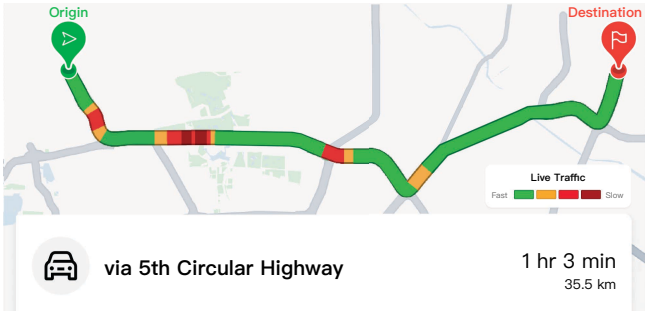
SSML: Self-Supervised Meta-Learner for En Route Travel Time Estimation at Baidu Maps

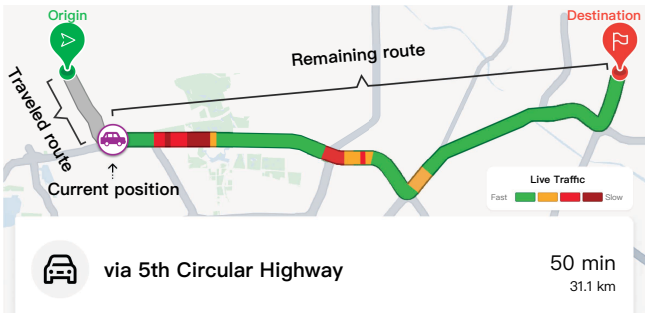
Xiaomin Fang, Jizhou Huang*, Fan Wang, Lihang Liu, Yibo Sun, Haifeng Wang
Baidu Inc., China
{fangxiaomin01,huangjizhou01,wang.fan,liulihang,sunyibo,wanghaifeng}@baidu.com

ABSTRACT

Travel time estimation (TTE) is one of the most critical modules at Baidu Maps, which plays a vital role in intelligent transportation services such as route planning and navigation. During the driving en route, the navigation system of Baidu Maps can provide real-time estimations on when a user will arrive at the destination. It automatically recalculates and updates the remaining travel time from the driver's current position to the destination (hereafter referred to as remaining route) every few minutes. The previously deployed TTE model at Baidu Maps, i.e., ConSTGAT [4], takes the remaining route as well as the current time as input and provides the corresponding estimated time of arrival. However, it ignores the route that has been already traveled from the origin to the driver's current position (hereafter referred to as traveled route), which could contribute to improving the accuracy of time estimation. In this work, we believe that the traveled route conveys valuable evidence that could facilitate the modeling of driving preference and take that into consideration for the task of en route travel time estimation (ER-TTE). This task is non-trivial because it requires adapting fast to a user's driving preference using a few observed behaviors in the traveled route. To this end, we frame ER-TTE as a few-shot learning problem and consider the observed behaviors in the traveled route as training examples while the future behaviors in the remaining route as test examples. To tackle the few-shot learning problem, we propose a novel model-based meta-learning approach, called SSML, to learn the meta-knowledge so as to fast adapt to a user's driving preference and improve the time estimation of the remaining route. SSML leverages the technique of self-supervised learning, which is equivalent to generating a significant number of synthetic learning tasks, to further improve the performance. Extensive offline tests conducted on large-scale real-world datasets collected from Baidu Maps demonstrate the superiority of SSML. The online tests before deploying in production were successfully performed, which confirms the practical applicability of SSML.



(a) An example of travel time estimation (TTE).



(b) An example of en route travel time estimation (ER-TTE).

Figure 1: Travel time estimation function at Baidu Maps.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Applied computing** → *Transportation.*

KEYWORDS

ETA, travel time estimation, meta-learning, self-supervised learning, transportation, Baidu Maps

ACM Reference Format:
Xiaomin Fang, Jizhou Huang, Fan Wang, Lihang Liu, Yibo Sun, Haifeng Wang. 2021. 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 and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467060

1 INTRODUCTION

Travel time estimation (TTE) is one of the most critical modules at Baidu Maps, which plays a vital role in intelligent transportation

*Corresponding author: Jizhou Huang.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467060

services such as route planning and navigation. TTE aims to predict the travel time in accordance with a given route and departure time, which greatly helps the drivers to know the traffic condition in advance and plan their trips wisely. Figure 1(a) shows an example of the TTE function at Baidu Maps, where the driver departing from the origin could arrive at the destination within 1 hour 3 minutes. In order to help a driver arrive at her destination timely, it is important to provide not only a once-only time estimation before the user starts her trip but, more importantly, real-time estimations with the ability of self-updating during the user's driving en route. Figure 1(b) shows an example of the en route travel time estimation (ER-TTE) function at Baidu Maps, where the driver could arrive at the destination within 50 minutes from the current position. This function automatically recalculates and updates the remaining travel time from the driver's current position to the destination (hereafter referred to as remaining route) every few minutes. In this paper, we address the task of ER-TTE, which aims to predict the travel time in accordance with (1) the remaining route, (2) departure time, and (3) the route that has been already traveled from the origin to the driver's current position (hereafter referred to as traveled route) during a user's driving en route. From the above illustrations, it is easily seen that ER-TTE is just a special case of TTE. The only difference is whether the traveled route is taken into consideration when estimating travel time.

To predict the en route travel time, the previously deployed TTE model at Baidu Maps, i.e., ConSTGAT [4], takes the remaining route as well as the current time as input and provides the corresponding estimated time of arrival. However, it ignores the traveled route, which conveys valuable evidence that could facilitate the modeling of driving preference. We believe that the observed behaviors in the traveled route could infer the driver's future behaviors in the remaining route. The effect of the observed behaviors in the traveled route is illustrated by Figure 2. Before driver A and driver B start their trips, the navigation service estimates the entire route's travel time, which is 50 minutes. Then, as driver B drove faster than driver A in the first 5 minutes, it is likely that driver B would drive faster than driver A in their remaining routes. Consequently, the remaining travel time of driver A (55 minutes) will be much longer than that of driver B (40 minutes). In order to provide the user with accurate and reliable real-time estimations during her driving en route, it is important to take the traveled route into consideration.

