Inductive Graph Transformer for Delivery Time Estimation

Xin Zhou
Nanyang Technological University
Singapore
xin.zhou@ntu.edu.sg

Jinglong Wang
Alibaba Group
China
jinglong.wjl@alibaba-inc.com

Yong Liu
Nanyang Technological University
Singapore
stephenliu@ntu.edu.sg

Xingyu Wu
Alibaba Group
China
zhuyang.wxy@alibaba-inc.com

Zhiqi Shen
Nanyang Technological University
Singapore
zqshen@ntu.edu.sg

Cyril Leung
Nanyang Technological University
Singapore
cleung@ntu.edu.sg

ABSTRACT

Providing accurate estimated time of package delivery on users' purchasing pages for e-commerce platforms is of great importance to their purchasing decisions and post-purchase experiences. Although this problem shares some common issues with the conventional estimated time of arrival (ETA), it is more challenging with the following aspects: 1) *Inductive inference.* Models are required to predict ETA for orders with unseen retailers and addresses; 2) *High-order interaction of order semantic information.* Apart from the spatio-temporal features, the estimated time also varies greatly with other factors, such as the packaging efficiency of retailers, as well as the high-order interaction of these factors. In this paper, we propose an inductive graph transformer (IGT) that leverages raw feature information and structural graph data to estimate package delivery time. Different from previous graph transformer architectures, IGT adopts a decoupled pipeline and trains transformer as a regression function that can capture the multiplex information from both raw feature and dense embeddings encoded by a graph neural network (GNN). In addition, we further simplify the GNN structure by removing its non-linear activation and the learnable linear transformation matrix. The reduced parameter search space and linear information propagation in the simplified GNN enable the IGT to be applied in large-scale industrial scenarios. Experiments on real-world logistics datasets show that our proposed model can significantly outperform the state-of-the-art methods on estimation of delivery time.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Learning latent representations**.

KEYWORDS

Delivery Time Estimation, Estimated Time of package Arrival, Inductive Graph Transformer, Graph Convolutional Networks

1 INTRODUCTION

The invention of the World Wide Web enables users to share information and products on the Web, which eventually drives the development of e-commerce platforms [30]. The contemporary prevalence of e-commerce leads to millions of packages being delivered worldwide every day [8, 27]. Retailers on e-commerce platforms show consumers various information about their products and services, such as prices, descriptions, and estimated delivery time. The delivery time is valued more important than price when consumers have a deadline to receive their packages. It has been shown to affect the decisions of 87% online consumers [5]. Hence, the accurate estimation of package arrival time (ETA) not only impacts the e-commerce revenue but also shapes the consumers' expectations. Either over-promising or under-promising ETA may result in detriment of online retailers and consumers.

In practice, the physical delivery time is always subject to uncertainty due to the packaging efficiency of retailers, inventory, scheduling, or transportation factors. In our scenario, we study the origin-destination (OD) delivery time estimation, which aims to predict the package delivery time given the information of retailer, OD addresses, and payment time. Although the OD travel time estimation is widely studied in the transportation domain [16, 22], its challenges are usually different from those of OD ETA in the delivery scenario. For example, both problems face the challenges of the absence of route information and efficiently utilizing the spatio-temporal features. However, the OD ETA in the delivery scenario inevitably involves predicting orders with unseen retailers or addresses, *etc.* Furthermore, its prediction varies greatly according to multiplex signals (*e.g.*, retailer, address, payment time) and the high-order interaction of these signals.

Existing works in the transportation domain [15, 16, 22] either assume all origins and destinations can be observed for spatial network construction or make straightforward use of the raw features. Graph-based methods [15, 22] exploit the power of graph neural networks (GNNs) to learn the representations of trips. However, these approaches require that all addresses in the OD graph are present during model training, making them hard to be generalized

to unseen trips. Another line of work [31, 35] utilizes deep neural networks to predict ETA based on the raw features. However, they failed in capturing the high-order semantic information that occurred in the interactions between the attributes of a package (*e.g.*, retailer, addresses, and payment time), which may also affect the accuracy of ETA prediction. The closest study is the food delivery time estimation conducted by Alibaba group on Ele.me [43]. However, it focuses on city-level food delivery and highlights its contributions on feature engineering of existing models.

In this paper, we propose an inductive graph transformer (IGT) for the estimation of package delivery time on a national level. Contrary to currently coupled graph transformers [10, 17] that leverage transformer as a substitution of information aggregator in GNN, we decouple the GNN and transformer pipeline in IGT. Specifically, we use transformer as a regression function and frame collected orders into sentences of NLP. Analogous to word in a sentence for language processing, we denote retailer, addresses, and payment time as "element" in an order. In this way, we can represent orders as a heterogeneous graph with different types of nodes (*i.e.*, element) and capture the high-order semantic relations with information propagation of GNN. Furthermore, we adapt our model to large-scale scenarios by simplifying GNN with the removal of non-linear activation and linear transformation matrix. In the following sections, we will be using the terms "element" and "node" interchangeably based on the discussed context. With our simplified GNN, the embeddings of unseen elements in an order can be approximated from the observed neighbor elements. In practice, orders have periodicity and recurring natures. We capture the temporal evolution of elements with a gated recurrent unit (GRU) [3]. Finally, we fuse the node embeddings and raw features of orders into a transformer-like architecture for ETA prediction.

In summary, the contributions made in this work are as follows:

- We propose an inductive graph transformer for ETA prediction in package delivery. Instead of coupling transformer as an alternative attention mechanism for neighborhood information aggregation in GNNs, we use a decoupled architecture that feeds both extracted heterogeneous information from GNNs and raw features of orders into transformers for ETA prediction.
- We design a simplified heterogeneous graph convolutional network that can encode high-order semantic information of orders into latent element embeddings and dynamically update element embeddings with GRU. Specifically, we remove the non-linear activation and the linear transformation matrix of the current graph convolutional network to adapt to large-scale industrial scenarios. As a result, only the linear operator for information aggregation and propagation is retained in IGT.
- We conduct comprehensive experiments on two large-scale logistics datasets, and the results show that the proposed model is significantly superior to the existing methods.

