Service Time Prediction for Delivery Tasks via Spatial Meta-Learning

Sijie Ruan
sijieruan@outlook.com
Beijing Institute of Technology, China
Xidian University, China

Cheng Long
c.long@ntu.edu.sg
Nanyang Technological University, Singapore

Zhipeng Ma
mazhipeng5@jd.com
Southwest Jiaotong University, China
JD iCity, JD Technology, China

Jie Bao
baojie@jd.com
JD iCity, JD Technology, China
JD Intelligent Cities Research, China

Tianfu He
tianfudhe@foxmail.com
JD iCity, JD Technology, China
JD Intelligent Cities Research, China

Ruiyuan Li
liruiyuan@cqu.edu.cn
Chongqing University, China

Yiheng Chen
chenyiheng@jd.com
JD Logistics, China

Shengnan Wu
wushengnan1@jd.com
JD Logistics, China

Yu Zheng
msyuzheng@outlook.com
Xidian University, China
JD iCity, JD Technology, China

ABSTRACT

Service time is a part of time cost in the last-mile delivery, which is the time spent on delivering parcels at a certain location. Predicting the service time is fundamental for many downstream logistics applications, e.g., route planning with time windows, courier workload balancing and delivery time prediction. Nevertheless, it is non-trivial given the complex delivery circumstances, location heterogeneity, and skewed observations in space. The existing solution trains a supervised model based on aggregated features extracted from parcels to deliver, which cannot handle above challenges well. In this paper, we propose MetaSTP, a meta-learning based neural network model to predict the service time. MetaSTP treats the service time prediction at each location as a learning task, leverages a Transformer-based representation layer to encode the complex delivery circumstances, and devises a model-based meta-learning method enhanced by location prior knowledge to reserve the uniqueness of each location and handle the imbalanced distribution issue. Experiments show MetaSTP outperforms baselines by at least 9.5% and 7.6% on two real-world datasets. Finally, an intelligent waybill assignment system based on MetaSTP is deployed and used internally in JD Logistics.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**.

KEYWORDS

delivery data mining; meta-learning; urban computing

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539027

ACM Reference Format:
Sijie Ruan, Cheng Long, Zhipeng Ma, Jie Bao, Tianfu He, Ruiyuan Li, Yiheng Chen, Shengnan Wu, and Yu Zheng. 2022. Service Time Prediction for Delivery Tasks via Spatial Meta-Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539027

1 INTRODUCTION

The last-mile delivery in logistics mainly relies on couriers. Typically, once a batch of parcels arrives at a delivery station, it would be assigned to couriers. Then, couriers would start a delivery trip to deliver the assigned parcels at several locations, as shown in Figure 1(a). Two types of time cost for the entire trip are involved in the delivery trip: *travel time* and *service time*. The former is the time cost traveling between locations, and the later is the time cost completing the delivery for a set of parcels at a certain location, namely, a *delivery task*. Estimating those two types of time cost facilitates many downstream applications, e.g., the route planning with time windows [5], the workload balancing and the delivery time prediction [33] as demonstrated in Figure 1(b). While the travel

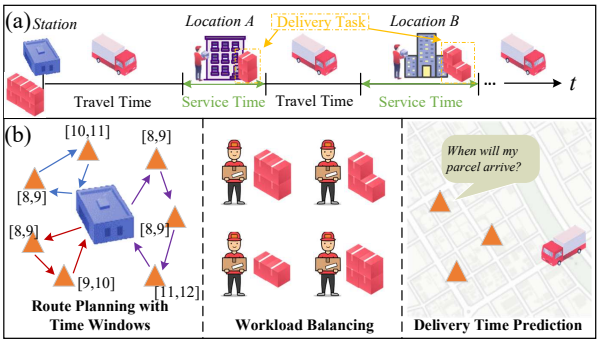
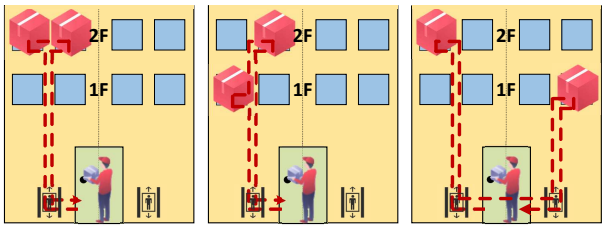


Figure 1: Service Time and Its Applications.



(a) Same Unit Same Floor. (b) Same Unit Diff. Floors. (c) Different Units.

Figure 2: Complex Delivery Circumstances.

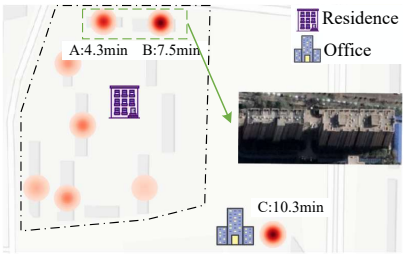


Figure 3: Location Heterogeneity & Skewed Observations.

time prediction has been widely studied [10, 30, 35], there are limited research on the service time prediction so far. Therefore, in this work, we mainly focus on the service time prediction.

Traditionally, the service time is estimated by assuming it to be proportional to the number of customers to deliver parcels to. However, this method can introduce great prediction errors due to its subjective nature. [24] made some initial efforts to predict the service time in a data-driven way, which uses some coarse aggregated features from delivery tasks to train a k-Nearest Neighbor regressor [1]. Nevertheless, it still fails to tackle the following challenges of the service time prediction problem.

- **Complex delivery circumstances.** Figure 2 shows a building with two units under three delivery circumstances, which have the same number of customers to be delivered to, i.e., two customers. However, their delivery processes are totally different. Though in Figure 2(a) and 2(b), the courier delivers only for one unit, he only needs to visit a floor in the former case, while he has to visit two floors in the latter case. And in Figure 2(c), the courier has to deliver for each unit one by one. Different delivery processes lead to different service time, leaving alone there could be more customers to deliver for in a task in the real world.
- **Location heterogeneity.** The service time could also vary greatly for deliveries at different locations. We report the average service time for delivering for one customer (the simplest delivery circumstance) at three different locations (A, B and C) in Figure 3. As observed, the average service time at location C (an office building) takes much longer time than that at location A and B (residential locations). Furthermore, even for locations at the same residential area, the service time can vary (as shown at location A and B), which might be attributed to different building structures at the locations (as shown by the satellite image).
- **Skewed observations.** The frequency of placing orders in the e-commerce platform follows a long-tailed distribution in space, which makes the historical delivery tasks, i.e., data samples for training, distributed highly imbalanced among locations, as shown by the heat map in Figure 3. Training a single model for all locations would favor those locations with more observations and work poorly for those with few observations. Training a model for each location is also infeasible since each location requires adequate data to model complex delivery circumstances, while most locations only have limited observations.