The task of ER-TTE is challenging because it requires adapting fast to a user's driving preference using a few observed behaviors in the traveled route. A user's driving preferences are unstable and may vary from one circumstance to another. Moreover, as a route generally contains dozens or hundreds of adjacent road segments, the observed behaviors in the traveled route are insufficient for estimating the travel time of the road segments in the remaining route, especially when the user has just started her trip. To this end, we frame ER-TTE as a few-shot learning problem and consider the observed behaviors in the traveled route as training examples while the future behaviors in the remaining route as test examples.

To tackle the few-shot learning problem, we propose a novel model-based meta-learning approach, called SSML, to learn the meta-knowledge to fast adapt to a user's driving preference. SSML is based on our previous work [4]. We regard the route, the departure time, and a user's observed behaviors in the traveled route as

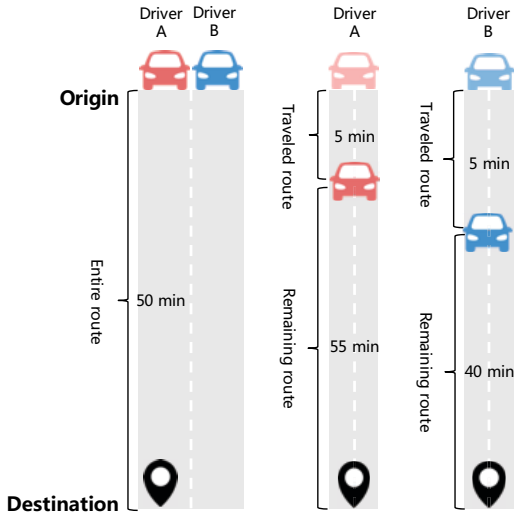


Figure 2: Illustration of the effect of the observed behaviors in the traveled route.

a circumstance and then estimate the remaining route's travel time. Each circumstance can be seen as a learning task in the settings of meta-learning. The meta-knowledge learned from a large number of learning tasks can contribute to fast adapting to the users' driving preferences in new circumstances. Besides, SSML leverages the technique of self-supervised learning, which is widely used in natural language processing. To improve the performance, we further design a self-supervised learning method that estimates part of the traveled behaviors from the other part of traveled behaviors, which is equivalent to generating a significant number of synthetic learning tasks for meta-learning. Extensive offline tests conducted on large-scale real-world datasets collected from Baidu Maps demonstrate the superiority of SSML.

Our contributions can be summarized as follows:

- **Potential impact:** We propose a novel model-based meta-learning approach, named SSML, as an industrial solution to the task of en route travel time estimation (ER-TTE). To the best of our knowledge, this work is a pioneering attempt at exploring the practical applicability of using the meta-learning paradigm to address this task.
- **Novelty:** The design and implementation of SSML are driven by the novel ideas that take advantage of the observed behaviors in the traveled route to infer a user's driving preference, and regard the task of ER-TTE as a few-shot learning problem by applying model-based meta-learning paradigm with self-supervised learning technique to fast adapt to the user's driving preference en route.
- **Technical quality:** Extensive offline tests conducted on large-scale real-world datasets collected from Baidu Maps demonstrate the superiority of SSML. The online tests also confirm the practical applicability of SSML.
- **Reproducibility**: We have released the source codes of ConSTGAT [4] and SSML at https://github.com/PaddlePaddle/Research/tree/master/ST_DM/KDD2020-ConSTGAT/ and

https://github.com/PaddlePaddle/Research/tree/master/ST_DM/KDD2021-SSML/.

2 PROBLEM DEFINITION

In this section, we first present the task of ER-TTE and formalize ER-TTE in the meta-learning settings. Then, we introduce the model-based meta-learning.

2.1 En Route Travel Time Estimation

2.1.1 TTE. The TTE function of a navigation service estimates the travel time of a given route r according to the departure time t. Generally, a route consists of dozens or hundreds of adjacent road segments. For the sake of clarity, "road segments" will be short for "links" hereafter. We represent a route r as a sequence of links $r = [l_1, l_2, \cdots, l_{n_r}]$, where n_r is the number of links in the route. Formally, the task of TTE takes a route r consisting of n_r links and a departure time t as input and outputs a sequence of the link's travel times $Y = [y_1, y_2, \cdots, y_{n_r}]$. An example is defined as a triple (t, l, y), where t denotes the time, l denotes the link to be estimated, and y denotes the travel time of link l. We estimate the travel time y of link l with respect to the request time t.

2.1.2 ER-TTE. When a user drives en route, the ER-TTE function of a navigation service recalculates the route's remaining travel times whenever it receives a request. We denote the time sequence that the requests are made as $[t_1, t_2, \cdots, t_m]$, where m denotes the number of the requests. To recalculate the remaining travel time, we consider the request time, the user's current position, and the user's driving behaviors in the traveled route. Assume that the navigation service receives a request at time t, the route r can be divided into two parts, $r^t_{traveled} = [l_1, l_2, \cdots, l_{i^t-1}]$ consisting of the links in the traveled route, and $r^t_{remain} = [l_{i^t}, l_{i^t+1}, \cdots, l_{n_r}]$ consisting of the links in the remaining route. Here, i^t is the index of the first link in r^t_{remain}. We regard the user's past driving behaviors as $Y^t_{traveled} = [y_1, y_2, \cdots, y_{i^t-1}]$, where each y_i is the actual travel time of each $l_i \in r^t_{traveled}$. Formally, the task of ER-TTE takes a route r consisting of n_r links, a request time t, and the observed driving behaviors $Y^t_{traveled}$ as input and outputs a sequence of travel times $Y^t_{remain} = [y_{i^t}, y_{i^t+1}, \cdots, y_{n_r}]$, where each y_i is the predicted travel time of each $l_i \in r^t_{remain}$.

2.2 Meta-Learning Settings

Each request is taken as a circumstance, and we estimate the remaining travel time under different circumstances. For the convenience of formulation, we define a circumstance as a triple $c = (r, t, Y^t_{traveled})$, where r is the route, t is the request time, and $Y^t_{traveled}$ is the travel times of the links in the traveled route. For that circumstance, ER-TTE estimates the remaining travel times Y^t_{remain} according to the observed behaviors $Y^t_{traveled}$ in the traveled route.