2 RELATED WORK

2.1 Estimated Time of Arrival

Previous ETA prediction methods can be categorized into two main groups: 1) route-based methods and 2) OD-based models.

2.1.1 Route-based Methods. Apparently, route-based models take route information into consideration. With the available of route information, a line of work uses (graph) neural networks to encode the spatial and temporal dependency. For example, WDR [33] devises a Wide-Deep-Recurrent (WDR) architecture based on sequential features to formulate ETA as a regression problem. CompactETA [12] and ConSTGAT [11] exploit graph attention networks (GAT) [29] on a spatio-temporal graph for representation learning. HetETA [15] models different turning directions into a heterogeneous information graph and then learns the representation of the graph for ETA prediction. Note that the road structure in HetETA is assumed to be static, which is different from the package delivery in which graphs evolve dynamically. In logistics system, DeepETA uses a spatial-temporal sequential neural network to capture the regularity of the delivery pattern and the sequence of packages in a delivery route [35]. The above networks assume the nodes are present in the training of the embeddings and do not apply to inductive settings.

2.1.2 OD-based Methods. Due to privacy consideration, it is usually difficult to obtain the path information in many application scenarios. Under such a setting, OD-based methods that utilize the features of origin and destination are proposed for ETA prediction. In [31], the authors propose a simple model to predict travel time of a given trip by weighing the travel time of neighbors. The neighbors are filtered according to both the spatial information of origin and destination and the temporal information of trip starting time. Li *et al.* develop a novel representation learning framework for travel time prediction, which uses road network, spatio-temporal smoothing prior, and additional trip attributes to represent the real world trip attributes [22]. The model can distill road network information in historical trips for accurate ETA prediction. However, it assumes the path information can be observed in historical trips, which is lack of generalization. The most relevant existing work is the ETA prediction for food delivery [43], which is as complex as our problem because it is dependent on numerous features from couriers, restaurants, and traffic. However, this work mainly focuses on the exploitation of feature engineering to feed existing models.

2.2 Graph Transformers

Transformer [28], GNNs, and their variants have achieved great success in many fields. A line of work has been proposed recently to fuse the two as a graph transformer. For example, Yun *et al.* develop graph transformer networks on heterogeneous graphs targeting to transform a given heterogeneous graph into a meta-path based graph and then perform convolution consequently [37]. Notably, their focus behind the use of attention framework is for interpreting the generated meta-paths. Another line of transformer on heterogeneous information networks is developed in [17]. The proposed graph transformer uses transformer to substitute the attention mechanism in GAT. Furthermore, Zhou *et al.* propose a transformer based generative model which generates temporal graphs by directly learning from dynamic information in networks [38]. In [10], the authors propose a generalization of transformer neural network architecture for homogeneous graphs of arbitrary structure. These existing methods generally employ the attention mechanism

in transformer for information aggregation. It is difficult to apply them to inductive graphs.

3 PROBLEM STATEMENT

This paper targets to solve the ETA prediction problem based on historical order delivery information. In our scenario, the platform will generate an order either after consumers have paid for their products or when the consumers are viewing a product page. We formally define an order and the problem of delivery time estimation as follows.

Definition 3.1: Order. An order is represented as a tuple: $\mathbf{x}_i = (r_i, o_i, d_i, t_i)$, where r_i denotes the retailer, o_i, d_i are the origin location and destination of the order, respectively. t_i is the payment time of this order. Each element in an order tuple is coupled with a hand-crafted dynamic feature vector describing its statistical information. Thus, we can denote the dataset with N historical orders by $\mathbf{X} = \{\mathbf{x}_i | i = 1, 2, \cdots, N\}$, and the associated delivery time by $\mathbf{y} = \{y_i | i = 1, 2, \cdots, N\}$. A delivery time y_i usually is calculated from the payment to the time when the order is signed.

Definition 3.2: Problem of Delivery Time Estimation. Given a set of historical orders $\mathbf{X}$, the delivery time $\mathbf{y}$ and the input features $\mathbf{Z}$, we aim to estimate the delivery time y_q of a query order $\mathbf{x}_q$.

4 INDUCTIVE GRAPH TRANSFORMER

Since graph convolutional network (GCN) [20] was first outlined to generalize convolutional neural networks (CNNs) on graph structured data, GCN and its variants have been successfully applied to various domains [2, 32, 36, 39–42]. As the order graph can be easily constructed from the historical orders. Each element in an order tuple is represented as node in the order graph, and two nodes are linked if they have occurred in the same order. Considering the computational complexity of dealing with large-scale order graphs, we further limit the links to adjacent elements of an order.

Definition 4.1: Construction of Heterogeneous Graph. Specifically, given an order (r_i, o_i, d_i, t_i), the retailer node r_i can only connect to the sender address node o_i. The sender address node o_i can link to both the retailer node r_i and the receiver address d_i, and so forth. That is, each order will generate only 6 edges instead of 12. In this way, the dense of the graph can be cut down by half. Meanwhile, the information from one node can be propagated to any other nodes via stacking more convolutional layers of GCN.

With the constructed heterogeneous graph, we can decouple the graph into a set of bipartite graphs and perform graph convolution on the bipartite graphs. The transformation operator and activation layer in GCN pose prohibitive cost for large-scale graphs. We simplify the graph convolutional layer with linear computation. However, apart from the structural information, the nodes in the graph may exhibit different temporal patterns. We adopt GRU to analyze temporal correlations on time-series axis at the node-level. GRU solves the vanishing gradient problem of a standard recurrent neural network (RNN) by using update gate and reset gate. Compared with the long short-term memory (LSTM) network, GRU is faster to train due to the fewer number of weights and parameters to update during training. The output of GRU as well as the raw input features are feed into a transformer variant for final prediction. Figure 1 shows the overall framework of the proposed IGT model.

