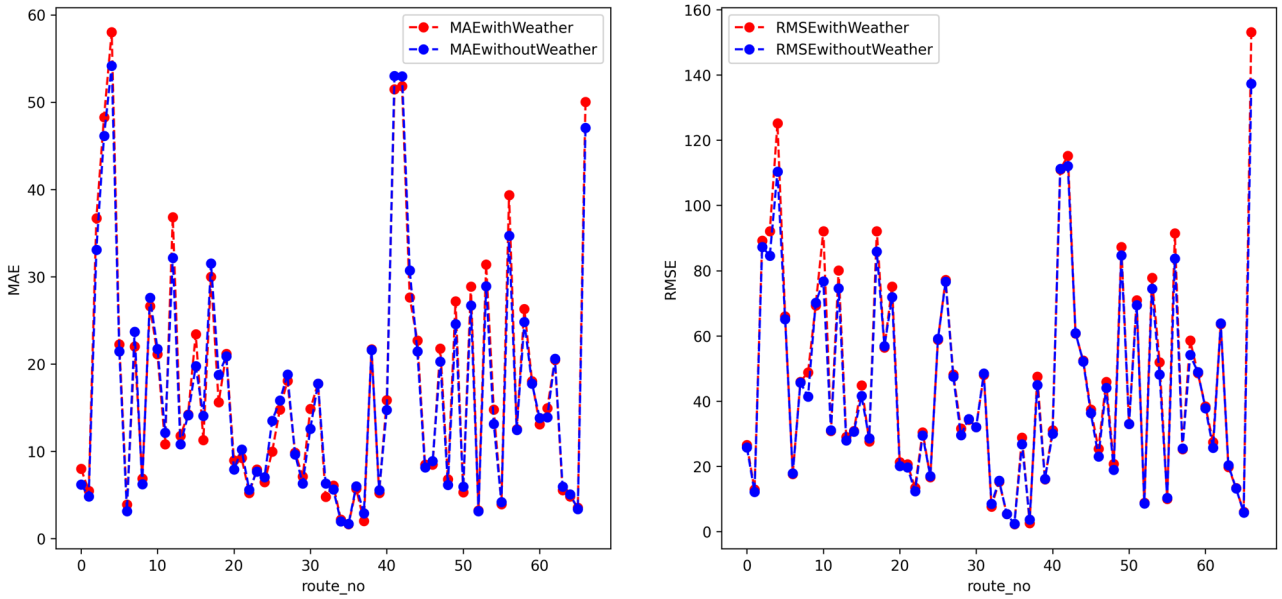

Fig. 18. Weather feature comparison by LSTM.

	Evaluation metric	Without extra features	With national holiday	With minute	With hour	With day	All features
Taiwan	MAE	16.892	16.793	16.011	15.984	16.662	**15.693**
	RMSE	45.997	46.063	45.509	**45.467**	45.820	45.973
California	MAE	4.144	4.1245	**3.519**	3.526	4.012	3.963
	RMSE	8.067	8.062	**7.195**	7.219	7.855	7.802

Table 3. With or without extra features comparison.

Analysis of temporal features
Clearly, periodical patterns are temporal features. In the following, we explain the effects for models made by different temporal features.

(a) Weekday Feature: From Sunday to Saturday. We used cyclic feature engineering to overcome the issue from one-hot encoding, which may cause too many dimensions for models to converge. Cyclic feature engineering uses the `sine` and `cosine` function getting values in [0, 1] to picture a circle on the two dimensional space. It made the distance between Saturday and Sunday equal to the distance between Sunday and Monday.
(b) National Holiday Feature: We label the row by 1 if the row is in a national holiday.
(c) Minute Feature: We used cyclic feature engineering to add minute features, dividing 12 points (one time slot is in the unit of five minutes) on the outline of circle.
(d) Hour Feature: We used cyclic feature engineering to add minute features, dividing 24 points on the outline of circle.

After attaining TTP from XGBoost and LSTM, we average MAE and RMSE from XGBoost and the LSTM to attain comprehensive results from the two models as shown in Table 3. In these two datasets, we derive some results as follows.

(a) Adding hour feature or minute feature will reach similar prediction due to their closed difference between MAE and RMSE.
(b) Adding national holidays does not improve model performance, because the trend of travel time in national holidays is similar to normal holidays.
(c) Adding all the features to our dataset does not bring positive outcomes in most situations. More features came with complexity as burden for the model to minimize the residual error.
(d) After all the experiments we took in the phase of data preprocessing, only hour or minute temporal features can be the candidate features adding to our dataset.