Each circumstance can be seen as a few-shot learning problem as the number of observed behaviors in the traveled route under a circumstance is limited. We consider each circumstance of ER-TTE as a learning task in the meta-learning settings. Following previous meta-learning studies [5], given a circumstance $c = (r, t, Y^t_{traveled})$, we refer to the learning task's training set as the support set and its test set as the query set. The demonstration of the support set and the query set of a given circumstance is shown by Figure 3.

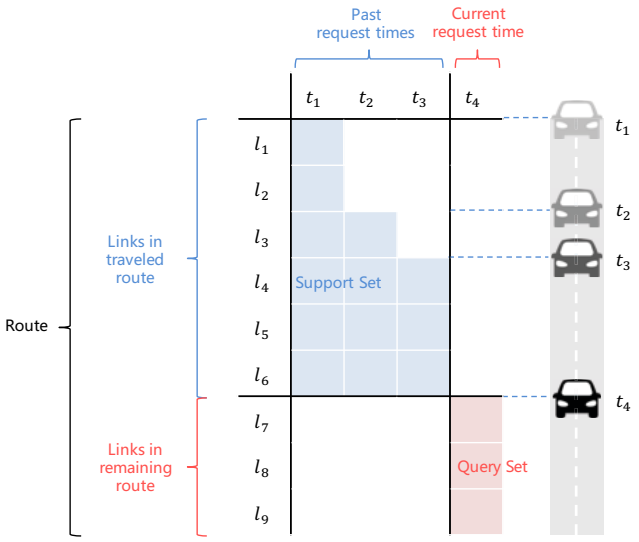


Figure 3: Demonstration of a given circumstance $c = (r, t_4, Y^{t_4}_{traveled})$. At current request time t_4, the user arrives at link l_7. We regard the observed behaviors collected from the traveled route at past request times t_1, t_2, and t_3 as support examples to train the meta-learner and regard the behaviors in the remaining route at current request time t_4 as query examples to evaluate their travel times.

The support set of a circumstance $c = (r, t, Y^t_{traveled})$ consists of the user's observed behaviors $Y^t_{traveled}$ in the traveled route with respect to the past request times. We use t' to denote a past request time ($t' < t$). Given a link in the traveled route, as the features (e.g., traffic conditions and request times) for different past request times are different, we can construct multiple support examples for the same link. We take links in the traveled route as well as the observed behaviors with respect to different past request times as support examples. Formally, the support set of a circumstance $c = (r, t, Y^t_{traveled})$ is defined as:

$$D^S_c = \{(t', l_j, y_j) | t' < t, i^{t'} \leq j < i^t\}. \quad (1)$$

More concretely, for a support example (t', l_j, y_j), at past request time t', the travel times of the links in $r^{t'}_{remain} = [l_{i^{t'}}, l_{i^{t'}+1}, \cdots, l_{n_r}]$ are estimated by the navigation service. Thus, j should satisfy $i^{t'} \leq j$. Similarly, at current request time t, only the ground-truth travel times of the links in $r^t_{traveled} = [l_1, l_2, \cdots, l_{i^t-1}]$ can be collected and used for training. Thus, j should satisfy $j \leq i^t$.

Take Figure 3 as an instance, the blue grids represent the support examples for circumstance $c = (r, t_4, Y^{t_4}_{traveled})$ with current request time t_4. From the figure, we can see that the observed behaviors in past request times $[t_1, t_2, t_3]$ are taken as the support examples. The support set consists of the $l_1 \sim l_6$, $l_3 \sim l_6$, and $l_4 \sim l_6$ for past request times t_1, t_2, and t_3, respectively.

The query set of a circumstance $c = (r, t, Y^t_{traveled})$ consists of the user's future behaviors Y^t_{remain} in the remaining route with

respect to current request time t. Formally, the query set D_c^Q of circumstance $c = (r, t, Y^t_{traveled})$ is defined as:

$$D_c^Q = \{(t, l_j, y_j) | i^t \le j\}. \tag{2}$$

For a query example (t, l_j, y_j), at current request time t, the travel times of the links in $r^t_{remain} = [l_{i^t}, l_{i^t+1}, \cdots, l_{n_r}]$ are estimated by the navigation service. Thus, j should satisfy $i^t \le j$.

In Figure 3, the red grids represent the query examples for circumstance $c = (r, t_4, Y^{t_4}_{traveled})$ with current request time t_4. The query set consists of the $l_7 \sim l_9$ for current request times t_4.

2.3 Model-Based Meta-Learning

Here we introduce the model-based meta-learning method that learns the meta-knowledge and adapts it to new circumstances. A circumstance in ER-TTE is taken as a learning task in this paper.

The source tasks and target tasks in meta-learning settings are as follows. The source tasks $\mathcal{D}_{src} = \{(D_c^S, D_c^Q)\}$ is used for meta-training. The target tasks $\mathcal{D}_{tgt} = \{(D_c^S, D_c^Q)\}$ is used for evaluating the meta-learner. We learn the meta-knowledge from the source tasks and evaluate the adaption abilities by the target tasks.

To simplify the notations, for circumstance c, we use $(x^S_{c,i}, y^S_{c,i})$ to denote the i-th support example and its label, where $x^S_{c,i} = (t, l)$ consisting of request time t and link l, and $y^S_{c,i}$ is the travel time. Similarly, we use $(x^Q_{c,i}, y^Q_{c,i})$ to indicate the i-th query example and its label. Thus, for circumstance $c = (r, t, Y^t_{traveled})$, its support set and the query set are redefined as $D_c^S = \{(x^S_{c,i}, y^S_{c,i})|_{i=1}^{n_c^S}\}$ and $D_c^Q = \{(x^Q_{c,i}, y^Q_{c,i})|_{i=1}^{n_c^Q}\}$, respectively, where n_c^S and n_c^Q denote the sizes of D_c^S and D_c^Q.