It consists of two main modules: 1) TheGCN (*i.e.*, Temporal and Heterogeneous GCN) and 2) ETAformer (*i.e.*, ETA Prediction with Transformers). Next, we introduce the details of each component.

4.1 TheGCN

Suppose $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the order graph derived from historical orders, where $\mathcal{V}$ and $\mathcal{E}$ denote the node set and edge set respectively. Each node $v \in \mathcal{V}$ and each edge $e \in \mathcal{E}$ are associated with type mapping functions. Our scenario includes four types of nodes and one type of edges. Hence, the constructed graph will be a heterogeneous graph. We first introduce how GCN works on homogeneous graphs, then we show how to apply the convolutional operator to heterogeneous graphs. A typical GCN on homogeneous graphs that recursively updates the hidden embedding $\mathbf{H}^l$ at the l-th layer is defined as:

$$\mathbf{H}^l = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{l-1}\mathbf{W}^l\right), \tag{1}$$

where $\mathbf{W}$ is the learnable transformation matrix and $\sigma(\cdot)$ is a non-linear function, *e.g.*, the ReLu function. $\hat{\mathbf{A}} = \hat{\mathbf{D}}^{-1/2}(\mathbf{A}+\mathbf{I})\hat{\mathbf{D}}^{-1/2}$ is the re-normalization of the adjacency matrix $\mathbf{A}$. $\hat{\mathbf{D}} = \text{diag}(d_1, \cdots, d_{|\mathcal{V}|})$ is the diagonal degree matrix of $\mathbf{A}+\mathbf{I}$, where each entry on the diagonal is equal to the row-sum of the adjacency matrix $d_i = 1 + \sum_j a_{ij}$. Each node v_i in the graph has a corresponding D-dimensional latent feature vector $\mathbf{h}_i \in \mathbb{R}^D$. The initial feature matrix $\mathbf{H}^0 = [\mathbf{h}_0, \cdots, \mathbf{h}_{|\mathcal{V}|}]^T$ stacks $|\mathcal{V}|$ feature vectors on top of one another.

In order to apply the graph convolution operator on heterogeneous graphs, a couples of GCN variants are proposed in other domains [26, 44]. However, they are both computationally expensive and prone to over-fitting. Inspired by the study of simplified homogeneous SGC [34], we design a heterogeneous GCN which towards computationally efficiency without compromising the performance. Compared with SGC [34], we further remove its transformation matrix to reduce the parameter search space. This allows our model can be trained with large-scale datasets. Given the heterogeneous graph $\mathcal{G}$, we first construct a set of bipartite subgraphs based on the combinations of the node types. Suppose the number of node types is n, the number of bipartite subgraphs will not exceed $\binom{n}{2} = \frac{(n)!}{2!(n-2)!}$. In detail, we construct a sub-graph $\mathcal{G}_{i,j}$ from the original graph $\mathcal{G}$ based on the two node types i and j. In $\mathcal{G}_{i,j}$, the type of a node belongs to either i or j, and any two nodes in subgraph $\mathcal{G}_{i,j}$ will be linked if they are connected in the original graph $\mathcal{G}$. As $\mathcal{G}_{i,j}$ is undirected, we have $\mathcal{G}_{i,j} = \mathcal{G}_{j,i}$. We denote the adjacency matrix of $\mathcal{G}_{i,j}$ as $\mathbf{A}_{i,j}$:

$$\mathbf{A}_{i,j} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^{\mathbf{T}} & \mathbf{0} \end{pmatrix}, \tag{2}$$

and each entry $R_{uv} \in \mathbf{R}$ is 1 if node $u \in \mathcal{G}_{i,j}$ is connected with node $v \in \mathcal{G}_{i,j}$ in $\mathcal{G}$; Otherwise, R_{uv} is set to 0. Following [20], we also perform the re-normalization trick on $\mathbf{A}_{i,j}$, resulting as $\hat{\mathbf{A}}_{i,j}$. To perform convolution operation on bipartite graphs, we update the vanilla GCN in Eq. (1) as:

$$\mathbf{H}^l_{i,j} = \hat{\mathbf{A}}_{i,j}\mathbf{H}^{l-1}_{i,j}, \tag{3}$$

where $\mathbf{H}^l_{i,j}$ denotes the embeddings of nodes in $G_{i,j}$ at the l-th GCN layer. Suppose the number of nodes with type i is N_i, then $\mathbf{H}_i \in \mathbb{R}^{N_i \times D}$ is the feature matrix of node type i by stacking this type of
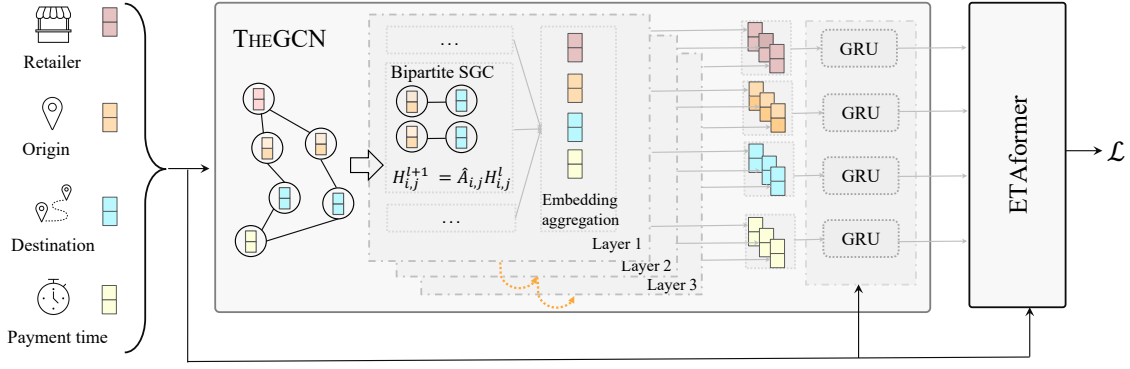
Figure 1: The overall framework of the proposed IGT model. There are two main modules in IGT: 1) TheGCN (*i.e.*, Temporal and Heterogeneous GCN) and 2) ETAformer (*i.e.*, ETA Prediction with Transformer).

nodes on top of one another. Analogously, $\mathbf{H}^l_{i,j} \in \mathbb{R}^{(N_i+N_j) \times D}$. Note that Eq. (3) simplifies the vanilla GCN by removing both the collapsing weight matrix and the non-linear layer, which further simplifies SGC [34] with the removal of the non-linear activation. Meanwhile, the reduction of trainable parameters prevents the model from over-fitting. The linear information propagation enables the model to be computationally efficient. We name this simplified graph convolution for bipartite graphs as Bipartite SGC in Figure 1.

