Towards An Efficient and Effective En Route Travel Time Estimation Framework

Zekai Shen[1,2], Haitao Yuan[3(✉)], Xiaowei Mao[1,2], Congkang Lv[1,2], Shengnan Guo[1,2(✉)], Youfang Lin[1,2], Huaiyu Wan[1,2]

[1] School of Computer Science and Technology, Beijing Jiaotong University, China
[2] Beijing Key Laboratory of Traffic Data Analysis and Mining, China
{zkshen, maoxiaowei, congkanglv, guoshn, yflin, hywan}@bjtu.edu.cn
[3] Nanyang Technological University, Singapore
haitao.yuan@ntu.edu.sg

Abstract. En route travel time estimation (ER-TTE) focuses on predicting the travel time of the remaining route. Existing ER-TTE methods always make re-estimation which significantly hinders real-time performance, especially when faced with the computational demands of simultaneous user requests. This results in delays and reduced responsiveness in ER-TTE services. We propose a general efficient framework U-ERTTE combining an Uncertainty-Guided Decision mechanism (UGD) and Fine-Tuning with Meta-Learning (FTML) to address these challenges. UGD quantifies the uncertainty and provides confidence intervals for the entire route. It selectively re-estimates only when the actual travel time deviates from the predicted confidence intervals, thereby optimizing the efficiency of ER-TTE. To ensure the accuracy of confidence intervals and accurate predictions that need to re-estimate, FTML is employed to train the model, enabling it to learn general driving patterns and specific features to adapt to specific tasks. Extensive experiments on two large-scale real datasets demonstrate that the U-ERTTE framework significantly enhances inference speed and throughput while maintaining high effectiveness. Our code is available at https://github.com/shenzekai/U-ERTTE

Keywords: En Route Travel Time Estimation · Uncertainty Quantification

1 Introduction

Travel Time Estimation (TTE) serves as a cornerstone of Intelligent Transportation Systems (ITS) [1], enabling the prediction of travel time for specific routes with departure time. This is critical for applications like route planning [2], navigation [3], traffic forecasting [4, 5], and online ride-hailing services [6]. Most methods, known as PRe-route TTE (PR-TTE), focus on predicting the total travel time before departure. However, many real-world scenarios require en route TTE (ER-TTE), which provides real-time estimates while driving. Although some studies [7, 8] have developed ER-TTE models that integrate traveled route data and dynamic features, they often overlook the need for real-time efficiency,

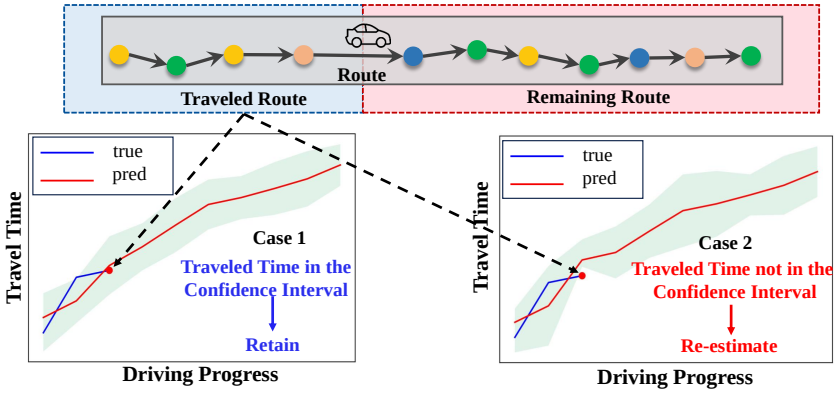


Fig. 1: Uncertainty quantification in ER-TTE using confidence intervals to determine when re-estimation is necessary.

limiting practical application. Thus, developing an efficient and effective ER-TTE framework faces several challenges.

C1: How to reduce the number of invoking estimation models without loss of accuracy? To provide timely navigation updates, ER-TTE must refresh remaining travel time estimates during the route. However, frequent model invocations cause delays due to inference time, and high concurrent user requests further strain system capacity. For instance, Didi reported an average of 33.01 million daily orders in China in Q2 2024[1], illustrating the demand for simultaneous ER-TTE updates. Limited server processing power may lead to delays for some users. Therefore, minimizing unnecessary model calls while maintaining accuracy is essential to improving efficiency and throughput.

C2: How to ensure the robustness of the TTE model throughout the route? It's essential to balance general driving patterns with route-specific conditions. The model should learn common travel patterns across routes while quickly adapting to individual driving styles, such as unique acceleration or braking habits. Additionally, it must promptly adjust to unexpected conditions like traffic congestion or accidents to deliver accurate, real-time predictions. However, existing approaches [7, 8] mainly emphasize capturing user driving preferences. Although useful, these models often lack the flexibility to adapt to dynamically changing traffic situations based on real-time information. For example, during traffic congestions, the models may underestimate delays, which lowers overall prediction accuracy.

In this paper, we introduce an efficient and effective framework U-ERTTE to tackle the key challenges in ER-TTE, featuring an innovative Uncertainty-Guided Decision (UGD) mechanism and Fine-Tuning with Meta-Learning (FTML). The UGD mechanism optimizes requests handling by strategically reusing previous estimates, thus minimizing redundant computations. As shown in Figure 1, the

[1] https://ir.didiglobal.com/news-and-events/

system first quantifies uncertainty across the entire route and stores confidence intervals. During travel, it monitors actual travel time. If travel time remains within the confidence intervals, the estimate is retained, avoiding unnecessary re-estimations. The ER-TTE is then derived by subtracting elapsed time from the initial estimate. Only when significant deviations occur, indicating changed traffic conditions, does the system trigger re-estimation. This UGD mechanism reduces model invocation frequency, addressing the challenge of high request volumes (i.e., **addressing C1**). Additionally, FTML is introduced to enhance prediction accuracy, using MAML [9] to pre-train on general driving patterns and then fine-tune for specific tasks, allowing rapid adaptation to route dynamics while preserving generalizability (i.e., **addressing C2**). Therefore, the framework U-ERTTE significantly boost both the effectiveness and efficiency of ER-TTE.