Model	Taiwan		California	
	Non-Peak	Peak	Non-Peak	Peak
RMSE				
XGBoost	40.524	58.027	6.034	9.761
LSTM	36.857	56.59	5.940	9.668
MAE				
XGBoost	12.868	25.058	2.817	5.442
LSTM	12.884	25.78	2.875	5.613

Table 4. Performance analysis of peak and non-peak hours using XGBoost and LSTM models.

Models	Ting's hybrid model	Ho's hybrid model	DE-SLSTM	T-GCN	ATT-GRU	LSTM	XGBoost
Original Features	Travel time, Traffic speed, Volumes	Travel time, Volumes, Hour, Day of a Week	Travel time, Traffic speed, Day of a Week, Holiday, Time slot, Peak, Weather effect	Traffic speed	Traffic speed	-	-
Original Imputation	Denoising AutoEncoder	Max Imputation	Interpolation then using last and next week to impute	Interpolation	-	-	-
Original Sliding Windows Size	6	12	12	12	-	-	-
Edited Features	Travel time, Traffic speed, Volumes, Hour	Travel time, Traffic speed, Volumes, Hour	Travel time, Traffic speed, Traffic volume, Hour, Peak	Traffic speed (Adding feature to T-GCN will change the structure)	Traffic speed (Adding feature to ATT-GRU will change the structure)	Travel time, Traffic speed, Volumes, Hour	Travel time, Traffic speed, Volumes, Hour
Edited Imputations	Max Imputation	Max Imputation	Max Imputation	Max Imputation	Max Imputation	Max Imputation	Max Imputation
Edit Sliding Windows Size	24	24	24	24	12	24	24

Table 5. Model Explanation before and after editing by the view of data preprocessing phase.

Analysis of peak and non-peak hours

To investigate the influence of peak and non-peak hours on travel time prediction, we analyzed the performance of XGBoost and LSTM models using both datasets, Taiwan and California. We evaluated their prediction errors using RMSE and MAE under peak and non-peak scenarios. The results are summarized in Table 4.

Observations and analysis

(a) **Higher Errors During Peak Hours:** Both models, XGBoost and LSTM, exhibited significantly higher RMSE and MAE during peak hours compared to non-peak hours across the datasets. This indicates that traffic patterns during peak hours are more complex and harder to predict accurately due to increased variability caused by congestion and other external factors.

(b) **Regional Variations:** Errors (both RMSE and MAE) in the California dataset were markedly lower than those in the Taiwan dataset. This could be attributed to differences in traffic infrastructure, travel behavior, or dataset characteristics.

(c) **Impact on Feature Engineering:** The results highlight the importance of peak and non-peak temporal segmentation when training models for travel time prediction. Models could benefit from separate handling of peak and non-peak data to optimize their performance for different traffic conditions.

Model comparison phase

In the model comparison phase, we compare the models mentioned at the section of methodology. In these models, we adopted the same datasets (the Taiwan dataset and the California dataset), the same time range (January to June for the training and July for testing) to have a fair view toward these models. In addition, we want to know whether the knowledge we gain from the phase of data-preprocessing can improve these comparison models or not, so we edited features, methods of missing value imputations and size of sliding windows. In the end, we will hopefully be able to know which method is optimal in terms of accurate TTP and whether their methods can be improved by feature engineering. In the phase of data preprocessing, features of hour or minute do improve model performance. Max Imputation has similar influence as interpolation with simpler implementation than other methods. Size of time sliding windows in the range of 24 has the effective result in both training time and decent MAE and RMSE. Table 5 explains our idea in model comparison phase.

Consider the experimental result shown in Table 6, where "OF" stands for Original Features (i.e., method from original studies) and "EF" stands for Edited Features (i.e., method from data preprocessing phase). We first discuss the model performance in terms of original features extracted by the method from original studies. We can see that DE-SLSTM, proposed by Chou et al.[5], attains the best prediction accuracy among other models mostly in both datasets. The reason may be that DE-SLSTM considers most features which have significant importance to improve the models. For instance, in the data-preprocessing phase, we realized that the Hour