As one type of nodes may be involved in multiple bipartite graphs and participate the information propagation simultaneously, we aggregate the propagated information for each type of nodes within each layer as follows,

$$\mathbf{H}^l_i = \text{AGGREGATE}(\{\mathbf{H}^l_{i,j}[: N_i] \mid j = 1, ..., n, \text{ and } \mathcal{G}_{i,j} \neq \emptyset\}), \quad (4)$$

where AGGREGATE is an aggregation function, and n is the number of node types. Here, we use the sum aggregation. Other aggregation operators [4] can also be applied for aggregating the embeddings of nodes belonging to the same type. In Eq. (4), we use the slice operator to extract the embeddings of i-type nodes.

One benefit of such information aggregation within each layer is that the information from different types of nodes can be completely integrated with each other. The integrated information is then fed into the next layer for propagation. Another benefit is that TheGCN updates the embeddings of all nodes instead of a batch of nodes in each iteration to avoid the memory staleness problem [18].

With the information obtained in each layer, we update the nodes' final embeddings at time $t-1$ by aggregating all intermediate embeddings. That is:

$$\mathbf{H}^{t-1}_i = \text{AGGREGATE}(\mathbf{H}^0_i, \mathbf{H}^1_i, ..., \mathbf{H}^L_i), \quad (5)$$

where L is number of propagation layers. The aggregation function AGGREGATE can be any functions discussed in [4]. Here, we use the mean aggregation for final node updating.

To further fuse the temporal information into the learnt embeddings $\mathbf{H}^{t-1}_i$ at time $t-1$, we use the gating mechanisms to control and manage the flow of information.

$$\mathbf{H}^t_i = \text{UPDATE}(\mathbf{H}^{t-1}_i, \mathbf{Z}^t_i), \quad (6)$$

where $\mathbf{Z}^t_i$ is the raw input feature vectors of i-type elements in orders at time t. Here, UPDATE is a learnable update function, *e.g.*,

a recurrent neural network such as LSTM or GRU. In this work, we adopt the GRU network because it owns fewer number of gates and is as effective as LSTM.

Note that we use a normalized adjacency matrix constructed from historical orders to avoid data leakage in model training stage. However, in inference stage, we use the normalized adjacency matrix built upon both training and test data [14]. Under inductive setting, our proposed TheGCN can fuse information from its neighbors within the same order $\mathbf{x}_i$ to learn the embeddings of unseen nodes. In extremely case, all nodes in $\mathbf{x}_i$ are never observed in training, which means that no neighbor information can be used. TheGCN adopts GRU to update the node's embeddings from raw input features. Hence, raw features will be used to estimate the delivery time of this order.

4.2 ETAformer

The standard transformer is designed for word prediction, and a sentence is treated as a fully connected graph. An order in our scenario is analogy to a sentence in natural language processing (NLP) with less than tens or hundreds of elements. As such, the ETA prediction is substantially amenable to transformers. Large transformer models can be trained on such fully connected graphs. Furthermore, each word attends to each other word in a sentence in the transformer architecture, which enables the high-order interactions among words. In NLP, the prediction of a word in a sentence can vary with context, other words etc. Equivalently, in our scenario, the ETA prediction relies not only on the raw features from retailers, addresses, payment time, but also the high-order interactions of these features. However, the standard transformer assume words are encoded within the same embedding space. In practice, the nodes in heterogeneous graph may vary in feature size as they belonging to different types. Thus, to encapsulate the input feature vectors into transformer, we first need to reshape the feature space. That is:

$$\mathbf{e}_i = f(\mathbf{z}_i, \mathbf{h}^t_i), \quad (7)$$

where $\mathbf{z}_i$ is the raw feature of node i, $\mathbf{h}^t_i$ is the output embedding of node i from TheGCN, and f is a mapping function to align nodes

Algorithm 1 Pseudo-code for Inductive Graph Transformer (IGT)