In summary, the contributions can be summarized as follows:

1. To our best knowledge, this study is the first to consider the efficiency of ER-TTE and achieve an efficient and effective ER-TTE framework.
2. We propose an Uncertainty-Guided Decision mechanism (UGD) that determines adaptively whether ER-TTE needs to be re-estimated, offering a novel approach that can be integrated with existing TTE backbone models.
3. We develop a new training strategy called Fine-Tuning with Meta-Learning (FTML) that enhances both generalizability and adaptability.
4. Experiments on two large-scale real-world datasets verify the efficiency and effectiveness of the framework. Our approach significantly improves inference speed and throughput while maintaining high performance.

2 RELATED WORK

2.1 Travel Time Estimation

Travel time estimation (TTE) can be divided into OD-based TTE, which relies on origin, destination, and departure time [10, 11, 12, 13, 14], and route-based TTE, which incorporates detailed route information. Early methods based on regression or decomposition offered limited accuracy [15], whereas deep learning techniques have recently achieved significant improvements. RNN-based models capture sequential dependencies in trajectories [6, 16], while graph neural networks and attention mechanisms more effectively integrate spatio-temporal features [17, 18]. Additionally, Transformer-based approaches and variational encoders have been employed to model complex dependencies and uncertainties [19, 20, 21]. In the domain of early remaining travel time estimation (ER-TTE), meta-learning strategies have enhanced adaptability [7, 8], although they tend to focus on specific locations rather than continuous route progression.

2.2 Uncertainty Quantification

Uncertainty quantification methods, widely applied to real-world problems [22], fall into two categories: Bayesian and non-Bayesian approaches. Bayesian methods

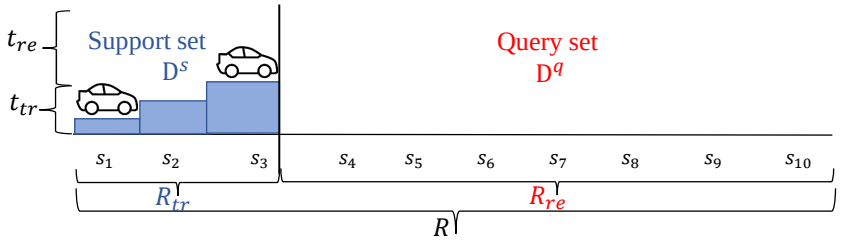


Fig. 2: An explanation of Meta-Learning for ER-TTE

face the difficulty of directly calculating the posterior distribution, necessitating the use of approximate methods such as variational inference and Markov chain Monte Carlo (MCMC). Bayesian neural networks (BNNs) [23] utilize approximate Bayesian inference to enhance the efficiency of their inferential processes. Monte Carlo dropout (MC Dropout) [24] assumes that the parameters of each layer of the neural network follow a Bernoulli distribution, thereby controlling the drop probability of hidden layer neurons. Non-Bayesian methods, typically frequentist in nature, offer greater flexibility through techniques like quantile regression [25] and MIS regression [26], which embed uncertainty directly into loss functions.

3 PRELIMINARY

3.1 En Route Travel Time Estimation

Given a driving route $R = [s_1, s_2, \ldots, s_n]$ composed of n road segments, and considering current segment s_m $(1 \leq m < n)$ at time t, the route can be segmented into two parts: traveled route $R_{tr} = [s_1, s_2, \ldots, s_m]$ and remaining route $R_{re} = [s_{m+1}, s_{m+2}, \ldots, s_n]$. Specifically, the travel time for the traveled portion of route R_{tr} is known and denoted as y_{tr}. The objective of the en route travel time estimation (ER-TTE) task is to predict the travel time for the remaining portion of route R_{re}. This task can be formally described by function:

$$\hat{y}_{re} = f_\theta(t, y_{tr}, R_{tr}, R_{re}),$$

where f_θ denotes the ER-TTE model, θ represents the model parameters, t is the current time, and $\hat{y}_{re}$ is the estimated travel time for R_{re}.

3.2 Meta-Learning in ER-TTE

Meta-learning, commonly called "learning to learn", is designed to train models capable of rapidly adapting to new tasks with limited new data. This approach enables a meta-learner to guide the model in effectively learning from diverse tasks. In the context of the ER-TTE problem, meta-learning is employed to discern general patterns of the entire routes, allowing the model to tailor its

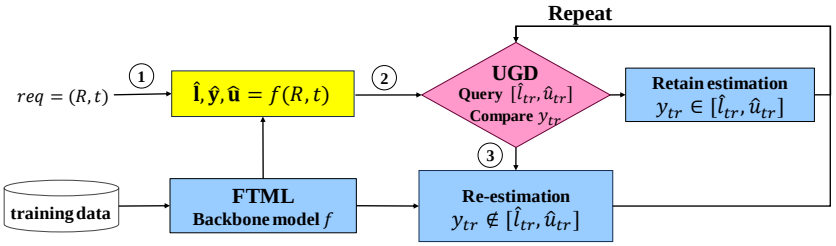


Fig. 3: Pipeline of our U-ERTTE framework.

predictions to the specific nuances of ER-TTE scenarios. Such as the traveled routes contain driver preferences such as acceleration.