We use the similar setting of the model-based meta-learner proposed by Mishra et al. [11]. During the meta-training process, we minimize the meta-learner's objective function by:

$$\mathcal{L}(\mathcal{D}_{src}; \theta) = \frac{1}{|\mathcal{D}_{src}|} \sum_{(D_c^S, D_c^Q) \in \mathcal{D}_{src}} \frac{1}{|D_c^Q|} \sum_{i=1}^{|D_c^Q|} L(D_c^S, x^Q_{c,i}, y^Q_{c,i}; \theta), \tag{3}$$

where θ is the parameters to be optimized. Here, we use Huber loss [8], a widely-used loss function for regression problems, as the loss function, which is defined as:

$$L(D_c^S, x^Q_{c,i}, y^Q_{c,i}; \theta) = huber\_loss(f_\theta(D_c^S, x^Q_{c,i}), y^Q_{c,i}), \tag{4}$$

where $f_\theta(D_c^S, x^Q_{c,i})$ is the function that represents the model to be learned, which takes the support set D_c^S as well as the query example $x^Q_{c,i}$ as input and predicts the travel time. Besides, the Huber loss $huber\_loss(\hat{y}, y)$ is defined as:

$$huber\_loss(\hat{y}, y) = \begin{cases} \frac{1}{2}(\hat{y} - y)^2 & |\hat{y} - y| < \delta, \\ \delta(|\hat{y} - y| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases} \tag{5}$$

Here, δ is a hyper-parameter of Huber loss to control the impact of the outliers. $\hat{y}$ represents the predicted travel time, and y represents the ground truth.

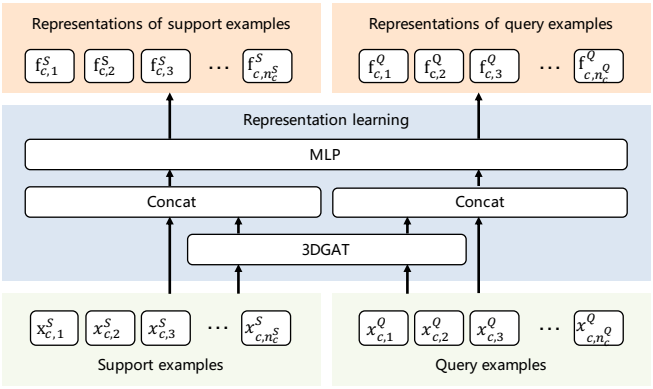


Figure 4: Representation learning of the support examples and query examples of a given circumstance c.

During the test process, we evaluate the trained meta-learner's performance on target tasks $\mathcal{D}_{tgt}$.

3 SELF-SUPERVISED META-LEARNER

We first present representation learning that outputs the representations of the support examples and query examples. Then, we describe a novel model-based meta-learner, called SSML, which incorporates self-supervised learning to the meta-learning paradigm to further enhance the ability of learning the meta-knowledge.

3.1 Representation Learning

We learn the representations of the support and query examples, which are the input of the meta-learner. The representations are generated according to the observed driving behaviors in the traveled route and the features used in ConSTGAT [4], such as road network, departure time, and traffics.

To predict the travel time of a query example of a given circumstance $c = (r, t, Y^t_{traveled})$, we utilize the support set D_c^S and the query example $x^Q_{c,i}$ itself as input. Then, support set D_c^S and the query set D_c^Q of circumstance c can be represented by:

$$\begin{aligned} D_c^S &= [(x^S_{c,1}, y^S_{c,2}), (x^S_{c,2}, y^S_{c,2}), \cdots, (x^S_{c,n_c^S}, y^S_{c,n_c^S})], \\ D_c^Q &= [(x^Q_{c,1}, y^Q_{c,2}), (x^Q_{c,2}, y^Q_{c,2}), \cdots, (x^Q_{c,n_c^Q}, y^Q_{c,n_c^Q})]. \end{aligned} \tag{6}$$

We use 3DGAT, a 3D-attention mechanism proposed in our previous work [4], to capture the spatial-temporal relations of the traffic conditions. Then, we use a multi-layer perceptron to learn the representations of the support and query examples. The network structure to learn the representations of the examples is shown by Figure 4. Given a circumstance c, the representation of support example $x^S_{c,i}$ and query example $x^Q_{c,i}$ are defined as:

$$\begin{aligned} f^S_{c,i} &= MLP(Concat(3DGAT(x^S_{c,i}), x^S_{c,i})), \\ f^Q_{c,i} &= MLP(Concat(3DGAT(x^Q_{c,i}), x^Q_{c,i})), \end{aligned} \tag{7}$$

where $3DGAT$ denotes the 3D-attention mechanism, MLP denotes multi-layer perceptron, and $Concat$ denotes concatenation operator. The representations of support examples $F_c^S = [f^S_{c,1}, f^S_{c,2}, \cdots, f^S_{c,n_c^S}]$

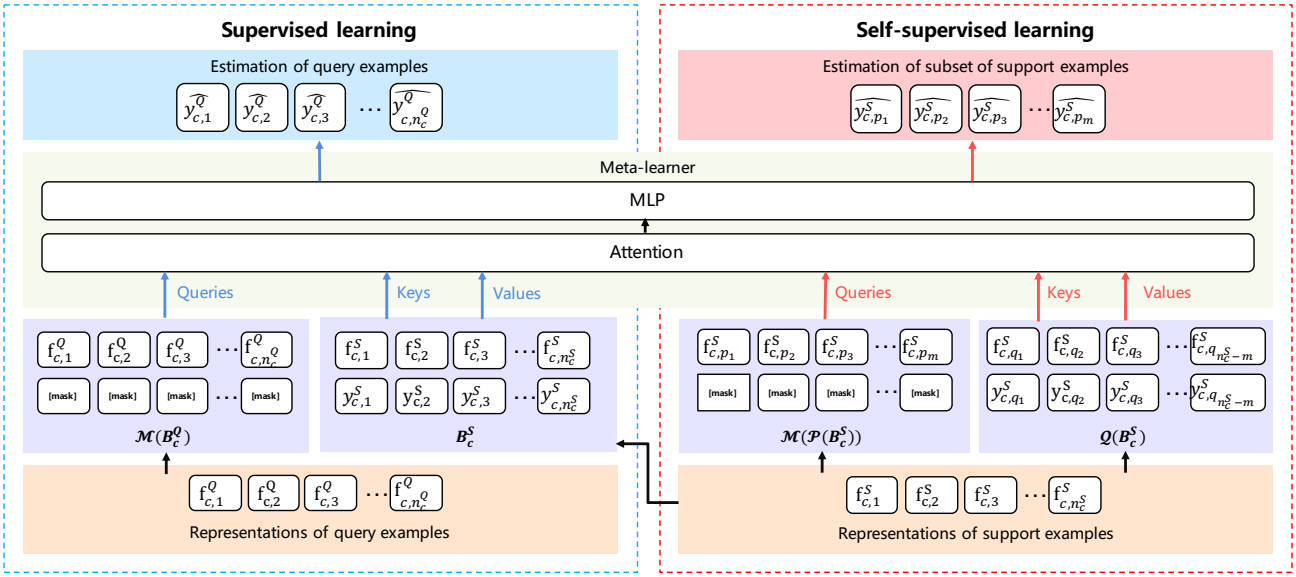


Figure 5: The framework of self-supervised meta-learner combining supervised learning and self-supervised learning. In supervised learning, we estimate the labels of the query examples from the support examples. In the self-supervised learning, we estimate the labels of the randomly sampled support examples from the other support examples.