1: **Inputs:** Set of orders $\mathbf{X}$, the associated delivery time $\mathbf{y}$ and input feature vectors $\mathbf{Z}$.
2: Construct heterogeneous graph $\mathcal{G}$ from $\mathbf{X}$.
3: Initialize latent embeddings $\mathbf{H}^0$ of nodes in $\mathcal{G}$ with Xavier.
4: Extract bipartite subgraphs with node type i and j $\mathcal{G}_{i,j}$ from $\mathcal{G}$ and the latent embeddings $\mathbf{H}^0_{i,j}$ from $\mathbf{H}^0$.
5: **for** $\mathbf{X}^t$ *in* $\mathbf{X}$ **do**   ▷ load a batch
6:   **for** l *in* $[1, ..., L]$ **do**   ▷ layer-wise propagation
7:     **for** i *in* $[1, ..., n]$ **do**
8:       **for** j *in* $[1, ..., n]$ **do**
9:         $\mathbf{H}^l_{i,j} = \hat{\mathbf{A}}_{i,j}\mathbf{H}^{l-1}_{i,j}$ ▷ bipartite SGC
10:       **end for**
11:     **end for**
12:     **for** i *in* $[1, ..., n]$ **do**
13:       $\mathbf{H}^l_i = \text{AGGREGATE}(\{\mathbf{H}^l_{i,j}[:N_i]\})$ ▷ Eq. 4
14:     **end for**
15:   **end for**   ▷ end layer propagation loop
16:   **for** i *in* $[1, ..., n]$ **do**
17:     $\mathbf{H}^{t-1}_i = \text{AGGREGATE}(\mathbf{H}^0_i, \mathbf{H}^1_i, ..., \mathbf{H}^L_i)$ ▷ Eq. 5
18:     $\mathbf{H}^t_i = \text{UPDATE}(\mathbf{H}^{t-1}_i, \mathbf{Z}^t_i)$ ▷ Eq. 6 for temporal update
19:   **end for**
20:   $\hat{\mathbf{y}}^t = \text{ETAformer}(\mathbf{H}^t, \mathbf{Z}^t)$ ▷ ETA prediction
21:   $\mathcal{L} = \text{MAE}(\mathbf{y}^t, \hat{\mathbf{y}}^t)$ ▷ loss
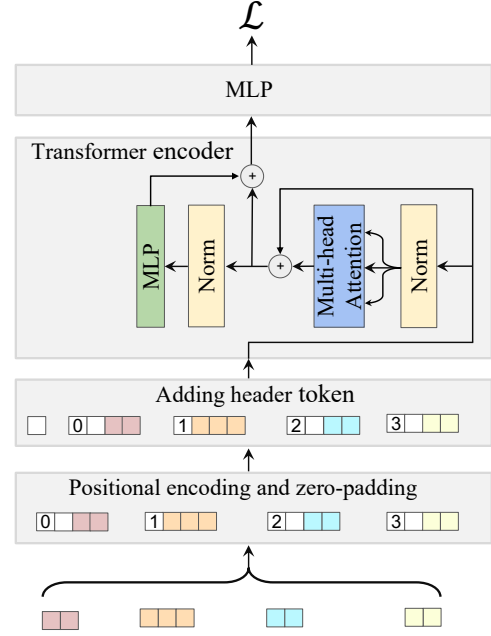22: $\mathcal{L}$.backward()   ▷ back-propagate
23: **end for**

with the same dimensionality. An intuitive function is using multilayer perceptron (MLP) to map features of different dimensional spaces to the same space. However, it introduces additional parameters to the model, so we adopt zero-padding on each node of an order. Finally, we reshape the order into $\mathbf{x}^i \in \mathbb{R}^{n \times W}$, where n is the number of elements in an order and W is the padded embedding size of each element.

For capturing the order-level information, we prepend a learnable header token similar to that of BERT [7] and ViT [9]. In implementation, we use a MLP with one hidden layer for token learning. On positional encodings, existing graph transformers design various methods to preserve the distance information between nodes. Our decoupled architecture utilizes the standard positional encoding of transformer. Because an order $\mathbf{x}_q$ in our scenario can be naturally framed as a sequence of elements. The header token $\mathbf{e}^{header}_q$, positional encodings $\mathbf{e}^{pos}_q$, and mapped node embeddings $\mathbf{e}_q$ of $\mathbf{x}_q$ are then fed into the transformer variants. Upon the output, we finally use a layer normalization (LN) and a single layer MLP to predict the ETA, as shown in Figure 2.

Formally, we denote the raw feature of a query order $\mathbf{x}_q$ as $\mathbf{z}_q = (z^1_q, z^2_q, ..., z^n_q)$, where z^i_q is the raw feature of i-th element in the order. The ETA prediction of order $\mathbf{x}_q$ based on the transformer-like models can be presented as follows,

$$\begin{aligned} \mathbf{e}_q &= f(\mathbf{z}_q, \mathbf{h}^t_q), \\ \hat{\mathbf{e}}^{header}_q &= \text{Transformer}(\mathbf{e}^{header}_q, \mathbf{e}^{pos}_q, \mathbf{e}_q), \\ \hat{y}_q &= \text{MLP}\left(\text{LN}(\hat{\mathbf{e}}^{header}_q)\right). \end{aligned} \quad (8)$$

Figure 2: The architecture of ETAformer.

Table 1: Statistics of experimental datasets.

Datasets	D1	D2
# of orders	375,393	1,817,766
# of retailers	4,729	8,741
# of days	110	47
validation/test days	(10, 15)	(5, 10)
# of orders in test	61,647	348,037
# of orders with unseen retailers	913	930
# of orders with unseen addresses	61	3

4.3 Model Training

For training the proposed IGT model, we utilize the Adam [19] as the optimizer, and use the Mean Absolute Error (MAE) as our loss function. Considering a set of N samples $\mathbf{y} = (y_1, y_2, \cdots, y_N)$ representing the ground truth and the associated predictions $\hat{\mathbf{y}}$, the MAE-based loss function is defined as follows:

$$\mathcal{L} = \text{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|. \quad (9)$$

The details of the optimization algorithm used to learn the IGT model are summarized in Algorithm 1.

5 EXPERIMENTS

5.1 Experimental Datasets

We utilize two large-scale logistics datasets collected from one of the world's largest e-commerce platforms to evaluate the performance of IGT as well as the baselines.

5.1.1 Data Description and Statistics. The first dataset contains 0.38 million samples from 4,729 retailers (denoted by D1), ranging from January 3rd to April 22nd, 2021. The second dataset includes

1.8 million samples collected from 8,741 retailer (denoted by D2), ranging from June 3rd to July 22nd, 2021. The two datasets are not only differentiate in graph size, collected time, but also in collection regions. Both datasets are orders targeting a fixed receiving city. Specifically, D1 is collected with all the orders sent to Weihai, China. D2 is for packages oriented to Hangzhou, China. The population of the receiving city of Hangzhou is 3 times more than it is in Weihai. From Table 1, we can also observe that the retailers in D2 are relatively more active than those in D1 with an average of 208 orders. For ETA prediction, we follow the chronological data splitting for training, validation, and test. To be specific, we split the orders in the last 25 days of D1 by 10:15 as validation and test, the last 15 days of D2 by 5:10, orders from other days are used for training. In Table 1, we denote the unseen nodes (*e.g.*, retailers, addresses) as the nodes in test that never occurred in training. Note that the datasets also collect the evolving features of retailer, payment time, sender address, and receiver address.