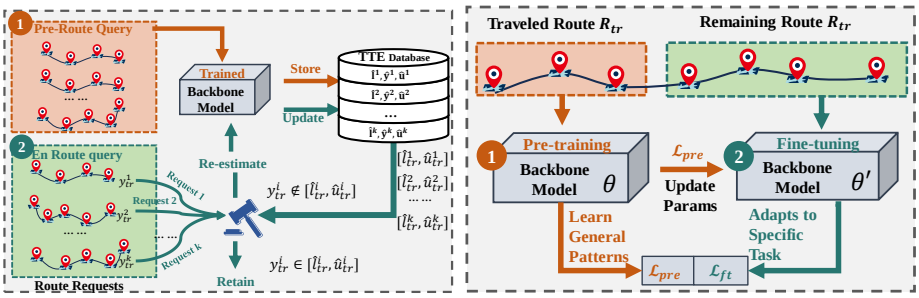
In the meta-learning framework applied to ER-TTE, the dataset is strategically split into a support set and a query set. Following the setting of prior research [7, 8], the traveled portion of the route R_{tr} is utilized as the support set D^s, while the untraveled segment R_{re} is treated as the query set D^q. This division aids the model in leveraging past experiences (support set) to make informed predictions about future segments (query set) of the route. A graphical representation of the support and query sets for a specific route is illustrated in Figure 2. This methodological approach underscores the adaptability and foresight that meta-learning imparts to the ER-TTE model.

4 METHODOLOGY

4.1 Overview

To enhance the efficiency and accuracy of the ER-TTE process, we propose a simple yet effective framework U-ERTTE as depicted in Figure 3. At first, for a given request $req = (R, t)$, the model f first estimates the travel time $\hat{\mathbf{y}}$ and the associated confidence intervals for the entire route before departure, utilizing the Fine-Tuning with Meta-Learning (FTML) strategy. The initial confidence intervals, denoted as $\hat{\mathbf{l}}$ and $\hat{\mathbf{u}}$, encapsulates the predicted travel time from the origin to the destination. Subsequently, during the route, the Uncertainty-Guided Decision (UGD) mechanism is employed to continuously assess the alignment between the actual travel time and the estimated confidence intervals. If the actual travel time y_{tr} falls within the current confidence interval $[\hat{l}_{tr}, \hat{u}_{tr}]$, the existing estimation is retained. Conversely, if y_{tr} deviates from $[\hat{l}_{tr}, \hat{u}_{tr}]$, a re-estimation is triggered using model f, and this validation cycle is repeated throughout the route to ensure reliable predictions.

Remark. We divide the entire route for k parts. The terms $\hat{\mathbf{l}} = \{l_1, l_2, ..., l_k\}$ and $\hat{\mathbf{u}} = \{u_1, u_2, ..., u_k\}$ represent the lower and upper bounds of the confidence intervals of the entire route, representing the complete range of expected travel times from origin to destination. In contrast, $[\hat{l}, \hat{u}]$ specifies the confidence interval at a particular point along the route.



(a) Fine-Tuning with Meta-Learning. (b) Uncertainty-Guided Decision.

Fig. 4: Components of the U-ERTTE Framework

4.2 Uncertainty-Guided Decision Mechanism

Motivation. To improve the efficiency of ER-TTE and minimize system resource consumption under the large request frequency, it is intuitive to design a decision mechanism to determine whether a re-estimation is needed at the request time t during the driving. Travel time is influenced by unpredictable factors, such as traffic congestion and road accidents, which introduce uncertainty into the predictions. Therefore, quantifying uncertainty allows us to capture the variability in TTE. Accordingly, we propose an Uncertainty-Guided Decision (UGD) Mechanism, quantifying the uncertainty in the route to make decisions.

Overview. As illustrated in Figure 4b, the Uncertainty-Guided Decision (UGD) process is categorized into two distinct phases: *pre-route query* and *en route query*. Initially, in the pre-route query phase, the trained backbone model is utilized to compute the confidence intervals for each route prior to departure, which are then stored in the TTE database. Subsequently, in the en route query phase, these confidence intervals are employed to verify the accuracy of ongoing predictions. If the actual travel time deviates from the established confidence intervals, indicating the necessity for adjustments, the model is invoked to re-estimate the travel time for the remainder of the route, updating the corresponding confidence intervals in the TTE database accordingly. If the current predictions fall within the confidence intervals, the existing estimations are retained to respond to the en route queries. Notably, the key point is to calculate, store, and update confidence intervals due to that the confidence intervals provide a solid foundation for decision-making.

Decision-Making Algorithm. The entire procedure employing the Uncertainty-Guided Decision (UGD) mechanism is outlined in Algorithm 1. Initially, we utilize the model to generate preliminary predictions for the pre-route query, represented as $\hat{\mathbf{y}}$, along with the associated confidence intervals for the entire route, denoted as $\hat{\mathbf{l}}$ (lower bounds) and $\hat{\mathbf{u}}$ (upper bounds). These predictions and confidence intervals are subsequently stored in a database, laying the groundwork for the en route query stage (lines 1-2). There are k parts in the route which system queries at specific locations. For each part i,the system compares the actual travel time

Algorithm 1 Uncertainty-Guided Decision.