To this end, we present MetaSTP, a Meta-learning-based method to make the Service Time Prediction. To tackle the location heterogeneity, MetaSTP treats the service time prediction at different locations as different *learning tasks*. When predicting the service time, MetaSTP first leverages a Transformer encoder-based representation module to obtain floor-distribution-aware delivery task embedding, which captures the complex delivery circumstances of each delivery task, then employs a location prior knowledge enhanced meta-learning method to produce accurate predictions based on location-specific historical observations, globally learned delivery knowledge and location prior knowledge, to mitigate the skewed observation issue. Our contributions are three folds:

- We identify major challenges of service time prediction, which is overlooked by the literature, and propose MetaSTP, a location prior knowledge enhanced meta-learning method with a carefully designed representation module to tackle those challenges.
- Extensive experiments based on two real delivery datasets from JD Logistics demonstrate the effectiveness of MetaSTP, which outperforms baselines by at least 9.5% and 7.6%, respectively.
- We present an intelligent waybill assignment system based on MetaSTP, which has been deployed and used internally in JD Logistics.

2 PRELIMINARIES

2.1 Problem Formulation

Definition 1 (Waybill). A waybill contains the information about the parcel to deliver, denoted as a 4-tuple $w = (addr, uid, \mathbf{F}, ts)$. $addr$ is the address, uid is the customer ID, and $\mathbf{F}$ denotes the features of the corresponding parcel, e.g., the weight and the volume. ts is a planned delivery time slot of the parcel (e.g., pending to be delivered during 8AM-11AM), which depends on how many delivery trips would be conducted in a day.

Definition 2 (Delivery Task). The set of parcels to be delivered together to the same delivery location is named as a delivery task. Parcels in a delivery task share the same planned delivery time slot, therefore, we explicitly denote a delivery task as $t = (W, ts)$, where W is the set of waybills of those parcels.

Note that, the delivery location of each waybill, is a geospatial coordinate in the urban space, which can be obtained based on its shipping address via Geocoding [1], or couriers' annotation [22, 23].

Problem Definition. Given historical delivery tasks and their corresponding service time, the service time prediction (STP) problem is to predict the service time for a delivery task in the future.

[1] https://en.wikipedia.org/wiki/Geocoding

2.2 Basic Concepts of Meta-Learning

Meta-learning [16] aims to extract the meta-knowledge that is globally shared among a set of related *learning tasks*, so that we can obtain ideal prediction for similar learning tasks based on learning task specific training set and the meta-knowledge even if the observations are quite limited.

In the terminology of meta-learning, the training set and validation/test set of each individual learning task $\mathcal{T}$ are usually named as support set $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$ and query set $\mathcal{D}^q = \{(\boldsymbol{x}_i^q, y_i^q)\}_{i=1}^{N_q}$. The inference of each query sample $\boldsymbol{x}^q$ can be formulated as $\hat{y}^q = f(\boldsymbol{x}^q, \mathcal{D}^s, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the meta-knowledge that is globally shared, and f is an arbitrary form of method that leverages $\boldsymbol{x}^q$,$\mathcal{D}^s$, and $\boldsymbol{\theta}$ to obtain the prediction $\hat{y}^q$.

Depending on the form of f, meta-learning can be categorized into three classes: optimization-based methods [11], metric-based methods [29], and model-based methods [20]. Currently, the metric-based meta-learning is usually limited to the classification problem. Given competitive performance of the optimization-based methods and the model-based methods, the latter are simpler and easier to be optimized [16]. Therefore, in this study, we mainly follow the paradigm of the model-based meta-learning. In model-based methods, f itself is a neural network parameterized by $\boldsymbol{\theta}$. It hides the internal procedure to obtain the prediction by a black-box. In such case, the inference can also be written as:

$$\hat{y}^q = f_{\boldsymbol{\theta}}(\boldsymbol{x}^q, \mathcal{D}^s) \tag{1}$$

, which receives $\mathcal{D}^s$ and $\boldsymbol{x}^q$, and produces $\hat{y}^q$ end to end.

To optimize $\boldsymbol{\theta}$, we usually assume access to a set of learning tasks already sampled from a learning task distribution $p(\mathcal{T})$, which is called meta-training tasks $\mathcal{T}_{meta-train}$ [2]. The optimal $\boldsymbol{\theta}$ learned from $\mathcal{T}_{meta-train}$ should adapt well to any learning task sampled from $p(\mathcal{T})$ based on Equation 1, which is achieved by optimizing the following meta loss function:

$$\mathscr{L}(\boldsymbol{\theta}) = \sum_{\mathcal{T}_i \in \mathcal{T}_{meta-train}} \frac{1}{|\mathcal{D}_i^q|} \sum_{(\boldsymbol{x}^q, y^q) \in \mathcal{D}_i^q} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}^q, \mathcal{D}_i^s), y^q) \tag{2}$$

where $\mathcal{D}_i^s$ and $\mathcal{D}_i^q$ are the support set and the query set of learning task $\mathcal{T}_i$, and $\mathcal{L}$ is the loss function of a learning task.

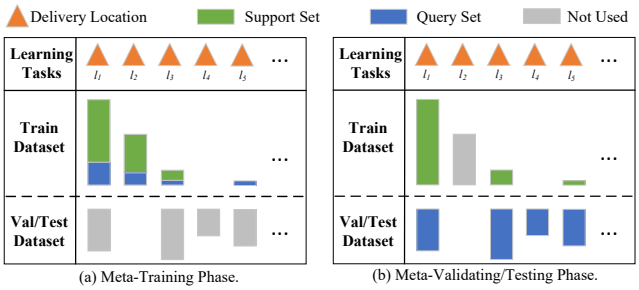
3 METHODOLOGY

In this section, we first introduce how we setup the learning tasks for meta-learning, then give an overview of our model, i.e., MetaSTP, and finally we present each module of the model in detail.

3.1 Learning Task Setup

In the real world, we cannot access future delivery tasks when training our model. Therefore, we leverage fixed timestamps to split historical delivery tasks into training dataset, validation datasets and testing dataset (same as traditional machine learning settings).

During the training phase (meta-training), we need to construct the meta-training dataset $\mathcal{T}_{meta-train}$ based on the training dataset. Given that the service time prediction at each delivery location is treated as an individual learning task, we first group training datasets based on delivery locations. As shown in Figure 4(a), each bar above the dash line indicates the number of observations can be utilized for each learning task during the meta-training. Recall in Section 2.2, for each learning task, both support set and query set are required. Therefore, for the observations can be used in each delivery location, we always keep $r\%$ proportion as the query set (in blue) and leave others as the support set (in green). During the inference phase (meta-validating/meta-testing), for each location, the whole training dataset of each location is treated as the support set, and the whole validation/test dataset is treated as the query set, as shown in Figure 4(b).