To facilitate our interpretation of the prediction results on different datasets, we plot the histogram of the diurnal pattern on order payment time in Figure 3. From the plots, we can observe the payment time pattern is consistent with human regularity. The two datasets also vary in order volume, that is, the number of orders received in Hangzhou is almost an order of magnitude of Weihai at each payment time.
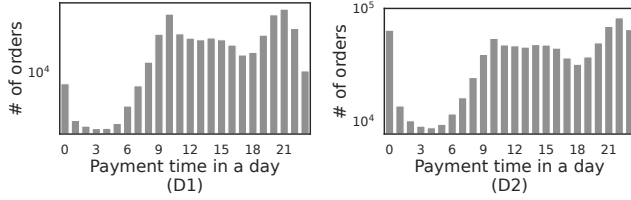

Figure 3: Diurnal pattern of orders. $y-$axis is in log scale.

5.2 Comparison Methods and Metrics

We compare the proposed IGT model with the following baselines. 1). **Linear Regression (LR)** models the delivery time of orders by training a linear regression model based on the raw features of orders. We use the LR in scikit-learn [25] for evaluation. 2). **XGBoost** [1] is a gradient boosting decision tree based regression model. The input is the same as the LR model. We use the XGBRegressor in xgboost [1] with MAE as its evaluation metric. 3). **MURAT** [22] captures underlying road network structures as well as spatio-temporal prior knowledge for OD travel time estimation. The road information is not available in our dataset, the OD graph is built upon the sender and receiver address. We use both ResNet18 and deep feed forward neural networks (DNN) to map the learned representations to target tasks. In our scenario, we found DNN performs better than ResNet18. The reported results in our paper are based on DNN. 4). **xDeepFM** [23] learns both low- and high-order feature interactions in both explicit and implicit fashions in an extreme deep factorization machine framework. We transform the recommendation task into a regression task for ETA prediction. 5). **TEMP$_{rel}$** [31] is an OD-based approach for travel time estimation. It averages the scaled travel times of all neighboring routes

with similar origin and destination for ETA prediction. 6). **DeepETA** [35] is an end-to-end network for last-mile package delivery estimation. It not only takes route features into consideration but also encodes the spatial and temporal information in deep neural networks for ETA prediction. As no route information is available in our data, we use the sender and receiver addresses as the route history. 7). **HetETA** [15] targets to estimate the arrival time of taxis within a city. It uses gated convolutional neural networks and ChebNet [6] to capture the relations between through spatial and temporal perspectives. This method relies on the trajectory information. Again, as our data lacks this type of information, we perform Het-ChebNets only on the road network. The road network is built up as the same as MURAT.

5.3 Performance Comparison

All models are trained on a single GeForce RTX 2080 Ti with 11 GB memory. For traditional machine learning models, we feed full training data into LR, XGBoost, and TEMP$_{rel}$ for ETA prediction. Other models are trained in batches with a size of 8192. We monitor the loss over the validation set and apply early stopping if no improvement happens after 100 consecutive epochs. Considering the model convergence, the total epochs for training are fixed at 1000. We initialize the latent embeddings $\mathbf{H}^0$ of nodes in $\mathcal{G}$ with the Xavier method [13], tune the number of layers L of TʜᴇGCN in $\{1, 2, 3, 4, 5\}$, and the embedding size D of nodes in $\{16, 32, 64, 128, 256\}$. We implement our model [2] in PyTorch [24].

Following [22], we evaluate our model and all baselines under the metrics of MAE, Mean Absolute Percentage Error (MAPE), and Mean Absolute Relative Error (MARE). The definitions of MAPE and MARE are as follows,

$$\text{MAPE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{10}$$

$$\text{MARE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{\sum_{i=1}^{N} y_i}. \tag{11}$$

Analogous to MAE defined in Eq. (9), $\mathbf{y}$ denotes the ground truth of package delivery time, $\hat{\mathbf{y}}$ denotes the predicted delivery time, and N denotes the number of samples in test data.

Table 2 presents the performance comparison between the proposed IGT model and the baselines on two datasets. The results on three metrics show our proposed model is capable of improving the best baseline by more than 6%. Especially on dataset D1, IGT gains significant improvement over the best baseline by up to 11%. Compared with D2 in Table 1, D1 is more sparse. The averaged number of orders for each retailer is 0.7 and 4.4 per day in D1 and D2, respectively. The results between D1 and D2 indicate that IGT performs better with sparse graph. However, sparsity may partially contribute to the performance. We analyze why the performance of the two datasets vary heavily through the perspective of information theory.

We utilize entropy to quantify the entropy of the distribution of delivery time under a payment time.

---

[1]https://xgboost.readthedocs.io

[2]Source code is available at: https://github.com/enoche/IGT-WSDM23.

Table 2: Performance comparison on ETA prediction between our model and the baselines. We mark the best results on each dataset under each metric in Bold face, and the second best underlined. *Improvement (%)* is calculated between the best result achieved from the baselines and the result of our model.

Datasets	Metric	LR	XGBoost	MURAT	xDeepFM	TEMP$_{rel}$	DeepETA	HetETA	IGT	*Improvement (%)*
D1	MAE(*hours*)	23.46	11.94	12.15	11.85	17.19	16.61	13.41	**10.53**	11.14
	MAPE(%)	31.28	15.46	16.11	15.73	26.48	26.38	18.86	**13.35**	13.65
	MARE(%)	34.02	17.32	17.62	17.18	24.93	24.09	19.45	**15.28**	11.06
D2	MAE(*hours*)	10.75	8.16	8.83	7.99	10.53	14.58	10.57	**7.39**	6.13
	MAPE(%)	26.21	17.24	19.31	16.50	24.37	35.69	24.66	**14.22**	9.82
	MARE(%)	22.92	17.39	18.83	17.03	22.45	31.08	22.53	**15.75**	6.11

$$p^t = -\sum_{y_i^t \in \mathbb{S}^t} p(y_i^t)\log p(y_i^t), \tag{12}$$

where $\mathbb{S}^t$ is the set of delivery time for orders placed at t. The entropy is larger for orders with greater divergent delivery time that placed at the same time t. As such, we plot the entropy of the two datasets with regard to the payment time in Figure 4. The figure reveals dataset D1 has a larger averaged entropy value than that of dataset D2. As a result, the MAE value for D1 is much larger than D2 for all models. We will re-examine the variation of entropy in Section 5.6.