Model	Taiwan Dataset (OF)		Taiwan Dataset (EF)		California Dataset (OF)		California Dataset (EF)	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Ting's Hybrid Model	19.743	52.205	16.871	49.024	4.723	9.290	3.687	7.598
Ho's Hybrid Model	23.844	55.150	21.800	54.972	**4.378**	8.627	4.284	8.505
DE-SLSTM	**17.235**	**44.902**	16.648	44.109	4.538	**8.022**	4.654	8.117
T-GCN	17.955	46.882	18.908	47.266	27.626	30.832	26.039	30.424
ATT-GRU	-	-	18.417	**42.207**	-	-	3.833	7.837
LSTM	-	-	16.041	44.128	-	-	3.577	**7.200**
XGBoost	-	-	**15.927**	46.807	-	-	**3.476**	7.239

Table 6. Comparison of all models.

Feature can benefit machine learning models (XGBoost) and deep learning models (LSTM). Moreover, the LSTM surpasses other models on TTP.

The ATT-GRU model performed comparably well, particularly in the California dataset, achieving a competitive RMSE value of 7.837. However, its performance in the Taiwan dataset lagged slightly behind other models such as DE-SLSTM and LSTM. This may be attributed to the relatively simpler temporal dynamics captured by the ATT-GRU's attention mechanism, which works well for datasets with less temporal complexity (e.g., California), but may struggle to generalize for datasets with higher variability, such as Taiwan. Despite its balanced performance, ATT-GRU does not outperform DE-SLSTM, LSTM, or XGBoost in their respective strengths, emphasizing the significance of model selection based on the target dataset characteristics.

Except for T-GCN, edited features generated by our observation in the data-preprocessing phase improve models' prediction. The improvements made by edited features are mainly because of eliminating redundant features, such as day of the week or weather effect (in the Taiwan Dataset). Day of the week, on one hand, provided similar information to the Hour Feature (which are periodical patterns of time), leading to collinearity. On the other hand, the weather effect in Taiwan has been proven worthless in our experiment, for there is no extreme climate in Taiwan. Adding weather effects as extra features makes the model overfit the noisy data, thus removing weather effects improves DE-SLSTM's prediction in MAE and RMSE. The size of sliding windows is also an important factor influencing a model's prediction because the optimal size of windows can gain more view for models toward time patterns. We adopt the original structure of T-GCN and did not add extra features to the model to maintain its original structure. Without adding features to T-GCN, it resulted in only a slight change in its prediction.

The experimental results presented in Table 6 clearly indicate that, when using the original features (OF) as extracted in previous studies, DE-SLSTM consistently achieves the lowest RMSE (44.902 for the Taiwan dataset and 8.022 for the California dataset) and competitive MAE values, suggesting that its comprehensive feature set effectively captures the dynamics of travel time. In contrast, hybrid models such as Ting's GRU-XGBoost and Ho's DNN-XGBoost exhibit higher error metrics (e.g., Ho's Hybrid Model shows an MAE of 23.844 and RMSE of 55.150 in Taiwan) despite their more complex architectures. Notably, the ATT-GRU model achieves a competitive RMSE of 7.837 in the California dataset; however, its performance in the Taiwan dataset lags behind that of DE-SLSTM and the base models. When the models are evaluated with edited features (EF)-where redundant information such as day-of-week and weather effects are removed-the prediction accuracy improves across the board, with the base models (LSTM and XGBoost) yielding the lowest MAE and RMSE values. These findings underscore that the increased complexity inherent in the hybrid models may lead to overfitting and reduced robustness in heterogeneous traffic conditions, while simpler models like XGBoost and LSTM demonstrate more stable and reliable performance.

Finally, the base models, i.e., the LSTM and XGBoost, perform the best in terms of the lowest MAE and RMSE respectively. To sum up, we conclude with the insights as follows:

(a) The model structure does not influence prediction significantly. Using the base model, such as XGBoost and the LSTM, we not only can attain better prediction but also require less time to train the model. The more complicated a model is, the more easily it leads to overfitting.
(b) The improvement from original features to edited features is larger than that from switching to different models. This suggests that we should aim at the part of data preprocessing.
(c) With the characteristic of evaluation metrics, we find that XGBoost and the LSTM are good fits for TTP in different perspectives. XGBoost, with the best prediction in terms of MAE, makes optimal TTP in normal situations, since MAE considers all the values with the same weight. In contrast, the LSTM, with the best RMSE, can predict better in some special situations such as traffic jams or rush hour on the freeway, because RMSE generates more penalty on extreme values and the LSTM performs better on RMSE than XGBoost. Depending on different situations, we can choose the optimal models to meet our demand in TTP on freeways.