and their labels $Y_c^S = [y_{c,1}^S, y_{c,2}^S, \cdots, y_{c,n_c^S}^S]$, as well as the representations of the query examples $F_c^Q = [f_{c,1}^Q, f_{c,2}^Q, \cdots, f_{c,n_c^Q}^Q]$ are taken as input of the meta-learner.

3.2 Self-Supervised Meta-Learner

We combine supervised learning and self-supervised learning in the meta-learner for ER-TTE. The framework of self-supervised meta-learner, SSML, is shown by Figure 5. We first present supervised learning that directly learns the meta-knowledge and then describe the approach that constructs self-supervised learning tasks to enhance the meta-learner's ability.

3.2.1 Supervised Learning. The left part of Figure 5 demonstrates the way that SSML learns the meta-knowledge by supervised learning. Given a circumstance c, in order to fast adapt to the user's driving preference, we use the attention mechanism [17] to capture the associations between the support examples and query examples. We aim at learning the user's driving preference from her observed driving behaviors in the traveled route to predict her future behaviors in the remaining route.

First, we introduce observed behavior $b_{c,i}^S$ in the traveled route and future behavior $b_{c,i}^Q$ in the remaining route. Here, the observed behavior is denoted by $b_{c,i}^S = (f_{c,i}^S, y_{c,i}^S)$, while the future behavior is denoted by $b_{c,i}^Q = (f_{c,i}^Q, y_{c,i}^Q)$. Then, for circumstance c, the sequence of observed behaviors is represented by $B_c^S = (F_c^S, Y_c^S)$ and the sequence of future behaviors is represented by $B_c^Q = (F_c^Q, Y_c^Q)$. Since the labels of the query examples are unknown to the model, we introduce placeholder $[mask]$ to denote the unknown label and M to denote the sequence of unknown labels. Then, the future behaviors are rewritten as $\mathcal{M}(B_c^Q) = (F_c^Q, M)$, which replaces the unknown labels of query examples Y_c^Q by M, where squiggles $\mathcal{M}(\cdot)$ is used to distinguish the rewritten behavior sequence from the original one.

For attention mechanism, we regard the sequence of a user's future behaviors $\mathcal{M}(B_c^Q)$ as the queries and the sequence of the user's observed behaviors B_c^Q as the keys and values, so as to learn the user's driving preference. We can leverage the observed behaviors to predict the user's future behaviors. After that, we apply MLP and predict the travel times of all the query examples. This process can be formalized as:

$$\hat{Y_c^Q} = MLP(Attention(\mathcal{M}(B_c^Q), B_c^S, B_c^S)), \tag{8}$$

where $\hat{Y_c^Q}$ denotes the predicted travel times of the query examples, and $Attention$ denotes attention layer. We use 3-layer MLP with ReLU [13] as the activation. The hidden size of the MLP is set to 64. The formulation of attention layer $Attention$ is defined as:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{(\mathbf{Q}\mathbf{W}^Q)(\mathbf{K}\mathbf{W}^K)^T}{\sqrt{d}})(\mathbf{V}\mathbf{W}^V), \tag{9}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are matrices and denote the queries, keys, and values of the attention mechanism, respectively. $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are parameter matrices to be learned. d is the hidden dimension of the attention mechanism. By applying attention mechanism, the query examples are able to learn from similar support examples to adjust their travel times.

3.2.2 Self-Supervised Learning. On the other hand, the right part of Figure 5 shows the self-supervised learning component of SSML. Inspired by the mask language model from BERT [2], we apply self-supervised learning to learn from the support set itself. We construct a great number of self-supervised learning tasks by predicting the labels of the subset of the observed behaviors from the

Table 1: Statistics of the datasets.

City	Basic information			t = 5 min		t = 10 min		t = 15 min	
	#Links	#Source tasks	#Target tasks	Support	Query	Support	Query	Support	Query
Taiyuan	216,208	2,556,593	714,464	10.572	61.806	41.130	44.921	82.671	32.791
Huizhou	346,396	2,136,949	608,723	10.544	49.180	37.200	34.606	67.739	26.652
Hefei	376,310	2,189,020	552,924	9.217	41.618	30.485	30.885	54.604	24.618

other observed behaviors. For example, given the observed behavior set B_c^S, we randomly sample a subset. We denote the sampled observed behaviors by $\mathcal{P}(B_c^S)$ and the other observed behaviors by $\mathcal{Q}(B_c^S)$, and $B_c^S = \mathcal{P}(B_c^S) + \mathcal{Q}(B_c^S)$. The indexes of the sampled observed behaviors in $\mathcal{P}(B_c^S)$ are denoted by $[p_1, p_2, \cdots, p_m]$ and the indexes of the other observed behaviors in $\mathcal{Q}(B_c^S)$ are denoted by $[q_1, q_2, \cdots, q_{m-n_c^S}]$, as shown by Figure 5. We assume the labels in $\mathcal{P}(B_c^S)$ are unknown and replace the labels by M. We estimate their labels $\mathcal{P}(\hat{Y}_c^S)$ according to the other observed behaviors $\mathcal{Q}(B_c^s)$. Then, for the attention mechanism, we regard the replaced behavior subset $\mathcal{M}(\mathcal{P}(B_c^S))$ as the queries and the other observed behaviors $\mathcal{Q}(B_c^S)$ as the keys and values, just like the supervised learning method presented. We predict the travel times of the sampled subset $\mathcal{P}(B_c^S)$ by:

$$\mathcal{P}(\hat{Y}_c^S) = MLP(Attention(\mathcal{M}(\mathcal{P}(B_c^S)), \mathcal{Q}(B_c^S), \mathcal{Q}(B_c^S))), \quad (10)$$

where the definitions of *MLP* and *Attention* are the same as Eq. 8.