Require: Entire route R; traveled routes $\{R^1_{tr}, R^2_{tr}, ..., R^k_{tr}\}$; Remained routes $\{R^1_{re}, R^2_{re}, ..., R^k_{re}\}$; trained model f_θ;

1: $\hat{\mathbf{l}}, \hat{\mathbf{y}}, \hat{\mathbf{u}} = f_\theta(R, t)$; // `Pre-route query`
2: UGD.store(confidence intervals);
3: **for** $i \longleftarrow 1...k$ **do**: // `En route query during the route`
4: UGD.query(y^i_{tr}, R^i_{tr}) ;
5: **if** $y^i_{tr} \in [\hat{l}^i_{tr}, \hat{u}^i_{tr}]$:
6: $[\hat{l}^i_{re}, \hat{y}^i_{re}, \hat{u}^i_{re}] = [l - \hat{l}^i_{tr}, \hat{y} - \hat{y}^i_{tr}, u - \hat{u}^i_{tr}]$;
7: **else**: // `re-estimation`
8: $[\hat{l}^i_{re}, \hat{y}^i_{re}, \hat{u}^i_{re}] = f_\theta(t, y^i_{tr}, R^i_{tr}, R^i_{re})$;
9: UGD.update(confidence intervals)
10: **end for**

y^i_{tr} with confidence interval $[\hat{l}^i_{tr}, \hat{u}^i_{tr}]$. If the actual travel time resides within the confidence interval, i.e., $y^i_{tr} \in [\hat{l}^i_{tr}, \hat{u}^i_{tr}]$, the prediction is considered accurate, and the original estimate is retained. The ER-TTE is calculated by subtracting the prediction for the traveled route from the initial estimate (lines 5-6). Conversely, if the actual travel time falls outside the confidence interval i.e., $y^i_{tr} \notin [\hat{l}^i_{tr}, \hat{u}^i_{tr}]$, this discrepancy suggests significant uncertainty and possible inaccuracies in the prediction, necessitating a re-estimation. In such cases, the system recalculates the travel time using updated route information. (lines 7-9).

Formally, the application of UGD in the en route query to get the remain estimation $[\hat{l}^i_{re}, \hat{y}^i_{re}, \hat{u}^i_{re}]$ can be represented by the following equation:

$$[\hat{l}^i_{re}, \hat{y}^i_{re}, \hat{u}^i_{re}] = \begin{cases} [l - \hat{l}^i_{tr}, \hat{y} - \hat{y}^i_{tr}, u - \hat{u}^i_{tr}] & \text{if } y_{tr} \in [\hat{l}_{tr}, \hat{u}_{tr}], \\ f_\theta\left(t, R_{tr}, R_{re}\right) & \text{if } y_{tr} \notin [\hat{l}_{tr}, \hat{u}_{tr}]. \end{cases} \tag{1}$$

Efficiency Analysis. The backbone model complexity is $O(m)$, where m depends on the components of model and input dimensions. When the UGD decides to retain estimation, the need for re-invoking the backbone model is bypassed, thereby reducing the complexity to $O(1)$. Specifically, assuming a confidence level of p (e.g., $p = 0.8$), indicating that there is an 80% probability that no re-estimation will be needed for a given request during the route, the process predominantly operates at $O(1)$ complexity. As the confidence level increases, the proportion of requests that do not require re-estimation grows, leading to a further reduction in overall time complexity. Consequently, The computational overhead is significantly reduced.

4.3 Fine-Tuning with Meta-learning

Motivation. To ensure the robustness of TTE models throughout the entire route, on the one hand, the model must effectively learn and generalize driving patterns across various routes to provide a solid foundation for understanding stable driving preferences. On the other hand, the model must be flexible enough

Algorithm 2 Fine-Tuning with Meta-Learning.

Require: Entire route R; traveled route R_{tr}; model parameters θ; total epoch N; total iteration n_{iter}.

for i $\leftarrow$ 1 to N:
  while $n < n_{\text{iter}}$:
    $\hat{l}, \hat{y}, \hat{u} = f_\theta(R)$; $\hat{l}_{tr}, \hat{y}_{tr}, \hat{u}_{tr} = f_\theta(R_{tr})$.
    compute the loss function according to Equation (8);
    Update $\theta' \leftarrow \theta - lr \cdot \nabla_\theta \mathcal{L}_{pre}(\theta)$; // `pre-train`
    $\hat{l}_{re}, \hat{y}_{re}, \hat{u}_{re} = f_{\theta'}(R_{re}, R_{tr})$;
    compute the loss function according to Equation (11);
    Update $\theta \leftarrow \theta - lr \cdot \nabla_\theta(\mathcal{L}_{re}(\theta') + \mathcal{L}_{pre}(\theta))$; // `fine-tune`
  end while
Return θ

to adapt to dynamically changing traffic scenarios, allowing it to adjust predictions based on real-time data. With these goals in mind, we implement Fine-Tuning with Meta-Learning (FTML) to enhance both generalizability and adaptability.

Overview. As shown in Figure 4a, FTML comprises two stages: *pre-training* and *fine-tuning*. In the *pre-training* stage, learns general driving patterns by predicting both the total route arrival time $\hat{y}$ and the traveled time $\hat{y}_{tr}$ at specific locations, along with their confidence intervals. The model parameters are updated to enable the model to learn general driving patterns. In the *fine-tuning* stage, the model further adjusts by predicting the travel time $\hat{y}_{re}$ and the confidence interval for the remaining route, enabling it to adapt to real-time conditions.

Learning Algorithm. The FTML procedure is depicted in Algorithm 2. In pre-training stage, the model is initialized with effective weights to learn general driving patterns for the entire route and specific features of the traveled route. The focus is on training the model to estimate the travel time $\hat{y}$ and its confidence interval $[\hat{l}, \hat{u}]$ for the entire route R as well as the traveled time $\hat{y}_{tr}$ and its confidence interval $[\hat{l}_{tr}, \hat{u}_{tr}]$ for traveled route R_{tr} within the support set D^s.