(a) Meta-Training Phase. (b) Meta-Validating/Testing Phase.

Figure 4: Illustration of Learning Task Setup.

Note that, unlike typical meta-learning settings [11, 20] where for each learning task, the support set is always available. In our case, it is possible that some locations have no observation in the training dataset, because the location has no delivery record during the time interval of the training dataset. In this case, the support set would be empty when we perform the inference, as l_4 shown in Figure 4(b). Given this fact, we also include those cases (learning tasks with empty support set) in the meta-training, e.g., l_5 in Figure 4(a). It is achieved by specifying a minimum query set threshold N_q^{min}. For a location in the meta-training, if the size of query portion is less than N_q^{min}, all samples would be assigned to the query set during the meta-training, which is formally given as follows:

$$N_q = \begin{cases} \lceil r\% * |\mathcal{D}| \rceil, & \text{if } \lceil r\% * |\mathcal{D}| \rceil \geq N_q^{min} \\ |\mathcal{D}|, & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{D}$ is the whole training dataset of a location.

3.2 Model Overview

Figure 5 depicts the framework of MetaSTP, which consists of three modules to predict the service time for each delivery task:

- **Delivery Task Representation**, which first extracts and embeds floor-level waybill features of each delivery task, then combines the embeddings with other delivery task features to obtain the fine-grained hidden representation of each delivery task;
- **Historical Observation Encoding**, which generates a vector that encodes the correlation between the hidden representation of the query task and tasks with labels in the support set;
- **Location-wise Knowledge Fusion**, which further enhances the output vector with the location prior knowledge so that an ideal prediction can still be achieved even if the support set has no or very limited observations.

Next, we elaborate each module in detail.

[2] Correspondingly, we also have meta-validation tasks $\mathcal{T}_{meta-val}$ and meta-test tasks $\mathcal{T}_{meta-test}$ for hyperparameter tuning and testing.

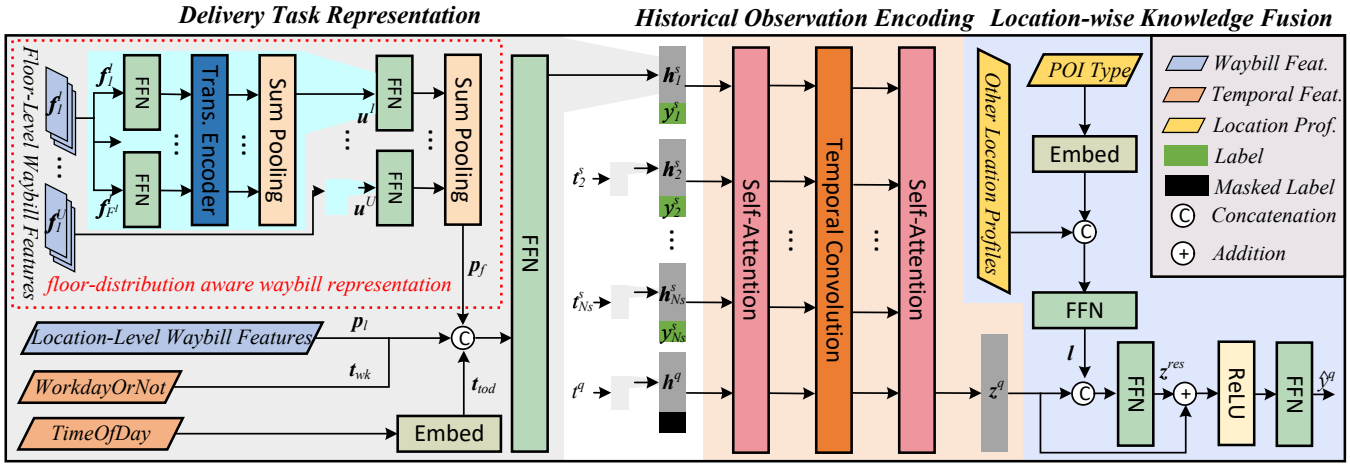


Figure 5: The Architecture of MetaSTP.

3.3 Delivery Task Representation

The delivery task representation module aims to encode each delivery task into an expressive representation to facilitate later observation correlation calculation and prediction.

Main Idea. To encode the complex delivery process, we propose the delivery task representation layer (as shown in the left part of Figure 5) to capture fine-grained waybill information. The module first groups waybills in the task by units and floors, and extracts waybill features for each floor involved in the delivery. Then the set of floor-level features are jointly considered to obtain a floor distribution-aware waybill representation for each unit. Those representations from all units are further fused to obtain the floor distribution-aware waybill representation of the delivery task. At last, it is combined with location-level waybill features as well as temporal features to further enrich the representation.

The floor distribution-aware waybill representation is inspired by three key insights from a delivery task:

(1) The transition time cost at the same floor is much smaller than that among different floors (which involve waiting for elevators and/or walking upstairs) as shown in Figure 2(a).
(2) Waybills distributed among all floors in a unit jointly determine its service time, as observed in Figure 2(b).
(3) Deliveries are usually conducted unit by unit, and the service times of all units contribute to the overall time cost, but they are less affected by each other as illustrated in Figure 2(c).

Implementation. To implement the above idea, we first identify the unit and floor information from the address of each waybill based on regular expression, since those information usually follows some fixed patterns (e.g., `Floor XX, Unit X, Building X, ...`). Based on the unit and floor information of each waybill, waybills in the delivery tasks are grouped.

Then, for each floor in each unit involved in the delivery task, we extract the following 5 floor-level features which potentially contribute to the service time by aggregating waybills at the floor: (1) the floor number; (2) the number of customers to deliver parcels to; (3) the total number of waybills; (4) the total weight of parcels; and (5) the total volume of parcels. Formally, we use $\mathbf{f}_i^j$ to denote floor-level features of the i^{th} floor of the j^{th} unit in the delivery task. Then, all floor-level features of a delivery task can be represented as a set $\{\mathbf{F}^j\}_{j=1}^{U}$, where $\mathbf{F}^j = \{\mathbf{f}_i^j\}_{i=1}^{F^j}$, F^j is the total number of floors in the j^{th} unit, and U is the total number of units in the task. Here, we consider only those units and floors that are involved in the delivery task. Note that U and $F^j (j \in 1, ..., U)$ would vary from task to task, which are obtained based on units and floors in waybills of a deliver task. For example, in Figure 2(b), the floor-level features of the task is $\{\{\mathbf{f}_1^1, \mathbf{f}_2^1\}\}$, and in Figure 2(c), the floor-level features of the task is $\{\{\mathbf{f}_1^1\}, \{\mathbf{f}_1^2\}\}$.