For each circumstance, we can sample multiple subsets of the observed behaviors. This enables us to construct multiple self-supervised learning tasks, which can contribute to improving the ability of adapting fast to a user's driving preference.

3.2.3 Discussion. The parameters of the *Attention* and *MLP* are shared between supervised learning (estimating query examples from support examples) and self-supervised learning (estimating subset of support examples from the other support examples). Both can contribute to learning the meta-knowledge that learns personalized adaption from only a few labeled examples. The subset $\mathcal{P}(B_c^S)$ in self-supervised learning is an analogy to B_c^Q in supervised learning, while the other support behaviors $\mathcal{Q}(B_c^S)$ in self-supervised learning is an analogy to B_c^S in supervised learning.

Besides, it enables us to design a great number of self-supervised learning tasks, which is equivalent to adding a significant number of synthetic learning tasks in meta-learning. The synthetic learning tasks play the role of data enhancement. SSML can be applied to other problems with similar settings.

4 EXPERIMENTS

4.1 Experimental Settings

We collected the records of ER-TTE in three cities: Taiyuan, Huizhou, and Hefei in China, ranging from Sep 1st to Sep 28th, 2019, from Baidu Maps. Records of the first three weeks are used for training, and those of the last week are used for evaluation. The statistics of the datasets are shown in Table 1. We use over two million source tasks for each city to train the models and hundreds of thousands of target tasks to evaluate the meta-models. Then, we produce three datasets for each city with different request times, t = 5 min, t = 10 min, and t = 15 min. For example, for dataset t = 5 min, all the request times of corresponding circumstances are set to 5 minutes. The last six columns of Table 1 present the average sizes of the support sets and query sets with different request times. As request time t goes up, there are more support examples and fewer query examples for each city.

Furthermore, given a circumstance $c = (r, t, Y^t_{traveled})$, instead of directly estimating the travel time of each link in the remaining route r^t_{remain}, we adjust the label of the examples. We first calculate the bias between the ground-truth travel time and the travel time predicted by ConSTGAT before the user starts the trip. Then, to reduce the bias's variance, we normalize bias by dividing the length of the corresponding link and taking it as the label.

For each circumstance, we use root mean square error (RMSE) and Mean Average Error (MAE) to evaluate the performance. Given a circumstance c, we calculate RMSE score and MAE score by:

$$\begin{aligned} \text{RMSE}(\hat{y^Q_{c,i}}, y^Q_{c,i}) &= \sqrt{\frac{1}{n_c^Q}\sum_{i=1}^{n_c^Q}(y^Q_{c,i} - \hat{y^Q_{c,i}})^2}, \\ \text{MAE}(\hat{y^Q_{c,i}}, y^Q_{c,i}) &= \frac{1}{n_c^Q}\sum_{i=1}^{n_c^Q}|y^Q_{c,i} - \hat{y^Q_{c,i}}|. \end{aligned} \quad (11)$$

We calculate the mean of RMSE scores and MAE scores among all target tasks to evaluate the performance of all the methods.

4.2 Baseline Methods

We evaluate SSML against the following strong baselines.

- ConSTGAT. The previous deployed model ConSTGAT [4] applies a spatial and temporal graph neural network to capture the mutual relations between the spatial and temporal information for TTE. We use ConSTGAT to estimate the travel times of the remaining route dynamically.
- Fine-tuning (FT). We fine-tune the model parameters of ConSTGAT on each circumstance with the support examples to estimate the travel times of the query examples. The learning rate for FT is set to 0.01.
- Recurrent-Based Meta-Learning (R-ML). Similar to previous model-based meta-learning methods [11, 16], R-ML uses bidirectional long short-term memory (LSTM) [6] to capture the meta-knowledge. R-ML takes the same learned representations as SSML does.

Table 2: RMSE scores of SSML and other methods on three real-world datasets with different request times.

Method	Taiyuan			Huizhou			Hefei		
	t = 5 min	t = 10 min	t = 15 min	t = 5 min	t = 10 min	t = 15 min	t = 5 min	t = 10 min	t = 15 min
ConSTGAT	33.425	35.153	37.460	44.558	45.202	46.537	45.698	44.992	46.001
FT	**33.408**	35.084	37.376	**44.523**	44.945	46.318	45.656	44.775	45.701
R-ML	33.669	34.838	36.525	45.014	44.275	44.338	46.106	43.053	42.548
A-ML	33.521	30.724	26.537	45.022	36.412	28.871	45.723	32.374	25.406
SSML	33.502	**30.154**	**26.038**	44.594	**35.222**	**28.470**	**45.471**	**32.228**	**24.725**

Table 3: MAE scores of SSML and other methods on three real-world datasets with different request times.

Method	Taiyuan			Huizhou			Hefei		
	t = 5 min	t = 10 min	t = 15 min	t = 5 min	t = 10 min	t = 15 min	t = 5 min	t = 10 min	t = 15 min
ConSTGAT	18.609	21.104	24.046	28.179	30.605	33.050	32.822	33.722	34.917
FT	18.594	21.049	23.979	28.142	30.374	32.852	32.768	33.519	34.640
R-ML	18.706	21.186	23.353	29.275	30.062	30.616	32.694	31.225	31.246
A-ML	**18.200**	17.636	16.084	28.628	23.729	18.853	31.782	**22.429**	17.601
SSML	18.338	**17.607**	**15.798**	**27.751**	**22.915**	**18.773**	**31.672**	22.669	**17.329**

- Attention-Based Meta-Learning (A-ML). The framework of A-ML is almost the same as that of SSML, except that A-ML only applies supervised learning without using self-supervised learning. A-ML is an ablation version of SSML.

For all the methods, including SSML, we set the embedding sizes of all the attribute to 8, and set the hidden sizes of attention and LSTM to 32. For SSML, we randomly mask 30% of the labels of the support examples for each circumstance. We implemented all the methods by *PaddlePaddle*[1], an open-source deep-learning platform maintained by Baidu. We apply stochastic gradient descent (SGD) to learn all the methods, and the learning rate is set to 0.1.

4.3 Offline Tests

To evaluate the effectiveness of the proposed SSML, we conducted offline tests to compare the performance of SSML and other methods on three real-world datasets, Taiyuan, Huizhou, and Hefei, with different request times. Table 2 and Table 3 show the RMSE scores and MAE scores of different methods on three cities with different request times. From the results, we make the following observations.