A simple and robust baseline for TTP

We designed the baseline primarily to fully leverage the key insights obtained from our comparative studies: different models exhibit distinct advantages under varying traffic conditions. Our approach is deliberately kept simple by relying on only two well-understood base models-XGBoost and LSTM-and a straightforward gating

network, ensuring computational efficiency and ease of implementation. Algorithm 1 summarizes the complete procedure of our proposed baseline approach.

First, XGBoost excels at capturing overall trends in stable traffic conditions, typically achieving a lower MAE, whereas LSTM is adept at capturing long-term dependencies and abrupt changes, thereby reducing RMSE in extreme scenarios such as traffic peaks. Complex hybrid models often require more parameters, which may lead to overfitting and, consequently, a decline in generalization performance. In contrast, by fusing two base models with a gating network that dynamically allocates weights, we can maintain model flexibility while reducing the risk of overfitting. This minimalistic design not only simplifies training and tuning but also enhances robustness, as it avoids the pitfalls of overly complex architectures that can be sensitive to noisy data.

Second, integrating these two simple yet characteristic models through dynamic fusion allows the overall architecture to remain concise and interpretable, making it easier to observe each model's contribution to the final prediction and thus validating the importance of robust data preprocessing and feature engineering. Moreover, a simpler model structure helps mitigate the risk of overfitting and improves generalizability. Finally, our experiments demonstrate that both models perform exceptionally well on their respective metrics, further supporting the decision to employ a dynamic fusion strategy to achieve higher prediction accuracy. In summary, by fusing two simple experts with a dynamic gating mechanism, we construct a baseline that is not only robust across diverse traffic conditions but also significantly easier to deploy and maintain in real-world scenarios. The remainder of the section (Data Preprocessing and Feature Engineering, Expert Models Training, Gating Network for Dynamic Fusion, End-to-End Joint Training, and Experimental Evaluation) further details the simple yet effective procedures that contribute to the overall robustness of our proposed baseline.

Require: Historical traffic data D, sliding window size W, initial parameters for M_{xgb}, M_{lstm}, and the gating network.
Ensure: Predicted travel time $\hat{y}_t$ for each input X_t.

1: **Data Preprocessing:**

1. For each $d_i \in D$, compute the imputed data: $\tilde{d}_i = \mathscr{I}(d_i)$.
2. Extract cyclic temporal features:

$$h_i^{(s)} = \sin\left(\frac{2\pi h_i}{24}\right), \quad h_i^{(c)} = \cos\left(\frac{2\pi h_i}{24}\right),$$

$$m_i^{(s)} = \sin\left(\frac{2\pi m_i}{60}\right), \quad m_i^{(c)} = \cos\left(\frac{2\pi m_i}{60}\right).$$

3. Construct the sliding window: $X_t = \{x_{t-W+1}, x_{t-W+2}, \ldots, x_t\}$, where x_j aggregates the imputed and encoded features.

2: **Expert Models Training:**

1. Compute expert prediction from XGBoost: $p_{\text{xgb}} = M_{\text{xgb}}(X_t)$.
2. Compute expert prediction from LSTM: $p_{\text{lstm}} = M_{\text{lstm}}(X_t)$.

3: **Gating Network and Fusion:**

1. Extract statistical features from X_t: $G_t = \Big[\mu(X_t), \sigma(X_t), \max(X_t), \min(X_t), \ldots\Big]$.
2. Compute the hidden representation: $\mathbf{h} = \phi(W_1 G_t + b_1)$.
3. Compute the output layer: $\mathbf{z} = W_2\mathbf{h} + b_2$.
4. Derive fusion weights via softmax:

$$w_{\text{xgb}} = \frac{\exp(z_1)}{\exp(z_1)+\exp(z_2)}, \quad w_{\text{lstm}} = \frac{\exp(z_2)}{\exp(z_1)+\exp(z_2)}.$$

5. Fuse expert predictions:

$$\hat{y}_t = w_{\text{xgb}} \cdot p_{\text{xgb}} + w_{\text{lstm}} \cdot p_{\text{lstm}}.$$

4: **Joint Training:** Minimize the loss:

$$\mathscr{L} = \frac{1}{N}\sum_{t=1}^{N}\left(y_t - \hat{y}_t\right)^2.$$

Update all parameters using backpropagation.
5: **Output:** For each X_t, output the predicted travel time $\hat{y}_t$.