Next, we extract floor distribution-aware waybill representation $\mathbf{p}_f$ for the task based on $\{\mathbf{F}^j\}_{j=1}^{U}$, which is a two-stage fusion (floor-stage and unit-stage). In the floor stage fusion, the correlation between floors in the same unit are captured, which is similar to many NLP tasks that accept a sentence with varying number of words [6]. RNN and Transformer [28] are commonly used module for the sequence correlation modeling. Here we adopt transformer encoder [28], which shows superior performance on many NLP tasks [6]. More specifically, we first apply a feedforward network on floor-level features in $\mathbf{F}^j$ to increase the dimension of each feature vector, and then apply a Transformer encoder to capture their correlation. The outputs of the transformer encoder are sent to a floor-level sum pooling to obtain the floor distribution-aware waybill representation of each unit, denoted as $\mathbf{u}^j$:

$$\mathbf{u}^j = \texttt{SumPool}(\texttt{TransEnc}(\texttt{FFN}(\mathbf{F}^j))) \tag{4}$$

where `FFN` contains a fully connected (FC) layer.

Since the service time of each unit together forms the total service time of the task, to obtain $\mathbf{p}_f$, we leverages a feedforward network to transform $\{\mathbf{u}^j\}_{j=1}^{U}$ into a hidden space, then the unit-level sum pooling is applied:

$$\mathbf{p}_f = \texttt{SumPool}(\texttt{FFN}(\{\mathbf{u}^j\}_{j=1}^{U})) \tag{5}$$

where `FFN` contains one FC layer and a ReLU activation.

Together with $\mathbf{p}_f$, we extract 6 location-level waybill features, which describe waybills in the task in a macro view: (1) the total

number of customers to deliver, (2) the total number of waybills, (3) the total weight and (4) volume of parcels, (5) the number of units to deliver, and (6) the total number of floors involved in the addresses of waybills. The obtained feature vector is denoted as $\mathbf{p}_l$.

Since the service time could also be affected by the delivery time slot considering the degree of crowdedness in the common pathway of the building, we further extract two types of temporal features from the planned delivery time slot of the task. The first one is a binary value $\mathbf{t}_{wk}$ indicating whether the time slot is on workdays or weekend. The second one is the time of the day. We discretize the working hours of couriers [8:00-23:00] into 5 bins with equal width. The time of the day feature is the bin index of which the planned delivery time slot fell in. It is fed into an embedding layer to obtain the dense representation $\mathbf{t}_{tod}$.

We concatenate floor distribution-aware waybill representation $\mathbf{p}_f$, location-level waybill features $\mathbf{p}_l$, workday or not feature $\mathbf{t}_{wk}$, and embedded time of day feature $\mathbf{t}_{tod}$, and send them to a feedforward network to obtain the delivery task representation $\mathbf{h}$.

$$\mathbf{h} = \texttt{FFN}([\mathbf{p}_f; \mathbf{p}_l; \mathbf{t}_{wk}; \mathbf{t}_{tod}]) \tag{6}$$

where FFN contains 2 FC layers with ReLU activation, and ; means concatenation.

The representation from the support set $\{\mathbf{h}_i^s\}_{i=1}^{N_s}$ and that of the query delivery task $\mathbf{h}^q$ are all sent to the next module.

3.4 Historical Observation Encoding

After the previous delivery task representation, all delivery tasks in the support set $\{t_i^s\}_{i=1}^{N_s}$ are transformed into dense representations $\{\mathbf{h}_i^s\}_{i=1}^{N_s}$, and the query task t^q is transformed into $\mathbf{h}^q$. For each $\mathbf{h}_i^s$, we concatenate it with its label y_i^s, and obtain a dense representation $\mathbf{o}^s$ of the support observation, i.e., $\mathbf{o}_i^s = [\mathbf{h}_i^s; y_i^s]$. The dense representation of the support set is denoted as $\mathcal{D}^{s'} = \{\mathbf{o}_i^s\}_{i=1}^{N_s}$.

Taking $\mathcal{D}^{s'}$ and $\mathbf{h}^q$ as inputs, the historical observation encoding aims to learn the correlation among them, and generate an embedding vector $\mathbf{z}_q$ (as shown in the middle part of Figure 5). $\mathbf{z}_q$ semantically is a high-dimensional representation of the prediction after "seeing" support set at the location.

We borrow the idea from a simple yet effective model-based meta-learning approach [20], which proposes to interleave self-attention [28] with temporal convolution [27] to encode past experiences. It enjoys the benefit of accepting infinite large past experiences (from self-attention) and having a high-bandwidth to direct access a batch of past experiences (from temporal convolution).

To implement, $\mathcal{D}^{s'}$ and $\mathbf{h}^q$ are fed into this module in a batch manner. We create a dimension expanded vector $\mathbf{o}^q$ by concatenating $\mathbf{h}^q$ with a masked label m, e.g., $m = 0$, to make sure the dimension consistency. After that, $\{\mathbf{o}_1^s, \mathbf{o}_2^s, ..., \mathbf{o}_{N_s}^s, \mathbf{o}^q\}$ are sent into a self-attention layer, followed by a temporal convolution layer. Then, another self-attention layer is applied to make sure the past experiences are fully utilized. At that point, those independent observation representations are transformed into experience-shared representations $\{\mathbf{z}_1^s, \mathbf{z}_2^s, ..., \mathbf{z}_{N_s}^s, \mathbf{z}^q\}$. At last, we take $\mathbf{z}^q$ as the output of the historical observation encoding module, which contains the knowledge about how to make the service time prediction for the query delivery task with historical observations encoded.

3.5 Location-wise Knowledge Fusion

The location-wise knowledge fusion module takes historical observation encoded representation $\mathbf{z}^q$ from the previous module, enhances it with location-wise prior knowledge (location profile), and gives the final service time prediction.

Main Idea. As we mentioned in Section 3.1, the support set of a learning task could be empty or has very limited observations. In those cases, the historical observation encoding module can hardly perceive the information brought by the specific location, which makes $\mathbf{z}^q$ have difficulty in capturing the uniqueness of its delivery location. However, even if the support set is empty, knowing where the location is still gives us some prior knowledge about it. For example, the locations with the same POI type is more likely to have similar building structures, thus their delivery situations may be similar. Therefore, it is beneficial to fuse such kinds of location-wise prior knowledge before the prediction (as shown in the right part of Figure 5).

Implementation. We consider four types of location prior knowledge to fuse: (1) region (row and column index of a $500m \times 500m$ cell in the gridded urban space), (2) POI type, (3) built year, and (4) second-hand house price per square meter. The last two features implicitly reflect whether the building is equipped with the elevator, which is usually not publicly available but also a very important factor to affect the service time. The region is directly derived from the delivery location of a certain learning task, while others are obtained from external data sources based on the location.