(1) We can see that ConSTGAT performs the worst among all methods on all cities, as it does not consider the observed behaviors in the traveled route. This confirms the importance of taking the traveled route into consideration to improve the accuracy of time estimation.

(2) FT works better than ConSTGAT because it leverages the observed behaviors by fine-tuning, which further shows the importance of considering the traveled route.

(3) The model-based meta-learners, i.e., R-ML, A-ML, and SSML, perform much better than ConSTGAT and FT due to the learned meta-knowledge, which enables them to fast adapt to new circumstances. A-ML and SSML outperforms R-ML in terms of t = 10 min and t = 15 min. Compared with R-ML that uses recurrent network, both A-ML and SSML use attention mechanism to capture the mutual relations of support examples and query examples. This demonstrates that the query example can learn from similar support examples directly and effectively by the attention mechanism.

(4) SSML significantly outperforms other methods in terms of t = 10 min and t = 15 min. This shows the effectiveness of combining self-supervised learning with meta-learning, which enables it to better learn the meta-knowledge.

(5) The results of different methods on three cities in terms of t = 5 min are similar. A possible reason is that the observed behaviors in a circumstance in terms of t = 5 min only contain short-distance information, making it difficult to make accurate long-distance predictions.

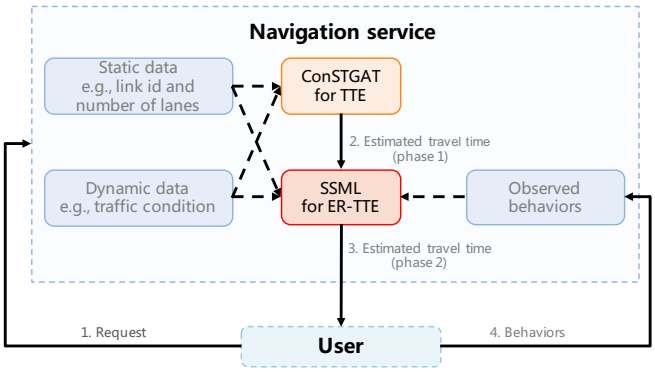
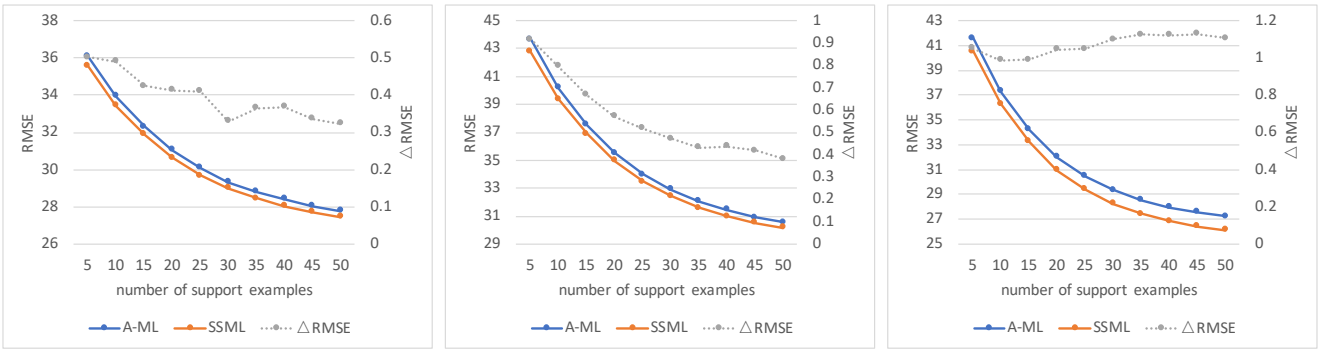


Figure 6: Navigation system for ER-TTE by applying SSML to ER-TTE at Baidu Maps.

4.4 Practical Applicability

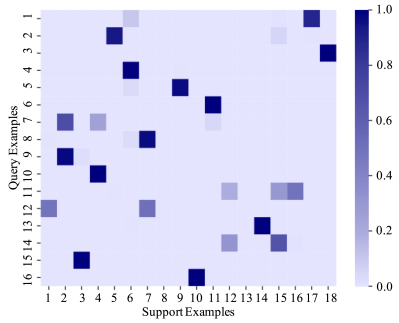
4.4.1 Navigation System. Before deploying SSML in production, we design a system that adds the SSML module to the navigation

[1]The official site is at https://www.paddlepaddle.org.cn/.



(a) Taiyuan (b) Huizhou (c) Hefei

Figure 7: The performance of A-ML and SSML with different numbers of support examples in terms of $t = 15$ min on different datasets. The RMSE scores of A-ML and SSML refers to the primary ordinate. The $\Delta RMSE = \text{RMSE}_{\text{A-ML}} - \text{RMSE}_{\text{SSML}}$ refers to the secondary ordinate.



(a) Attention weight matrix

Support examples

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
speed limit	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5
number of lanes	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3
start cross	2	4	4	1	4	2	2	2	2	4	2	2	2	2	2	2	2	1
end cross	2	1	1	4	2	4	4	2	4	2	1	2	2	4	2	2	2	2
ground-truth label	-23	29	-38	-1	-32	0	39	120	-42	-47	-2	2	-30	-4	-35	-1	-21	-30

Query examples

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
speed limit	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4
number of lanes	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2
start cross	2	4	1	2	2	2	1	2	4	1	2	2	2	2	4	4
end cross	2	2	2	4	4	1	1	4	1	4	2	2	4	2	1	2
predicted label	-16	-26	-28	0	-42	5	5	39	35	-5	-4	-2	0	3	-28	-41

(b) Features of support and query examples

Figure 8: Case analysis. When the predicted label of a query example is positive, the ground-truth labels of similar support examples are usually positive, and vice versa.

service at Baidu Maps to validate the practicability of SSML. The illustration of the navigation system for ER-TTE is shown by Figure 6. When the navigation service receives a request, the previously deployed model ConSTGAT takes the static features and dynamic features as input and estimates the travel time. Then, SSML considers the predicted travel time by ConSTGAT, static features, dynamic features, and the observed behaviors in the traveled route collected from a user to estimate the travel time of the remaining route. Finally, the navigation service sends the predicted travel time to the user to update the remaining travel time. This process repeats every few minutes until the user finishes her trip.