$$\hat{l}, \hat{y}, \hat{u} = f_\theta(R), \tag{2}$$

$$\hat{l}_{tr}, \hat{y}_{tr}, \hat{u}_{tr} = f_\theta(R_{tr}). \tag{3}$$

To accurately supervise the predictions and corresponding confidence intervals, we employ *quantile regression* [25], which is a robust and effective method for quantifying uncertainty. Quantile regression is distribution-free and directly optimizes the quantile loss function for providing precise supervision on specific quantile values without requiring any external parameters. In particular, its basic form is illustrated as follows:

$$\begin{aligned} \mathcal{L}^{qua} = & \mathbb{I}_{\hat{l} \geq y} \alpha^{\hat{l}} \left| y - \hat{l} \right| + \mathbb{I}_{\hat{l} < y} \left(1 - \alpha^{\hat{l}}\right) \left| y - \hat{l} \right| + \\ & \mathbb{I}_{\hat{y} \geq y} \alpha^{\hat{y}} \left| y - \hat{y} \right| + \mathbb{I}_{\hat{y} < y} \left(1 - \alpha^{\hat{y}}\right) \left| y - \hat{y} \right| + \\ & \mathbb{I}_{\hat{u} \geq y} \alpha^{\hat{u}} \left| y - \hat{l} \right| + \mathbb{I}_{\hat{u} < y} \left(1 - \alpha^{\hat{u}}\right) \left| y - \hat{l} \right|, \end{aligned} \tag{4}$$

where α is the quantile ($0 < \alpha < 1$), $\mathbb{I}$ is the indicator function. This loss function is to evaluate the relationship between $\hat{l}, \hat{y}, \hat{u}$ and the label y, applying different weights based on the target quantile α. Specifically, the $\alpha^{\hat{l}} < 0.5$, the loss function imposes a greater penalty for cases where the prediction is lower than the label (i.e., lower bound). Conversely, when $\alpha^{\hat{u}} > 0.5$, the loss function imposes a greater penalty for cases where the prediction is higher than the label (i.e., upper bound). $\alpha^{\hat{y}} = 0.5$, the loss function is for the predication $\hat{y}$.

Building on this, we define a composite loss L_{pre} for the pre-training stage that integrates the quantile loss $\mathcal{L}_{en}$ for the entire route R and the quantile loss $\mathcal{L}_{tr}$ in support set D^s. To further refine the confidence interval, we use Mean Prediction Interval Width (MPIW) [27] to constrain the confidence interval to ensure it remains tight and reliable throughout the route.

$$\mathcal{L}_{en}(\theta) = \mathcal{L}^{qua}([\hat{l}, \hat{y}, \hat{u}], y), \tag{5}$$

$$\mathrm{MPIW} = \hat{u}_{tr} - \hat{l}_{tr}, \tag{6}$$

$$\mathcal{L}_{tr}(\theta) = \mathcal{L}^{qua}([\hat{l}_{tr}, \hat{y}_{tr}, \hat{u}_{tr}], y_{tr}) + \mathrm{MPIW}. \tag{7}$$

The total loss $\mathcal{L}_{pre}$ of the pre-training stage is given by the sum of the two losses (lines 3-4):

$$\mathcal{L}_{pre}(\theta) = \mathcal{L}_{en}(\theta) + \mathcal{L}_{tr}(\theta). \tag{8}$$

Then update the model parameters θ through the $\mathcal{L}_{pre}$ (line 5):

$$\theta' \leftarrow \theta - lr \cdot \nabla_{\theta} \mathcal{L}_{pre}(\theta), \tag{9}$$

where lr is the learning rate and θ' denotes the updated model parameters. By the end of this stage, the model achieves a good weight initialization, enabling it to rapidly adapt to new tasks (i.e.,ER-TTE) in the fine-tuning stage and make accurate predictions.

During the fine-tuning stage, the inputs for this stage include features of both the already traveled and remaining routes which ensure that the model can adapt to the dynamic traffic conditions of partially traveled routes and then provide accurate ER-TTE. The pre-trained model with parameters θ' is fine-tuned specifically for prediction $\hat{y}_{re}$ and its confidence interval $[\hat{l}_{re}, \hat{u}_{re}]$ for the remaining route in the query set D^q. The fine-tuning quantile loss $\mathcal{L}_{ft}$ for the remaining route is calculated as follows (lines 6-7):

$$\hat{l}_{re}, \hat{y}_{re}, \hat{u}_{re} = f_{\theta'}(R_{re}, R_{tr}), \tag{10}$$

$$\mathcal{L}_{ft}(\theta') = \mathcal{L}^{qua}([\hat{l}_{re}, \hat{y}_{re}, \hat{u}_{re}], y_{re}). \tag{11}$$

The model parameters are finally updated based on the two losses (line 8).

$$\theta \leftarrow \theta - lr \cdot \nabla_{\theta}(\mathcal{L}_{ft}(\theta') + \mathcal{L}_{pre}(\theta)). \tag{12}$$

In summary, FTML makes the model generalize from previous tasks (entire and traveled TTE) and swiftly adapt to new tasks (ER-TTE) through a two-stage

training process: *pre-training* and *fine-tuning*. During *pre-training*, the model captures general driving patterns for entire routes and the specific features from the traveled route, enabling it to understand the overall dynamics of travel time across different routes. The *fine-tuning* stage allows the model to specialize in the task of ER-TTE, ensuring that it can adapt to the specific conditions of partially traveled routes and provide accurate predictions.

5 Experiment

5.1 Experimental Settings

Datasets and Preprocessing. We utilize two real-world taxi trajectory datasets collected from the cities of Porto[2], and Xian[3]. For both datasets, We removed outlier data (i.e. driving distances that were too short and too long). The processed Porto dataset contains 1,011,761 routes and Xian contains 1,191,125 routes

Metrics. Similar to existing methods [20], we use four metrics for performance evaluation, including mean absolute percentage error (MAPE), mean absolute error (MAE), root mean squared error (RMSE), and satisfaction rate (SR). Specifically, SR refers to the proportion of routes with a MAPE less than 10%, and a higher SR indicates better performance and customer satisfaction. They are defined as follows: $\mathrm{SR} = \frac{1}{N}\sum_{i}^{N}(|\frac{\hat{y_i}-y}{y}| \leq 10\%) \times 100\%$

Implementation Details. The Adam optimizer is used with a fixed learning rate of 10^{-3} and a weight decay of 10^{-3} as a regularization term to prevent overfitting. We use the training set to train the model, select the model with the best MAPE on the validation set, and use the test set to evaluate the performance. All experiments are implemented in Python using the Pytorch toolbox, using an NVIDIA RTX A4000 GPU. The platform runs on an Ubuntu 20.04 operating system. Following [8], each route is divided into two parts: **30%** of the route has already been traveled and **70%** remaining route. The quantities α are [0.1,0.5,0.9].