As for the baselines, the performance of LR is poor mainly because it can only use the raw features for regression tasks. Other regression models such as XGBoost and xDeepFM can capture the interactions between features, thus achieving better performance. TEMP$_{rel}$ improves LR on both datasets implying the importance of spatial (*i.e.*, longitude, latitude) and temporal information in ETA prediction. The performance of MURAT, DeepETA and HetETA is compromising as they are designed to take advantage of route information for ETA prediction. However, MURAT and HetETA utilize graph representation learning to learn the meaningful embeddings of OD links and achieve better performance over DeepETA. Similarly, our model also uses graph learning (*i.e.*, GCN) to capture the structural information between orders. Nevertheless, we focus on node representation learning, and a link that connects retailer, addresses and payment time is represented by concatenating the embeddings of these nodes.
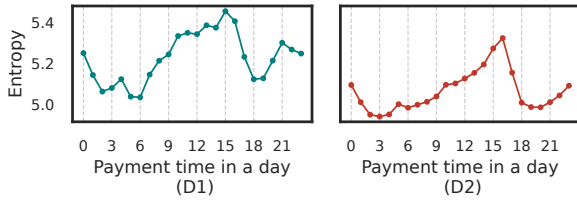

Figure 4: Entropy of the delivery time for orders placed at payment time.

5.4 Ablation Study

We perform ablative study on each component of IGT to examine its contribution in ETA prediction. Recalling that IGT consists of two components, namely TheGCN and ETAformer. The decoupled architecture of IGT enables us to perform ETA prediction on each component without the prerequisite of the other one. First, to evaluate TheGCN, we feed the concatenation of the latent embeddings

Table 3: Ablation study of IGT.

Datasets	Metric	TheGCN	ETAformer	IGT
D1	MAE(*hours*)	11.79	10.67	10.53
	MAPE(%)	15.67	13.22	13.35
	MARE(%)	17.09	15.48	15.28
	time/epoch(*seconds*)	2.66	10.47	13.34
D2	MAE(*hours*)	7.99	7.66	7.39
	MAPE(%)	16.62	15.43	14.22
	MARE(%)	17.03	16.34	15.75
	time/epoch(*seconds*)	11.31	42.86	63.06

obtained from TheGCN into a single layer MLP for ETA prediction. As for ETAformer, it can cope with the raw features directly. We report the performance on the two datasets under each metric in Table 3.

The results show that ETAformer achieve better performance than TheGCN in ETA prediction. Because ETAformer acts on inter-order, it further explores the high-order interactions between elements of an order. Comparing the performance of TheGCN with that of xDeepFM in Table 2, we can observe that merely using of TheGCN in IGT can obtain competitive performance to xDeepFM. Because TheGCN is also able to extract the high-order information between orders by virtue of linear information propagation. The training time in Table 2 demonstrates TheGCN is significantly faster than ETAformer. The combination of TheGCN and ETAformer gains significant improvements over the baselines.

5.5 Hyper-parameter Study

We examine how the number of layers L and the embedding size D of nodes in TheGCN affect the performance of IGT. We collect the MAE performance of IGT under the grid-search of hyper-parameters L and D on both datasets (D1 and D2) and plot the heatmap in Figure 5. We observe that the dimensionality D of the hidden embeddings in TheGCN has a more decisive impact on the ETA prediction. Stacking too many layers L in TheGCN may result in over-smoothing [21] and degenerate the prediction performance. A good choice for the dimensionality of hidden embeddings may depend on the number of orders in the dataset.

5.6 Inductive and Transductive Learning of IGT

We further analyze the prediction results with regard to the types of nodes and the number of orders in each node type. Especially, we examine the prediction performance of IGT under inductive setting where nodes have no historical orders. In our analysis, we compare IGT with xDeepFM to elucidate the use cases where IGT may provide a significant edge.
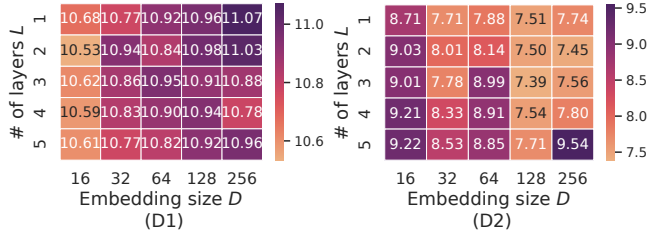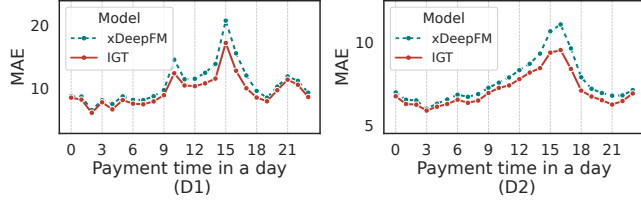
Figure 5: Hyper-parameter study of IGT.

Figure 6: ETA prediction performance with regard to payment time in a day.

Payment time. We plot the ETA prediction performance of IGT and xDeepFM with regard to payment time in a day in Figure 6, from which we can observe that our proposed model always shows better performance against the best baseline method xDeepFM at every payment time. Obviously, the performance of both models exhibits certain fluctuations over payment time. When comparing Figure 6 with Figure 3 and Figure 4, we find the entropy of the delivery time on orders is highly correlated with the prediction results. More orders for training may not improve the ETA prediction performance of the models. It is rational because larger entropy makes the models unable to estimate the delivery time accurately. Both models show large prediction error around 15:00 on the two datasets. The reason is that courier companies usually dispatch the collected packages to the next station around 15:00, products ordered after this time need to wait for the next dispatching slot. In both datasets, our proposed IGT shows marginal improvements over xDeepFM around the peaks.