4.4.2 Fast Adaption. We compare the fast adaption abilities of A-ML and SSML with only a few support examples in the navigation system for ER-TTE. We compare the RMSE scores of A-ML and SSML with different sizes of support sets with request time $t = 15$ min. For each circumstance, we keep the query set unchanged and randomly select a subset of the support set to ensure that the size of support set does not exceed a preset number. Figure 7 shows the RMSE scores of A-ML and SSML with different sizes of support sets. It also shows the gap between the RMSE scores of A-ML and SSML, i.e., $\Delta\text{RMSE} = \text{RMSE}_{\text{A-ML}} - \text{RMSE}_{\text{SSML}}$. We can see that ΔRMSE is larger when the size of support sets is smaller on Taiyuan and Huizhou. It demonstrates that SSML is able to learn the meta-knowledge better with only a few support examples.

4.4.3 Case Analysis. To analyze the knowledge learned by SSML in the navigation system, we present the attention weight matrix, some of the features, and labels of a circumstance, as shown by Figure 8. The attention matrix in Figure 8(a) describes the associations between the support examples and query examples. Figure 8(b) shows the features of the support and query examples, the ground-truth labels of the support examples, and the predicted labels of the query examples. All the features shown are discretized, including speed limit, the number of lanes, start cross, and end cross. For most cases, when the predicted label of a query example is positive, the ground-truth labels of similar support examples are positive, and vice versa. This shows that SSML is able to estimate the label

of a query example by capturing the information from the support examples with similar features.

5 RELATED WORK

Here we briefly review the closely related work in the fields of travel time estimation and meta-learning.

5.1 Travel Time Estimation

TTE is an important module for mapping services. In recent years, a lot of studies [1, 4, 10, 18–21, 24] successfully apply deep neural networks on travel time estimation. Several studies [18, 21, 24] have considered a user's driving preference by extracting features from the driver's profile, such as driver id and driver type. However, they mainly focus on the task of TTE rather than ER-TTE. Besides, they have not considered the user's observed behaviors in the traveled route. By contrast, we study the problem of ER-TTE by taking the traveled route into consideration, which has been shown to be effective in improving the accuracy of time estimation.

5.2 Meta-Learning

Meta-learning [7] has made significant progress recently, which is able to learn models that rapidly adapt to new environments. Optimization-based [5, 14] and model-based [11, 12, 16] meta-learning methods have drawn lots of attention. Optimization-based methods adjust the optimization algorithm so that the learned model can fast adapt to new tasks with a few examples. Model-based methods design models for fast adaption by updating their parameters.

Meta-learning has been applied to intelligent transportation [15, 22, 23] and recommender systems [3, 9, 25]. For intelligent transportation, Zang et al. [23] apply value-based meta-reinforcement learning to traffic signal control. Pan et al. [15] leverage the meta knowledge extracted from geo-graph attributes to generate the network's parameter weights for urban traffic prediction. Yao et al. [22] transfer the spatial-temporal information from the source cities to the target cities. For recommender systems, the meta-learning methods are mainly used to solve the cold-start problem. Lee et al. [9] take each user in a recommender system as a task and apply MAML [5] to adapt the cold-start users' preferences.

Although the idea of applying meta-learning to learn a user's preference with insufficient user behaviors is conceptually similar, our model is novel in that it successfully combines meta-learning with self-supervised learning for data argumentation, which enables the navigation service to provide more accurate time estimations as quickly as possible while observing the user behaviors en route for only a few minutes.

6 CONCLUSION

ER-TTE is a particular case of TTE, which dynamically estimates the travel times of the remaining routes. We take the traveled route into consideration, which can improve the accuracy of the time estimation. ER-TTE is challenging due to the lack of sufficient observed driving behaviors. To this end, we frame ER-TTE as a few-shot learning problem and apply a novel model-based meta-learning approach to adapt fast to the user's driving preference. We combine supervised learning and self-supervised learning to enhance the ability of learning meta-knowledge. Extensive experiments conducted on large-scale real-world datasets collected from Baidu Maps demonstrate the superiority of SSML.

REFERENCES

[1] Pouria Amirian, Anahid Basiri, and Jeremy Morley. 2016. Predictive Analytics for Enhancing Travel Time Estimation in Navigation Apps of Apple, Google, and Microsoft. In *SIGSPATIAL*. 31–36.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[3] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In *KDD*. 2895–2904.

[4] Xiaomin Fang, Jizhou Huang, Fan Wang, Lingke Zeng, Haijin Liang, and Haifeng Wang. 2020. ConSTGAT: Contextual Spatial-Temporal Graph Attention Network for Travel Time Estimation at Baidu Maps. In *KDD*. 2697–2705.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint:1703.03400* (2017).

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[7] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. *arXiv preprint:2004.05439* (2020).

[8] Peter J. Huber. 1964. Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics* 35, 1 (1964), 73–101.

[9] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *KDD*. 1073–1082.

[10] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *KDD*. 1695–1704.

[11] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. A simple neural attentive meta-learner. *arXiv preprint:1707.03141* (2017).

[12] Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. *Proceedings of machine learning research* 70 (2017), 2554.

[13] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.

[14] Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint:1803.02999* 2, 3 (2018), 4.

[15] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *KDD*. 1720–1730.

[16] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *ICML*. 1842–1850.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[18] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *AAAI*. 2500–2507.

[19] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A Simple Baseline for Travel Time Estimation Using Large-scale Trip Data. *TIST* 10, 2, Article 19 (2019), 22 pages.

[20] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *KDD*. 25–34.

[21] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to estimate the travel time. In *KDD*. 858–866.

[22] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *WWW*. 2181–2191.

[23] Xinshi Zang, Huaxiu Yao, Guanjie Zheng, Nan Xu, Kai Xu, and Zhenhui Li. 2020. MetaLight: Value-Based Meta-Reinforcement Learning for Traffic Signal Control. In *AAAI*. 1153–1160.

[24] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. 2018. DeepTravel: a Neural Network Based Travel Time Estimation Model with Auxiliary Supervision. In *IJCAI*. 3655–3661.

[25] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to Retrain Recommender System? A Sequential Meta-Learning Method. In *SIGIR*. 1479–1488.