Based on those prior knowledge of the locations, we first send POI type into an embedding layer to obtain a embedded representation. It is then concatenated with other features and sent to a feedforward network with two FC layers activated by ReLUs to obtain the dense representation of the location prior knowledge, denoted by $\mathbf{l}$.

A straightforward approach is then to concatenate $\mathbf{l}$ with $\mathbf{z}^q$ to make the final prediction. However, this strategy performs badly according to our experimental results. We guess the reason is that the output of historical observation encoding already contains rich information for the time prediction, while $\mathbf{l}$ is a little bit noisy.

Inspired by ResNet [15], we propose to learn a residual information based on $\mathbf{z}^q$ and $\mathbf{l}$ that is able to refine $\mathbf{z}^q$, and ultimately make the prediction more accurate, which is formally defined as follows:

$$\hat{y}^q = \texttt{FFN}(\texttt{ReLU}(\mathbf{z}^q + \texttt{FFN}([\mathbf{z}^q; \mathbf{l}]))) \tag{7}$$

where FFN to calculate the residual contains 2 FC layers, and the first FC layer is followed by a ReLU activation following [15], and FFN to obtain the output contains one FC layer.

3.6 Optimization

To optimize MetaSTP, we need to choose the loss function $\mathcal{L}$ for each learning task (delivery location). Here, we employ MSE, which is widely used for regression problem:

$$\mathcal{L}(\hat{y}^q, y^q) = (y^q - \hat{y}^q)^2 \tag{8}$$

Then, MetaSTP is trained end to end by minimizing meta loss $\mathscr{L}$ (Equation 2) with the above learning task loss $\mathcal{L}$ (Equation 8). The overall training procedure is given in Algorithm 1.

Algorithm 1 MetaSTP Training Algorithm.

Input: Delivery task datasets $\mathcal{D}$; spatial external knowledge $\mathcal{E}$; MetaSTP model f_{θ}; query set rate r; minimum size of query set N_q^{min}.
Output: The optimized parameters θ of MetaSTP.

construct location task datasets $\mathscr{D}$ via grouping $\mathcal{D}$ by locations;
initialize θ by random;
repeat
  randomly select a batch of location task datasets $\mathscr{D}_b$ from $\mathscr{D}$;
  $\mathscr{L} \leftarrow 0$;
  for location delivery task dataset $\mathcal{D}_l \in \mathscr{D}_b$ **do**
    calculate N_q based on Equation 3 with $\mathcal{D}_l$, r and N_q^{min};
    randomly split $\mathcal{D}_l$ into $\mathcal{D}_l^s$, $\mathcal{D}_l^q$ with size $|\mathcal{D}_l| - N_q$ and N_q;
    $\mathscr{L} \leftarrow \mathscr{L} + \frac{1}{N_q} \sum_{(t_i^q, y_i^q) \in \mathcal{D}_l^q} \mathcal{L}(f_{\theta}(t_i^q, \mathcal{D}_l^s, \mathcal{E}_l), y_i^q)$;
  update θ by minimizing $\mathscr{L}$;
until *stopping criteria is met*
return θ;

4 EXPERIMENTS

4.1 Datasets

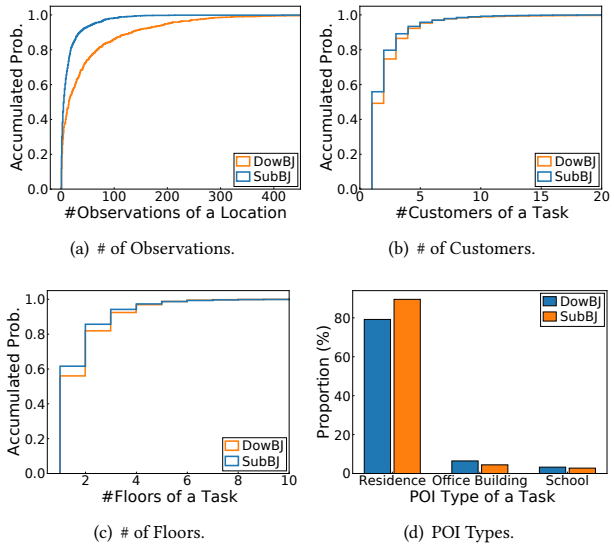
Our datasets consist of historical delivery tasks and spatial external knowledge, which are introduced as follows.

- **Historical Delivery Tasks.** We use two real world datasets from JD Logistics for evaluation, which are collected in the downtown area (DowBJ) and suburban area (SubBJ) of Beijing (splitted by the 3rd Ring) over a period of 20 months (from Jan. 1st, 2018 to Sept. 1st, 2019). The raw data consist of couriers' trajectories and waybills. We first use our previous work [22, 23] to infer the delivery location of each waybill, then group waybills in each delivery trip into delivery tasks. The pending delivery time slot of each task is set according to the start time of each delivery trip, since STP is usually called before departure. Finally, we match each delivery task to a stay point (detected from couriers' trajectories) according to the accurately annotated delivery time or the spatial closeness [23]. The duration of the stay point is treated as the service time of the corresponding task. After the previous data pre-processing steps, we obtain a database consisting of historical delivery tasks as well as their corresponding service times. We use the data from the first 16 months as training set, the data from the following 2 months as validation set, and use the last 2 months for testing. The details of each dataset are summarized in Table 1.

Table 1: Statistics of Datasets ("/" is train/val/test separator).

Datasets	DowBJ	SubBJ
#Delivery Tasks	53,979/5,978/6,138	22,403/4,630/5,520
#Delivery Locations	1,166/628/591	1,520/865/1,018
#New Loc. in Val/Test	13/16	266/424
Avg. Service Time (s)	418	395

- **Spatial External Knowledge.** For each delivery location, we obtain its POI type via reverse Geocoding [3], which contains 18 POI types, and the built year and the second-hand house price for Residence are crawled from the Web [4], while for other types of POI, we use the mean to fill missing values.



(a) # of Observations. (b) # of Customers. (c) # of Floors. (d) POI Types.

Figure 6: Data Distributions.

We provide the distribution of some important aspects of both datasets as follows.

Number of Observations Distribution. Figure 6(a) shows the distribution of the number of observations of a delivery location in both datasets. As can be seen, the observations are distributed highly skewed in the urban space. For 80% locations in DowBJ, there are less than 69 observations for training. The case is even worse in SubBJ, which only have less than 19 observations under the same criteria. It indicates that it is impossible for us to train a separate model for each individual location given limited observations.

Number of Customers Distribution. Figure 6(b) shows the distribution of the number of customers involved in a delivery task. As observed, both datasets have similar distribution, for around 40%-50% delivery tasks, couriers have to deliver parcels for more than one customer at a location, which introduces many uncertainties for the delivery time prediction.