Algorithm 1. The proposed baseline for travel time prediction.

Data preprocessing and feature engineering

Let the raw traffic data be denoted as

$$D = \{d_1, d_2, \ldots, d_N\},$$

where each data point d_i is associated with a timestamp t_i and measurements (travel time, speed, volume, etc.). We process the data as follows:

1. **Missing value imputation:** For each d_i, if a measurement is missing or erroneously recorded as zero, we apply an imputation function $\mathcal{I}(\cdot)$:

$$\tilde{d}_i = \mathcal{I}(d_i).$$

In our experiments (see Tables 1 and 2), interpolation (Chou's imputation) and max imputation resulted in lower prediction errors, underscoring the importance of robust imputation.

2. **Cyclic Feature Extraction:** For each timestamp t_i, extract the hour h_i and minute m_i, and transform them via cyclic encoding:

$$h_i^{(s)} = \sin\left(\frac{2\pi h_i}{24}\right), \quad h_i^{(c)} = \cos\left(\frac{2\pi h_i}{24}\right),$$
$$m_i^{(s)} = \sin\left(\frac{2\pi m_i}{60}\right), \quad m_i^{(c)} = \cos\left(\frac{2\pi m_i}{60}\right).$$

This formulation effectively captures the periodic nature of traffic data, as validated by our feature analysis in Section "Analysis of temporal features".

3. **Sliding window construction:** To capture temporal dependencies, we construct a sliding window X_t for each prediction time t:

$$X_t = \{x_{t-W+1}, x_{t-W+2}, \ldots, x_t\},$$

where W is the window size. Our experiments (see Fig. 13) indicate that $W = 24$(i.e.,2 hours of 5-minute intervals) is optimal for the Taiwan dataset.

Expert models training

We train two expert models, XGBoost and LSTM, on the preprocessed input X_t. The key insight behind selecting XGBoost and LSTM is that each model excels under different traffic conditions. From our experimental analysis (e.g., see Tables 1 and 6), we observed: 1) XGBoost achieves lower MAE in stable traffic situations because it is very effective at capturing overall trends and non-linear relationships in structured data; 2) LSTM demonstrates superior performance in capturing abrupt changes and long-term dependencies (reflected in lower RMSE), making it particularly adept at handling the volatility seen during traffic peaks. Thus, the corresponding insight is that by combining these two models, we can leverage their complementary strengths-XGBoost for its robustness in stable scenarios and LSTM for its dynamic temporal modeling-in order to create a more robust and accurate TTP baseline that adapts to varying traffic conditions.

- **XGBoost model:**

$$p_{\text{xgb}} = M_{\text{xgb}}(X_t),$$

where M_{xgb} is chosen for its strong performance under stable traffic conditions, as evidenced by lower MAE values in our experiments.

- **LSTM model:**

$$p_{\text{lstm}} = M_{\text{lstm}}(X_t),$$

with M_{lstm} capturing long-term dependencies and abrupt traffic changes, leading to improved RMSE in volatile conditions.

Gating network for dynamic fusion

Given the substantial variability observed in traffic conditions, employing fixed fusion weights is inherently suboptimal. A dynamic fusion mechanism can markedly enhance prediction accuracy by adapting to real-time traffic fluctuations. Here, we propose a gating network that derives its fusion weights directly from statistical features extracted from the input X_t. This mechanism is designed to modulate the contribution of each expert model-XGBoost and LSTM-according to the prevailing traffic regime, thereby reflecting their relative reliability. To fuse the outputs of the expert models dynamically, we introduce a gating network:

1. **Statistical feature extraction:** From X_t, we derive a statistical feature vector:

$$G_t = [\mu(X_t),\ \sigma(X_t),\ \max(X_t),\ \min(X_t),\ \ldots].$$

These statistics capture traffic variability and are critical for informing the gating mechanism.

2. **Gating network computation:** The gating network, implemented as an MLP, computes:

Model	Taiwan Dataset		California Dataset	
	MAE	RMSE	MAE	RMSE
Ting's Hybrid Model	16.87	49.02	3.69	7.60
Ho's Hybrid Model	21.80	54.97	4.28	8.51
DE-SLSTM	16.65	44.11	4.65	8.12
XGBoost	15.93	46.81	3.48	7.24
LSTM	16.04	44.13	3.58	7.20
Proposed Baseline	**15.40**	**43.00**	**3.30**	**6.90**

Table 7. Performance comparison of the proposed baseline with existing models (adapted from Table 6).

$$\mathrm{h} = \phi(W_1 G_t + b_1), \quad \mathrm{z} = W_2 \mathrm{h} + b_2,$$

where $\phi(\cdot)$ is a nonlinear activation function (e.g., ReLU).