We compare with two strategies that can improve throughput: (1)**Random**: All samples are not specially processed and are first come, first served. (2)**Greedy**: Since long routes are often difficult to predict, long-distance routes are predicted first. We re-estimate samples that could not be effectively filtered under various strategies. For samples that are successfully filtered, the prediction value is calculated as $\hat{y} - y_{tr}$.

Backbone Architectures We select two groups of backbones including different architecture and integrate them into our framework. As shown in Table 2, we analyze the time complexity of different backbone models.

1. TTE Method: (1)**MLPTTE**: A 16-layer multilayer perceptron with ReLU activation function is used. The specific approach is to estimate the travel time of each road segment separately and sum it as the overall travel time estimate of the route. (2)**WDR** [6]: a wide-deep-recurrent architecture is introduced to handle sparse features, dense features, and road segment sequence features respectively.

[2] `https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i`
[3] `https://gaia.didichuxing.com/`

(3)**WDR-LC** [16]:Enhances WDR by jointly modeling road segments and intersections. (4)**ConSTGAT** [18]: It is a spatiotemporal graph neural network structure that uses graph attention to capture spatiotemporal correlations and the contextual information of the route.
2. ER-TTE Method: (1)**SSML** [7]: It is the first meta-learning model for ER-TTE, which aims to learn meta-knowledge to quickly adapt to users' driving preferences. (2)**MetaER-TTE** [8]: A new adaptive meta-learning model is proposed, which generates cluster-aware initialization parameters through soft clustering and uses distribution-aware adaptive learning rate optimization.

5.2 Overall Effectiveness Comparison

As shown in Table 1, We analyze the effectiveness in two aspects, e.g., Overall comparison and Model fitness.
Overall Comparison. We using different backbone models with UGD improves average MAPE, MAE, RMSE and SR by at least 24.3%, 17.6%, 12.6%, and 18% in on Porto dataset, and 16.7%, 10.3%, 7.5%, and 8.9% on Xian dataset. This demonstrates the effectiveness of UGD in identifying routes needing re-estimation, and achieving SOTA results. In contrast, the greedy strategy performs worse due to higher prediction errors on longer routes.
Model Fitness. The attention-based models ConSTGAT, MetaER-TTE, and SSML perform well Because the attention mechanism can integrate information from different segments to obtain accurate predictions. The MLP-based MLPTTE model uses a pure MLP architecture to independently process the features of each road segment and sum them as an estimated value. Although it ignores the association between road segments, it shows advantages in processing independent and complex road segment features.

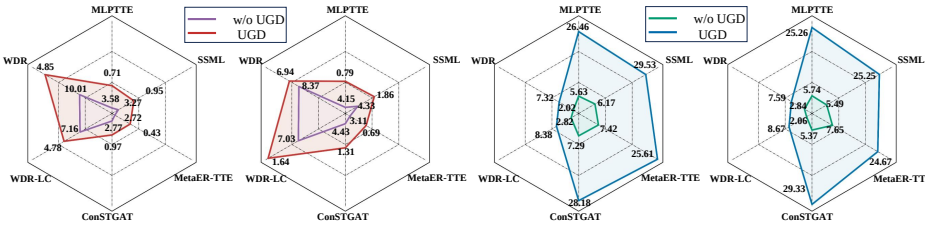
The RNN-based models WDR and WDR-LC have a slightly poorer performance, which may be because each step of the RNN is calculated based only on the current input and the hidden state of the previous time step. Although RNN can partially integrate historical information, its processing method may not effectively capture the complex associations between different locations, especially when it is necessary to capture the state of the intermediate process of the vehicle's driving process. This causes the RNN-based model to inaccurately estimate travel time, making it difficult to construct an effective confidence interval.

5.3 Efficiency Comparison

We analyze the efficiency in inference time and throughput(the number of samples that can be processed per second). As shown in Figure 5, w/o UGD means without UGD which all samples re-estimate. On the Xian dataset, MLPTTE, WDR, WDR-LC, ConSTGAT, and SSML do not require model calls in 77.3%, 32.1%, 24.7%, 77.5%, and 78.1% of cases, respectively. On the Porto dataset, these models perform 80.3%, 74.2%, 37.9%, 36.7%, 73.9%, and 70.7%, respectively. We have the following observations:

Table 1: Overall performance on Porto and Xian dataset

Method	Strategy	Porto				Xian			
		MAPE↓	MAE↓	RMSE↓	SR↑	MAPE↓	MAE↓	RMSE↓	SR↑
MLPTTE	Random	22.25	116.45	255.04	38.53	23.75	200.75	312.71	29.90
	Greedy	23.90	122.03	258.30	36.35	28.81	217.87	309.41	25.98
	UGD	**16.09**	**91.04**	**214.19**	**47.19**	**18.85**	**170.83**	**275.37**	**35.36**
WDR	Random	46.67	343.86	493.88	10.61	29.78	347.44	492.96	18.56
	Greedy	40.22	240.24	359.06	20.86	36.13	370.64	475.88	16.21
	UGD	**20.08**	**111.28**	**236.03**	**36.71**	**20.63**	**193.95**	**327.10**	**32.01**
WDR-LC	Random	32.69	234.09	384.21	24.34	32.72	381.16	530.64	16.18
	Greedy	44.15	287.5	417.41	14.11	36.66	363.42	476.42	19.43
	UGD	**19.23**	**111.27**	**243.54**	**40.54**	**21.88**	**190.82**	**294.56**	**30.34**
ConSTGAT	Random	23.24	121.40	257.38	37.24	25.02	209.19	319.02	28.52
	Greedy	26.84	129.93	252.64	27.08	25.72	215.85	323.57	26.98
	UGD	**15.86**	**91.76**	**217.85**	**48.75**	**20.83**	**176.09**	**269.62**	**32.82**
SSML	Random	22.92	116.21	248.36	36.54	24.39	211.95	332.44	29.26
	Greedy	32.39	134.37	241.69	25.17	25.77	211.56	315.78	27.68
	UGD	**16.89**	**90.15**	**210.03**	**46.30**	**19.14**	**170.10**	**272.28**	**35.40**
MetaER-TTE	Random	25.41	129.86	264.13	31.64	24.20	200.94	304.43	29.76
	Greedy	32.08	145.30	254.39	19.73	27.84	256.36	379.26	21.70
	UGD	**16.36**	**95.96**	**223.77**	**44.98**	**19.98**	**180.27**	**281.72**	**32.42**



(a) Inference time comparison (s) (left: Porto, right: Xian) **(b) Throughput comparison (/s× 10^4) (left: Porto, right: Xian)**

Fig. 5: The efficiency comparison of inference time and throughput.

(1) The experimental results demonstrate that integrating UGD inference time and throughput by at least 1.49 times and 2.97 times, respectively, on the Porto dataset, and by 1.2 times and 2.67 times on the Xian dataset.

(2) Three Attention-based models and MLPTTE can filter more samples, they can make greater progress. There are more routes need to re-estimation with two RNN-based methods, resulting in smaller improvements. However, they still improved by 1.2 and 2.67 times in inference time and throughput.

The improvements are attributed to the ability of UGD to minimize unnecessary inference operations, thereby optimizing the use of computational resources and accelerating system response time.

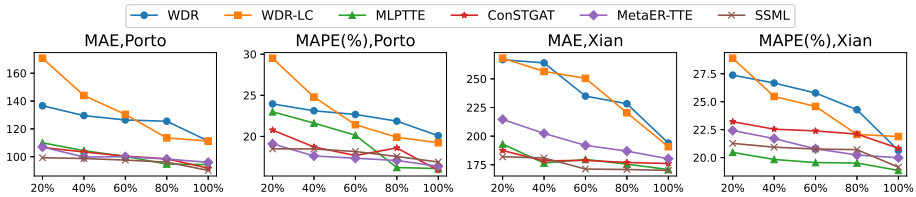


Fig. 6: MAPE & MAE vs. the Scalability.

Table 2: Backbone complexity analysis. n refers to the number of segments, h refer to the feature dimensions.

Method	Complexity	Components
MLPTTE	$O(nh^2)$	MLP
WDR	$O(nh^2)$	RNN
WDR-LC	$O(nh^2)$	RNN
ConSTGAT	$O(n^2h)$	Attention
SSML	$O(n^2h)$	Attention
MetaER-TTE	$O(n^2h)$	Attention

Table 3: Ablation study on the FTML and replace quantile regression to MIS.

Dataset	Model	MAPE	MAE
Porto	w/o FTML	17.58	94.43
	MIS	18.46	96.88
	U-ERTTE	**16.89**	**90.15**
Xian	w/o FTML	20.36	176.81
	MIS	24.75	228.07
	U-ERTTE	**19.14**	**170.10**

5.4 Scalability Comparison

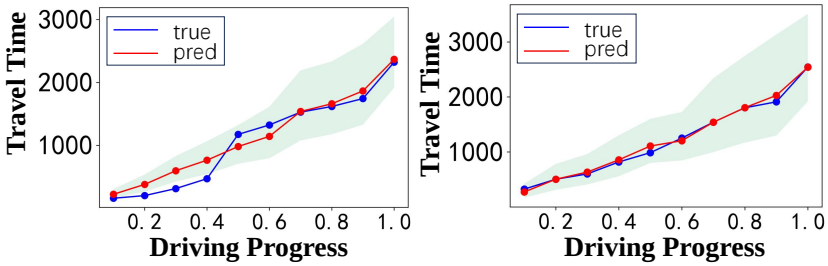
To validate the scalability of our framework, we conduct experiments using 20%, 40%, 60%, and 80% of the training data. As shown in Figure 6, we observe that:

(1) More data covers a wider range of scenarios, allowing the models to learn more effectively.

(2) Attention-based models achieve good performance even with only 20% of the data, whereas RNN-based models require larger amounts of data to see significant improvements.

5.5 Ablation Study

To validate our method, we conduct ablation studies on SSML with two modifications: (1) w/o FTML: Removing FTML, and (2) MIS: Replacing the quantile loss with the Mean Interval Score (MIS) loss [26]. As shown in Table 3, removing FTML results in decreased performance, indicating that FTML helps the model capture both general patterns and task-specific features. Moreover, substituting quantile loss with MIS significantly reduces both prediction effectiveness and uncertainty quantification. This demonstrates that quantile loss not only provides accurate confidence intervals to enhance efficiency but also improves overall prediction performance.



(a) Xian Oct. 2nd 15:39 2322s (b) Xian Oct. 14th 23:06 2544s

Fig. 7: Visualization of U-ERTTE in ER-TTE process. (The title of each subfigure is labeled in the form of "City, Date, Departure time, and Travel time".)

5.6 Online Test

We conduct an online test simulation on the Xian dataset using a framework with SSML as the backbone model. In this simulation, we select 100 routes that feature both temporal (e.g., weekdays/weekends) and spatial (e.g., short/long distances) characteristics. Each route is divided into 10 equal segments (10% intervals) to assess online efficiency. Before departure, the model generates confidence intervals for the entire route, and at each 10% interval (from 10% to 90%), an ER-TTE request is made—resulting in 9 queries per route. Although this setup would normally yield 900 requests (9 queries $\times$ 100 routes), our framework reduces the number to 334, significantly improving efficiency and throughput.