Number of Floors & Units Distribution. Figure 6(c) shows the distribution of the number of floors of a delivery task in both datasets, which also show similar distribution. For around 40% delivery tasks, couriers need to go to different floors to complete the delivery task at those locations. In addition, there are 10% delivery tasks in both datasets, in which couriers need to go to multiple building units at a deliver location. Those complex factors further bring challenges to the modeling.

POI Types Distribution. Figure 6(d) shows the distribution of POI type of a delivery task in both datasets. The top 3 POI types of delivery tasks are the same in both datasets: Residence, Office Building, and School. The deliveries for Residence take up for around 80%. It can also be noticed that the portion of deliveries for Office Building is a bit higher in DowBJ than SubBJ, which is consistent with our common sense.

[3] https://lbs.qq.com/

[4] https://www.fang.com/

4.2 Experimental Settings

Baselines. We compare MetaSTP with the following baselines.

- HA, which always gives the historical average service time.
- HLA, which gives the location-specific historical average value. If a location is newly appeared, the global HA would be used.
- HCA, which assumes the time is proportional to the number of customers. The factor is estimated from the training data.
- GBRT [12], which trains a gradient boosting regression tree based on historical observations to make prediction.
- MLP [14], which trains a 3-layer MLP to make prediction.
- KNN [24], which is the state-of-the-art service time prediction model in literature. It trains a KNN regressor [1].

For all machine learning baselines, features from waybills only contain location-level ones, which are concatenated with others as input, since it is impractical to pad the floor-level waybill features to the maximum length given long-tailed involved floors in samples.

Variants. We also compare MetaSTP with following variants to show the effectiveness of each component of MetaSTP.

- MetaSTP-nMeta, which removes the meta-learning component. That is, we change the last layer of the delivery task representation to let it directly make the prediction. And the model is trained in the way like traditional machine learning methods.
- MetaSTP-nSeq, which removes the floor distribution-aware waybill representation $\mathbf{p}_f$ from MetaSTP.
- MetaSTP-nLP, which drops the entirely location-wise knowledge fusion module, and makes the output of historical observation encoding to directly predict the time via a FC layer.
- MetaSTP-nRes, which drops the residual connection, and directly concatenates $\mathbf{l}$ with the output from the historical observation encoding to make the prediction.

Evaluation Metrics. We leverage three commonly used metrics for regression problem, i.e., MAE, RMSE and MAPE, to evaluate the performance of different methods. MAE $= \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$, where N is the total number of delivery tasks in the test set. MAE characterizes the average prediction error with respect to the ground-truth over all test samples. RMSE $= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$, which is more sensitive to samples with large prediction errors. MAPE $= \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$, which measures the average relative errors of the prediction and the ground-truths.

Training Details & Hyperparameters. Our method as well as baselines are completely implemented in Python using PyTorch on a docker with 16 Cores@2.2GHz, 64GB memory and Red Hat Linux. In meta-training dataset setup, for each learning task, $r = 0.2$ and $N_q^{min} = 1$. We leverage Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to perform the meta-training with a learning rate $1e$-3. We sample one learning task in each iteration, and the whole meta-training dataset is iterated over 2 times. The hidden size of FC before the transformer, in the transformer encoder, in the representation output as well as in the self-attention are all set to 8. FC to obtain $\mathbf{u}$ contains 16 neurons. WorkdayOrNot is embedded to $\mathbb{R}^3$. The temporal convolution is stacked by 4 dilated 1D convolutions with 16 filters. POI Type is embedded to $\mathbb{R}^2$. To obtain $\mathbf{l}$, the first FC contains 4 neurons, while the second contains 2 neurons. To learn $\mathbf{z}_{res}$, the hidden size is 2. The best hyperparameters of baselines are also selected based on their performance on the validation set.

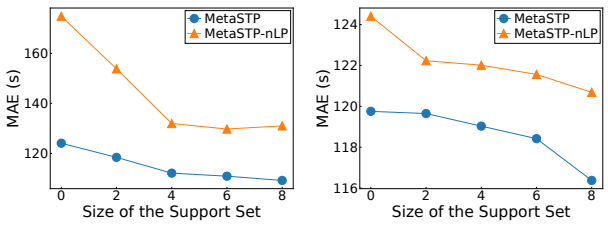
Table 2: Overall Evaluation.

Methods	DowBJ			SubBJ		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	253.9	368.1	126.0	229.6	320.3	112.0
HLA	204.2	295.0	88.2	205.8	282.9	93.8
HCA	174.0	269.3	68.7	157.7	235.5	64.4
MLP	161.4	257.7	65.9	150.0	259.5	61.5
GBRT	156.4	239.5	67.0	152.0	241.4	56.9
KNN	154.4	237.4	63.5	153.5	223.0	60.7
MetaSTP-nRes	151.7	244.3	58.9	145.6	374.8	59.9
MetaSTP-nMeta	149.8	232.5	59.4	145.8	246.2	56.3
MetaSTP-nSeq	144.3	228.0	56.7	144.5	283.2	51.3
MetaSTP-nLP	144.1	229.4	54.8	141.2	241.6	53.8
MetaSTP-nW	201.1	304.7	78.7	188.2	285.7	70.2
MetaSTP-nSE	142.6	227.5	52.3	140.7	221.1	50.8
MetaSTP-nT	142.8	230.6	53.7	142.9	267.0	51.0
MetaSTP (Ours)	**139.7**	**226.9**	**48.9**	**138.5**	**216.0**	**50.0**

4.3 Evaluation

Overall Performance. The overall performance of MetaSTP compared with baselines over all three metrics is shown in Table 2. As can be observed, directly predicting the service time based on historical average (HA) leads to huge errors, indicating that it is far from enough to empirically estimate the service time without considering the information of a certain delivery task. HLA is better than HA, which shows the uniqueness of different locations. HCA is better than HLA, which shows the number of customers surely is a shared strong signal affects the time among different locations. Traditional machine learning methods (MLP, GBRT, and KNN) show superior performance than aforementioned empirical ones, since they are able to model various factors by aggregating waybills in the delivery task. Nevertheless, the coarse-grained location-level features is not representative enough, and the universal model naturally is not able to fit locations with few observations. Those limitations leave us the room for improvements. Our method, i.e., MetaSTP, not only encodes the delivery task into a finer-grained level, but also leverages meta-learning and the prior knowledge to tackle the problem of skewed observations among locations. MetaSTP consistently outperforms baselines over three metrics on two datasets. Its MAE is 139.7s on DowBJ and 138.5s on SubBJ, which outperforms the best baseline by 9.5% and 7.6%, respectively.