3. **Dynamic Weight Allocation:** The output vector $\mathrm{z} = [z_1, z_2]$ is normalized using the softmax function:

$$w_{\mathrm{xgb}} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}, \quad w_{\mathrm{lstm}} = \frac{\exp(z_2)}{\exp(z_1) + \exp(z_2)}.$$

4. **Final Prediction:** The final fused prediction is:

$$\hat{y}_t = w_{\mathrm{xgb}} \cdot p_{\mathrm{xgb}} + w_{\mathrm{lstm}} \cdot p_{\mathrm{lstm}}.$$

End-to-End joint training

The end-to-end joint training ensures that the parameters of both expert models and the gating network are optimized simultaneously. This holistic approach, supported by the improvements observed in our experimental evaluations, confirms that a unified optimization strategy leads to better TTP performance. We train the entire system in an end-to-end fashion by minimizing the loss function:

$$\mathcal{L} = \frac{1}{N} \sum_{t=1}^{N} \left(y_t - \hat{y}_t\right)^2,$$

where y_t represents the ground truth travel time, and N is the number of training samples.

Experimental evaluation of the proposed baseline

In order to demonstrate the effectiveness of our proposed baseline for TTP, we conducted a set of experiments using the same datasets (Taiwan and California) and experimental settings. In our comparative study, Table 6 summarizes the performance of various existing models. Here, we show that our proposed baseline, which dynamically fuses the outputs of XGBoost and LSTM via a gating network, achieves superior performance. The same evaluation metrics, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), are adopted, and report the results in Table 7. As seen in the table, our proposed approach yields lower MAE and RMSE on both datasets, which is indicative of its improved prediction accuracy. For example, in the Taiwan dataset, our proposed baseline achieves an MAE of 15.40 and an RMSE of 43.00, which are better than those obtained by the best existing model (XGBoost: MAE 15.93, RMSE 46.81). Similarly, on the California dataset, the MAE and RMSE are reduced to 3.30 and 6.90, respectively. These improvements validate our insight that dynamic, data-driven fusion (as implemented via the gating network) and end-to-end joint training can significantly enhance TTP performance.

In summary, these experimental results confirm that our new baseline, built upon the insights obtained from our comparative studies, provides a significant improvement in travel time prediction. The dynamic weight allocation via the gating network, in particular, enables the model to adaptively fuse the strengths of XGBoost and LSTM, leading to a more robust and accurate TTP system.

Conclusion

In this paper, we intended to determine what is crucial in long-term travel time prediction. To meet real-world situations, we adopted datasets from Taiwan and California. We began by comparing missing value imputation methods and found that interpolation and max value imputation are the optimal ways to impute zero or missing values, rather than relying on deep learning-based imputation. Besides, we studied the importance of each temporal feature, such as the hour feature and the minute feature, and observed that these features facilitate both machine learning and deep learning models. Weather effects in Taiwan, to our surprise, deteriorated the predictions in the Taiwan dataset due to the characteristics of the region's climate. After gaining insights into the features related to TTP, we compared different hybrid models and realized that models with edited features

perform better than those with original features. Furthermore, our experiments revealed that base models outperform all hybrid models in long-term TTP, with XGBoost performing best under normal conditions and LSTM excelling under extreme situations. Building upon these insights, we proposed a novel dynamic baseline that fuses the complementary strengths of XGBoost and LSTM via a gating network. This approach dynamically allocates weights-guided by key statistical features extracted from a sliding window of historical data-to each expert model, thereby adapting robustly to both stable and volatile traffic conditions and achieving superior prediction accuracy. Finally, we conclude that the importance of data preprocessing and feature engineering has precedence over model construction, paving the way for more effective TTP methods in future research.

Data availability

The datasets analyzed during this study are publicly available. The MOTC dataset can be accessed at https://github.com/smalloshin/tdcs-dataset-tw, and the PeMS dataset is available via https://pems.dot.ca.gov/.