Figure 7 illustrates the relationship between travel time and route completion percentage for two routes. In the left figure, only 3 initial requests needed re-estimation when actual travel times deviated from predicted confidence intervals, primarily due to holiday traffic in Xian and initial acceleration phases. In contrast, the right figure, which depicts late-night travel, did not require any re-estimations due to stable traffic conditions. These results demonstrate that our framework can efficiently adapt to varying traffic conditions, initiating re-estimations only when necessary, thereby ensuring flexibility and efficiency.

6 Conclusion

In this paper, we introduce the U-ERTTE framework, which enhances ER-TTE by integrating UGD and FTML. This framework achieves efficient and effective ER-TTE, an area not previously explored. UGD provides confidence intervals that help the system determine when re-estimation is needed, optimizing computational resource use. FTML improves effectiveness by learning general driving patterns and adapting to specific tasks. This framework advances the state-of-the-art in ER-TTE and offers a practical, scalable solution to enhance system efficiency and performance. In the future, we aim to integrate more diverse real-time data sources, improve scalability, effectiveness, and refine decision mechanisms to handle rare traffic anomalies.

Acknowledgments

This work is supported by the Beijing Natural Science Foundation (Grant No. 4242029).

References

[1] Haitao Yuan and Guoliang Li. "A survey of traffic prediction: from spatio-temporal data to intelligent transportation". In: *Data Science and Engineering* 6.1 (2021), pp. 63–85.

[2] Christian S Jensen et al. "Routing with Massive Trajectory Data". In: *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE. 2024, pp. 5542–5547.

[3] Haitao Yuan et al. "Automatic road extraction with multi-source data revisited: completeness, smoothness and discrimination". In: *Proceedings of the VLDB Endowment* 16.11 (2023), pp. 3004–3017.

[4] Shengnan Guo et al. "Self-supervised spatial-temporal bottleneck attentive network for efficient long-term traffic forecasting". In: *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE. 2023, pp. 1585–1596.

[5] Haitao Yuan, Gao Cong, and Guoliang Li. "Nuhuo: An Effective Estimation Model for Traffic Speed Histogram Imputation on A Road Network". In: *Proceedings of the VLDB Endowment* 17.7 (2024), pp. 1605–1617.

[6] Zheng Wang, Kun Fu, and Jieping Ye. "Learning to estimate the travel time". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 858–866.

[7] Xiaomin Fang et al. "Ssml: Self-supervised meta-learner for en route travel time estimation at baidu maps". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 2840–2848.

[8] Yu Fan et al. "MetaER-TTE: An Adaptive Meta-learning Model for En Route Travel Time Estimation." In: *IJCAI*. 2022, pp. 2023–2029.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[10] Yilun Wang, Yu Zheng, and Yexiang Xue. "Travel time estimation of a path using sparse trajectories". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 25–34.

[11] Yan Lin et al. "Origin-destination travel time oracle for map-based services". In: *Proceedings of the ACM on Management of Data* 1.3 (2023), pp. 1–27.

[12] Xiaowei Mao et al. "GMDNet: A graph-based mixture density network for estimating packages' multimodal travel time distribution". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 4. 2023, pp. 4561–4568.

[13] Xiaowei Mao et al. "Drl4route: A deep reinforcement learning framework for pick-up and delivery route prediction". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 4628–4637.

[14] Xiaowei Mao et al. "DutyTTE: Deciphering Uncertainty in Origin-Destination Travel Time Estimation". In: *arXiv preprint arXiv:2408.12809* (2024).

[15] Wei Chen et al. "Deep learning for trajectory data management and mining: A survey and beyond". In: *arXiv preprint arXiv:2403.14151* (2024).

[16] Xiaowei Mao et al. "Estimated time of arrival prediction via modeling the spatial-temporal interactions between links and crosses". In: *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 2021, pp. 658–661.

[17] Austin Derrow-Pinion et al. "Eta prediction with graph neural networks in google maps". In: *Proceedings of the 30th ACM international conference on information & knowledge management*. 2021, pp. 3767–3776.

[18] Xiaomin Fang et al. "Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2697–2705.

[19] Haitao Yuan, Guoliang Li, and Zhifeng Bao. "Route travel time estimation on a road network revisited: Heterogeneity, proximity, periodicity and dynamicity". In: *Proceedings of the VLDB Endowment* 16.3 (2022), pp. 393–405.

[20] Zebin Chen et al. "Interpreting trajectories from multiple views: A hierarchical self-attention network for estimating the time of arrival". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 2771–2779.

[21] Xiucheng Li et al. "Learning travel time distributions with deep generative model". In: *The World Wide Web Conference*. 2019, pp. 1017–1027.

[22] Jakob Gawlikowski et al. "A survey of uncertainty in deep neural networks". In: *Artificial Intelligence Review* 56.Suppl 1 (2023), pp. 1513–1589.

[23] Hao Wang and Dit-Yan Yeung. "Towards Bayesian deep learning: A framework and some existing methods". In: *IEEE Transactions on Knowledge and Data Engineering* 28.12 (2016), pp. 3395–3408.

[24] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[25] Roger Koenker and Gilbert Bassett Jr. "Regression quantiles". In: *Econometrica: journal of the Econometric Society* (1978), pp. 33–50.

[26] Dongxia Wu et al. "Quantifying uncertainty in deep spatiotemporal forecasting". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 1841–1851.

[27] Tim Pearce et al. "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach". In: *International conference on machine learning*. PMLR. 2018, pp. 4075–4084.