Ablation Study. The ablation study is also conducted in Table 2 to validate the effectiveness of different components of MetaSTP. After removing the meta-learning strategy (MetaSTP-nMeta), a significant performance drop is witnessed, which shows the advantages of using meta-learning to tackle the skewed observation issues. If we ignore the encoding for floor-level waybill features (MetaSTP-nSeq), the performance degradation is also obvious, indicating that the floor distribution-aware waybill representation learning did provide finer-grained information for service time prediction. The prior knowledge about the location is also vital for the service time prediction. As can be observed, it further boosts the performance of the service time prediction (compared with MetaSTP-nLP). The last variant MetaSTP-nRes shows the necessity to use the residual structure to fuse the loation-wise knowledge with the previous output. The residual structure is helpful to learn how to fuse those two types of information to produce a more accurate prediction.



(a) Residence. (b) Office.

Figure 7: Different # of Support Observations.

Importance of Features. We are also interested in the importance of different types of features for the service time prediction problem. MetaSTP-nW, MetaSTP-nSE and MetaSTP-nT are MetaSTP with features from waybills, spatial external knowledge and temporal information dropped. As expected, the features from waybills play the most important role for the service time prediction. The features from spatial and temporal domain also contribute to the prediction, which shows the complexity of the service time prediction.

Performance w.r.t. Different Sizes of Support Set. To demonstrate the performance of MetaSTP at locations with limited observations, we choose two representative types of locations (residence and office building), and report the change of MAE with the increase of the size of the support set in Figure 7. Note that in our problem, the empty support set is also included (0-shot), since this case would appear if a location has never appeared in the train datasets as we previously mentioned. As expected, with the increase of support examples, MAE is decreasing, since the model could have more examples to refer to when making the prediction. To further show the effectiveness of location prior knowledge in helping the prediction when observations are limited, we also report the performance of MetaSTP-nLP in Figure 7. As can be observed, MetaSTP consistently outperforms MetaSTP-nLP with different number of support examples. Most importantly, MetaSTP is much more robust when the support set is empty, since it considers the location prior knowledge which reflects the delivery situation of locations to some extent.

5 DEPLOYMENT

An intelligent waybill assignment system based on MetaSTP is used internally in JD Logistics. It provides the reference to the station master. The system interface is shown in Figure 8, which consists of 3 panels, *Delivery Station*, *Computation* and *Assignment Result*.

Firstly, in the *Delivery Station* panel, the station master can select one of delivery trips of today, and the details of the batch of waybills pending to be delivered would be listed in the table.

Secondly, in the *Computation* panel, the station master is asked to input the number of couriers to conduct the delivery for the batch of waybills. After the "Assign" button is clicked, the batch of waybills would be assigned to couriers. The assignment is formulated as the well-known distance/time-constrained capacitated vehicle routing problem (DCVRP) [18] to minimize the total travel cost as well as to balance couriers' workload. The locations in DCVRP are derived from the waybills, the travel time between locations is estimated from couriers' historical transitions, and the service time at each location is predicted by MetaSTP. When deployed online, MetaSTP

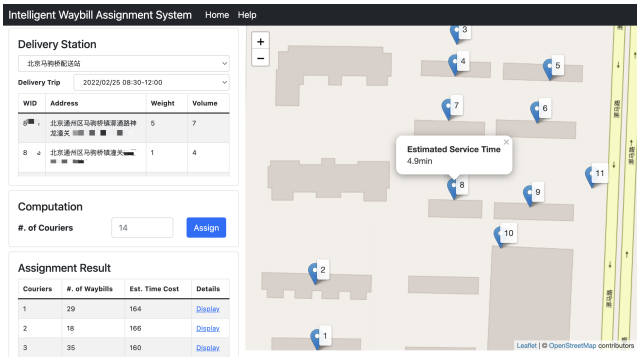


Figure 8: System Interface.

can predict the service time for 250 tasks/s per thread. The multi-thread processing is further implemented to increase its throughput. After that, DCVRP is efficiently solved by a commercial MIP solver.

At last, the results are shown in the *Assignment Result* panel, which contains the number of waybills assigned to each courier as well as the estimated time for completing his/her delivery. When "Display" button is clicked, the assignment for the courier is shown on the map. Each marker is a delivery location, and the number on the marker indicates the planned delivery order. When a marker is clicked, the service time estimated by MetaSTP will be shown.

After the system is deployed, for a courier, the error of delivery trip time estimation is reduced by 14 minutes, which greatly balances the working hours of couriers at the delivery station.

6 RELATED WORK

Stay Time Prediction. The stay time prediction focuses on modeling the time cost of a user staying at a specific POI. Chen et al. [2] use GBRT based on various spatio-temporal features extracted from historical stay points at that location, Liu et al. [19] leverage MLP considering spatio-temporal features as well as context logs in smartphones, and Gidófalv et al. [13] propose to use Markov models based on individual's previous stay sequence. The service time prediction differs from the stay time prediction in that the underlying semantics of stay are specific and different from general ones, i.e., they are driven by the delivery tasks but not others, e.g., working, eating. In stay time prediction, what a moving object is doing at a location usually is unknown, and can only be inferred from sequential patterns or smartphone logs, while we explicitly know the stay is caused by delivery, and the delivery task should be leveraged. The most similar work is [24], which predicts the service time using a KNN regressor. However, [24] fails to consider the fine-grained floor information in waybills, as well as omits the issue that observations are distributed highly skewed in urban spaces.

Meta-Learning Applications in Spatio-temporal Data Mining. Recently, meta-learning has been adopted in the field of spatio-temporal data mining when data is limited. For example, Yao et al. [34] make spatio-temporal prediction in cities with limited data, and Qin et al. [21] predict the purchase volume in different regions and day types. In addition, in the map search engine, Fang et al. [10] estimate the en route travel time, Fan et al. [9] auto-complete POI and Chen et al. [3] predict the next POI to search. Different from

these studies, we are the first work to leverage meta-learning to improve the predictability of service time in logistics. A representation layer is specially designed to embed the delivery task. Besides, in our scenario, the support set could be empty, which is not the case for the aforementioned works. We design a prior knowledge fusion module to enhance the performance under this case.

Delivery Data Mining. With the digitization progress of the logistics/delivery industry, there are emerging studies focusing on delivery data mining. In addition to travel time estimation [33] and service time prediction [24], there are many other studies of delivery data mining, which are reviewed as follows. Based on waybill data, Ding et al. [7] infer the delivery scope of each merchant, and Wen et al. [31, 32] predict the pick-up order of parcels. With couriers' trajectories, Dahiya et al. [4] find the regions of interest, Srivastava et al. [26] improve the quality of Geocoding, [22, 23, 25] infer the delivery location, and Jiang et al. [17] detect fake locations registered by the merchants. Leveraging couriers' encounter data, Ding et al. [8] estimate the relative location of couriers indoors.