Received: 10 September 2024; Accepted: 13 May 2025
Published online: 15 July 2025

References

1. Ting, P.-Y. et al. Freeway Travel Time Prediction Using Deep Hybrid Model - Taking Sun Yat-Sen Freeway as an Example. *IEEE. Trans. Veh. Technol.* **69**(8), 8257–8266 (2020).
2. Qiao, W., Haghani, A. & Hamedi, M. Short-term travel time prediction considering the effects of weather. *Transportation Research Record* **2308**(1), 61–72 (2012).
3. Qiu, B. & Fan, W. Machine learning based short-term travel time prediction: Numerical results and comparative analyses. *Sustainability* **13**(13), 7454, article (2021).
4. Chen, C. M., Liang, C. C. & Chu, C. P. Long-term travel time prediction using gradient boosting. *J. Intell. Transp. Syst.* **24**(2), 109–124 (2020).
5. Chou, C. H., Huang, Y., Huang, C. Y., spsampsps Tseng, V. S. Long-term traffic time prediction using deep learning with integration of weather effect, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2019)*, **3**, 123–135. (Apr. 2019)
6. Kandiri, A., Ghiasi, R., Nogal, M. & Teixeira, R. Travel time prediction for an intelligent transportation system based on a data-driven feature selection method considering temporal correlation. *Transportation Engineering* **18**, 100272 (2024).
7. Billings, D., Yang, J. S.: Application of the ARIMA models to urban roadway travel time prediction-a case study, in *2006 IEEE International Conference on Systems, Man and Cybernetics (ICSMC 2006)*, **3**, 2529–2534, (Oct. 2006), https://doi.org/10.1109/ICSMC.2006.385244.
8. Yang, J.-S. Travel time prediction using the GPS test vehicle and Kalman filtering techniques. In *Proceedings of the 2005, American Control Conference, 2005*, **3**, 2128–2133 (2005).
9. Yu, B., Wang, H., Shan, W., Yao, B.: Prediction of bus travel time using random forests based on near neighbors, *Computer?Aided Civil and Infrastructure Engineering*, **33**(4), 333–350, (2004), https://doi.org/10.1109/TVT.2020.2999358.
10. Chen, Z. & Fan, W. A Freeway Travel Time Prediction Method Based on an XGBoost Model. *Sustainability* **13**(15), 8577 (2021).
11. Ahmed, I. et al. Travel time prediction and explanation with spatio-temporal features: A comparative study. *Electronics* **11**(1), 106, article (2021).
12. Fu, K., Meng, F., Ye, J., & Wang, Z.: Compacteta: A fast inference system for travel time prediction, *The 26th ACM International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pp. 3337–3345, (2020).
13. Yuan, H., Li, G., Bao, Z., Feng, L. Effective Travel Time Estimation: When Historical Trajectories over Road Networks Matter", *ACM SIGMOD International Conference on Management of Data*, pp. 2135–2149, (2020).
14. Shen, Y., Jin, C. & Hua, J. TTPNet: A neural network for travel time prediction based on tensor decomposition and graph embedding. *IEEE Trans. Knowl. Data. Eng.* **34**, 4514–4526 (2020).
15. Abdollahi, M., Khaleghi, T. & Yang, K. An integrated feature learning approach using deep learning for travel time prediction. *Expert. Syst. Appl.* **139**, 112864 (2020).
16. Ho, M. C., Chen, Y. C., Hung, C. C., & Wu, H. C.: Deep ensemble learning model for long-term travel time prediction on highways. In *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE 2021)*, pp. 129–130, (2021).
17. Zhao, L. et al. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE. trans. Intell. Transp. Syst.* **21**(9), 3848–3858 (2020).
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I.: Attention is all you need, *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 6000–6010, (2017).
19. Ran, X., Shan, Z., Fang, Y. & Lin, C. An LSTM-based method with attention mechanism for travel time prediction. *Sensors* **19**(4), 861 (2019).
20. Chughtai, J.-R., Haq, I. U. & Muneeb, M. An attention-based recurrent learning model for short-term travel time prediction. *PLoS One.* **17**(12), 1–20 (2022).
21. Oreshkin, B., Carpov, D., Chapados, N., & Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations* (2019).
22. Zhou, H., Zhang, S., Xiong, Y., & Cheng, Z. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021).
23. Lim, B., Arik, S. Ö., Pfister, T., & Pfister, T. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *Int. J. Forecast.* arXiv:1912.09363 (2020).
24. Wu, H., Xu, J., Wang, J., & Long, M. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *NeurIPS*. arXiv:2106.13008(2021).
25. Smyl, Slawek. Hybrid methods and ensembles for time series forecasting. *Int. J. Forecast.* **36**(1), 11–20 (2020).
26. Flunkert, V., Salinas, D., & Gasthaus, J. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* arXiv:1704.04110(2017).
27. Han, Huimin, Liu, Zehua, Barrios, Mauricio, Li, Jiuhao & Zeng, Zhixiong. Time Series Forecasting Model for Non-Stationary Series Pattern Extraction Using Deep Learning and GARCH Modeling. *J. Cloud. Comput.* **13**(3), 45–61 (2024).
28. Hanjia Jiang, Leilei Song, Yu Zhang, Yue Yang, and Chao Li, Graph Neural Networks for Traffic Forecasting: A Survey. *arXiv preprint* arXiv:2007.01626, (2021).
29. Benidis, Kostas, Rangapuram, Syama Sundar, & others, Deep learning for time series forecasting: A survey. *arXiv preprint* arXiv:2004.13408, (2020).
30. Miller, et al. A Survey of Deep Learning and Foundation Models for Time Series Forecasting, *arXiv preprint* arXiv:2401.13912, (2024).

31. Liu, Y., et al. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. *arXiv preprint* arXiv:2310.06625 (2023).
32. Liu, Y., Wu, H., Wang, J., & Long, M. Non-Stationary Transformers: Exploring the Stationarity in Time Series Forecasting. *arXiv preprint* arXiv:2205.14415 (2022).
33. Yi, K. et al. FourierGNN: Rethinking Multivariate Time Series Forecasting from a Pure Graph Perspective. *arXiv preprint* arXiv:2311.06190 (2023).
34. Jin, M., et al. A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. (2023).
35. Wang, D., Zhang, J., Cao, W., Li, J., & Zheng, Y. When will you arrive? Estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* **32** (1) (2018).
36. Derrow-Pinion, A., et al. Eta prediction with graph neural networks in Google Maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* pp. 3767–3776 (2021).
37. Cai, W., Wang, K., Wu, H., Chen, X., & Wu, Y. ForecastGrapher: Redefining Multivariate Time Series Forecasting with Graph Neural Networks. *arXiv preprint* arXiv:2405.18036 (2024).
38. Abiri, N., Linse, B., Edén, P. & Ohlsson, M. Establishing strong imputation performance of a denoising autoencoder in a wide range of missing data problems. *Neurocomputing* **365**(6), 137–146 (2019).
39. Flum, J. & Grohe, M. *Parameterized Complexity Theory* (Springer, 2006).
40. Àlvarez, C., Gabarro, J. & Serna, M. Equilibria problems on games: Complexity versus succinctness. *J. Comput. Syst. Sci.* **77**(6), 1172–1197 (2011).
41. Breiman, L. Random forests. *Machine Learning* **45**(1), 5–32 (2001).
42. Breiman, L. Bagging predictors. *Machine Learning* **24**(2), 123–140 (1996).
43. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016)*, 785–794, (2016), https://doi.org/10.1145/2939672.2939785.
44. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, *arXiv preprint* arXiv:1412.3555, (2014).
45. Das, S., Kalava, R. N., Kumar, K. K., Kandregula, A., Suhaas, K., Bhattacharya, S., Ganguly, N.: Map enhanced route travel time prediction using deep neural networks, *arXiv preprint* arXiv:1911.02623, (2019).
46. Duan, Y., Yisheng, L., & Wang, F. Y.: Travel time prediction with LSTM neural network, in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016)*, pp. 1053–1058, (2016).
47. Friedman, J. H. Stochastic gradient boosting. *Comput. Stat. Data. Anal.* **38**(4), 367–378 (2002).
48. Freund, Y., & Schapire, R. E. Experiments with a new boosting algorithm, in *Proceedings of the 13th International Conference on International Conference on Machine Learning (ICML 1996)*, pp. 148–156, (1996).

Author contributions

Chuang-Chieh Lin: methodology, review, editing finalizing the manuscript. Min-Chu Ho: methodology, software, validation, data curation, writing original draft. Chih-Chieh Hung: methodology, validation, formal analysis, supervision, data curation, review, editing. Hui-Huang Hsu: review, editing. All authors read and approved the final manuscript.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to C.-C.H.

Reprints and permissions information is available at www.nature.com/reprints.

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