7 CONCLUSION

In this paper, we study the problem of service time prediction (STP), which is fundamental for intelligent logistics, and propose MetaSTP to solve it. MetaSTP treats STP at each location as a learning task, leverages a fine-grained representation layer to encode the complex delivery circumstances of each delivery task, and devises a location prior knowledge enhanced meta-learning method to tackle the location heterogeneity and skewed observation problem. Experiments show MetaSTP outperforms baselines by at least 9.5% and 7.6% on two real-world datasets. Finally, an intelligent waybill assignment system based on MetaSTP is used internally in JD Logistics. A possible future research direction is to incorporate the service time with the travel time to provide an end-to-end delivery time prediction.

ACKNOWLEDGMENTS

We thank Zheyi Pan from JD Technology for the early discussion of this work. This research is supported by the National Key R&D Program of China (2019YFB2103201), the Ministry of Education, Singapore, under its Academic Research Fund (Tier 2 Award MOE-T2EP20220-0011 and Tier 1 Award (RG77/21)), the NSFC Grant (62076027, 61976168, 62076191). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of any funding agencies.

REFERENCES

[1] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.
[2] Jie Chen, Zhu Xiao, Dong Wang, Wangchen Long, Jing Bai, and Vincent Havyarimana. 2019. Stay time prediction for individual stay behavior. *IEEE Access* 7 (2019), 130085–130100.
[3] Yudong Chen, Xin Wang, Miao Fan, Jizhou Huang, Shengwen Yang, and Wenwu Zhu. 2021. Curriculum meta-learning for next poi recommendation. In *KDD*. 2692–2702.
[4] Manjeet Dahiya, Devendra Samatia, and Kabir Rustogi. 2020. Learning locality maps from noisy geospatial labels. In *SAC*. 601–608.
[5] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research* 40, 2 (1992), 342–354.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[7] Xuetao Ding, Runfeng Zhang, Zhen Mao, Ke Xing, Fangxiao Du, Xingyu Liu, Guoxing Wei, Feifan Yin, Renqing He, and Zhizhao Sun. 2020. Delivery Scope: A New Way of Restaurant Retrieval for On-demand Food Delivery Service. In *KDD*. 3026–3034.
[8] Yi Ding, Dongzhe Jiang, Yu Yang, Yunhuai Liu, Tian He, and Desheng Zhang. 2022. P2-Loc: A Person-2-Person Indoor Localization System in On-Demand Delivery. *IMWUT* (2022).
[9] Miao Fan, Yibo Sun, Jizhou Huang, Haifeng Wang, and Ying Li. 2021. Meta-Learned Spatial-Temporal POI Auto-Completion for the Search Engine at Baidu Maps. In *KDD*. 2822–2830.
[10] Xiaomin Fang, Jizhou Huang, Fan Wang, Lihang Liu, Yibo Sun, and Haifeng Wang. 2021. SSML: Self-Supervised Meta-Learner for En Route Travel Time Estimation at Baidu Maps. In *KDD*. 2840–2848.
[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. PMLR, 1126–1135.
[12] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.
[13] Győző Gidófalvi and Fang Dong. 2012. When and where next: Individual mobility prediction. In *Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*. 57–64.
[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT.
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[16] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-Learning in Neural Networks: A Survey. *TPAMI* (2021).
[17] Dongzhe Jiang, Yi Ding, Hao Zhang, Yunhuai Liu, Tian He, Yu Yang, and Desheng Zhang. 2021. ALWAES: an Automatic Outdoor Location-Aware Correction System for Online Delivery Platforms. *IMWUT* 5, 3 (2021), 1–24.
[18] Alvina GH Kek, Ruey Long Cheu, and Qiang Meng. 2008. Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots. *Mathematical and Computer Modelling* 47, 1-2 (2008), 140–152.
[19] Sen Liu, Huanhuan Cao, Lei Li, and MengChu Zhou. 2013. Predicting stay time of mobile users with contextual information. *T-ASE* 10, 4 (2013), 1026–1036.
[20] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *ICLR*.
[21] Huiling Qin, Songyu Ke, Xiaodu Yang, Haoran Xu, Xianyuan Zhan, and Yu Zheng. 2021. Robust Spatio-Temporal Purchase Prediction via Deep Meta Learning. In *AAAI*, Vol. 35. 4312–4319.
[22] Sijie Ruan, Cheng Long, Xiaodu Yang, Tianfu He, Ruiyuan Li, Jie Bao, Yiheng Chen, Shengnan Wu, Jiangtao Cui, and Yu Zheng. 2022. Discovering Actual Delivery Locations from Mis-Annotated Couriers' Trajectories. In *ICDE*. IEEE.
[23] Sijie Ruan, Zi Xiong, Cheng Long, Yiheng Chen, Jie Bao, Tianfu He, Ruiyuan Li, Shengnan Wu, Zhongyuan Jiang, and Yu Zheng. 2020. Doing in one go: delivery time inference based on couriers' trajectories. In *KDD*. 2813–2821.
[24] Junxian Song, Rong Wen, Chi Xu, and Joel Wei En Tay. 2019. Service Time Prediction for Last-Yard Delivery. In *Big Data*. IEEE, 3933–3938.
[25] Yatong Song, Jiawei Li, Liying Chen, Shuiping Chen, Renqing He, and Zhizhao Sun. 2021. A Semantic Segmentation based POI Coordinates Generating Framework for On-demand Food Delivery Service. In *SIGSPATIAL*. 379–388.
[26] Vishal Srivastava, Priyam Tejaswin, Lucky Dhakad, Mohit Kumar, and Amar Dani. 2020. A Geocoding Framework Powered by Delivery Data. In *SIGSPATIAL*. 568–577.
[27] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. 2016. WaveNet: A generative model for raw audio. *SSW* (2016).
[28] 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.
[29] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *NIPS* 29 (2016), 3630–3638.
[30] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to estimate the travel time. In *KDD*. 858–866.
[31] Haomin Wen, Youfang Lin, Huaiyu Wan, Shengnan Guo, Fan Wu, Lixia Wu, Chao Song, and Yinghui Xu. 2022. DeepRoute+: Modeling Couriers' Spatial-temporal Behaviors and Decision Preferences for Package Pick-up Route Prediction. *TIST* 13, 2 (2022), 1–23.
[32] Haomin Wen, Youfang Lin, Fan Wu, Huaiyu Wan, Shengnan Guo, Lixia Wu, Chao Song, and Yinghui Xu. 2021. Package Pick-up Route Prediction via Modeling Couriers' Spatial-Temporal Behaviors. In *ICDE*. IEEE, 2141–2146.
[33] Fan Wu and Lixia Wu. 2019. Deepeta: A spatial-temporal sequential neural network model for estimating time of arrival in package delivery system. In *AAAI*, Vol. 33. 774–781.
[34] 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.
[35] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. 2018. DEEPTRAVEL: A neural network based travel time estimation model with auxiliary supervision. In *IJCAI*, Vol. 19. 3655–3661.