Retailer and address. As both datasets have orders with unseen retailers and addresses, we can evaluate the performance of IGT under inductive settings. We take data binning to reduce the cardinality of the number of orders with regard to retailer and address. To be specific, we divide the number of orders N received by one retailer into groups: unseen ($N = 0$), small ($0 < N \leq 100$), medium ($100 < N \leq 500$) and large ($500 < N$). Similarly, we divide the number of orders N associated with sender addresses into groups: unseen ($N = 0$), small ($0 < N \leq 500$), medium ($500 < N \leq 1000$) and large ($1000 < N$). We plot the prediction performance on groups with a size larger than 50 in Figure 7. In inductive learning setting, IGT shows better prediction performance than xDeepFM on both datasets under retailer and address. Note that IGT reduces the MAE value by half of xDeepFM on unseen addresses of dataset D1. Although the number of orders on unseen addresses is relatively small as the prediction values show higher variance, we can see IGT is superior to xDeepFM in inductive learning from the left part of Figure 7. The right part of Figure 7 shows that both models do not always learn well with the accumulation of historical orders

from the same retailer or address. Especially on dataset D1, larger number of orders from retailers degrades the performance of both models. Compared with the retailer perspective, the accumulation of orders on address shows continuous improvements from both models on the two datasets.
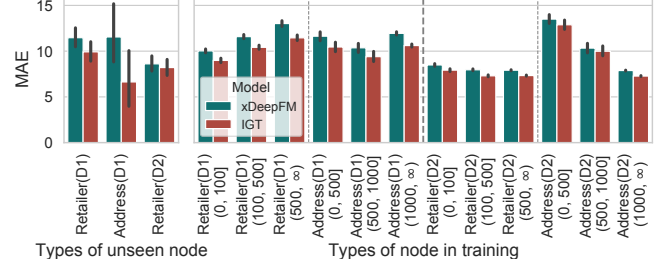

Figure 7: Inductive and transductive learning of IGT with regard to nodes with different number of historical orders.

6 ONLINE EXPERIMENT

To further evaluate the effectiveness of IGT, we evaluate its performance online. We evaluate the performance of IGT on the e-commerce platform for one week. We split out the entire user requests by 1/10 to test our model. The MAE, MAPE and MARE are 9.86 hours, 14.36% and 15.83%, respectively. Compared with Table 2, the national level evaluation performance is between that of D1 and D2. The results indicate cities with a larger population usually have lower MAE on ETA prediction. Compared with the original served model, our model can reduce the average predictive error (*i.e.*, MAE) by 15%.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose an inductive graph transformer (IGT) to estimate the package delivery time. We use a decoupled architecture to capture both the high-order interactions of orders via a temporal and heterogeneous GCN and the interactions of elements in an order through a transformer-based model. The information of unseen nodes can be learnt by propagating messages from neighboring nodes in our model. Hence, both structural information and raw features can be utilized in the transformer-based model for accurate ETA prediction. In our model, IGT, we simplify GCN by removing both its non-linear layer and the transformation matrix. This allows industrial scenarios to apply IGT on large-scale datasets. Experimental results on two large-scale offline logistics datasets demonstrate the effectiveness of IGT in both inductive and transductive learning. We further evaluate the performance of the model in a national level for online test. IGT is promising in enhancing the user experiences of billions of customers in deployment on e-commerce platforms. For future work, we will focus on providing confidence intervals on ETA prediction instead of a point estimation.

ACKNOWLEDGMENTS

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* 785–794.

[2] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. 2020. Skeleton-based action recognition with shift graph convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 183–192.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 1724–1734.

[4] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal neighbourhood aggregation for graph nets. In *Proceedings of the 34th International Conference on Neural Information Processing Systems.*

[5] Ruomeng Cui, Zhikun Lu, Tianshu Sun, and Joseph Golden. 2020. Sooner or later? Promising delivery speed in online retail. *Promising Delivery Speed in Online Retail* (2020).

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics.*

[8] Malik Doole, Joost Ellerbroek, and Jacco Hoekstra. 2020. Estimation of traffic density from drone-based delivery in very low level urban airspace. *Journal of Air Transport Management* 88 (2020), 101862.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations.*

[10] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699* (2020).

[11] 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 *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2697–2705.

[12] Kun Fu, Fanlin Meng, Jieping Ye, and Zheng Wang. 2020. Compacteta: A fast inference system for travel time prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 3337–3345.

[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, 249–256.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 1025–1035.

[15] Huiting Hong, Yucheng Lin, Xiaoqing Yang, Zang Li, Kung Fu, Zheng Wang, Xiaohu Qie, and Jieping Ye. 2020. HetETA: Heterogeneous Information Network Embedding for Estimating Time of Arrival. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* Association for Computing Machinery, New York, NY, USA, 2444–2454.

[16] Jilin Hu, Bin Yang, Chenjuan Guo, Christian S Jensen, and Hui Xiong. 2020. Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE).* IEEE, 1417–1428.

[17] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020.* 2704–2710.

[18] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation Learning for Dynamic Graphs: A Survey. *Journal of Machine Learning Research* 21, 70 (2020), 1–73.

[19] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations.*

[20] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations.*

[21] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence.*

[22] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-Task Representation Learning for Travel Time Estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* Association for Computing Machinery, New York, NY, USA, 1695–1704.

[23] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1754–1763.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[26] Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. 2021. Hetegcn: heterogeneous graph convolutional networks for text classification. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining.* 860–868.

[27] Yuehuan Su, Huabo Duan, Zinuo Wang, Guanghan Song, Peng Kang, and Dongjie Chen. 2020. Characterizing the environmental impact of packaging materials for express delivery via life cycle assessment. *Journal of Cleaner Production* 274 (2020), 122961.

[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 *Advances in neural information processing systems.* 5998–6008.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the Sixth International Conference on Learning Representations.*

[30] Zwass Vladimir. 1996. Electronic commerce: structures and issues. *International journal of electronic commerce* 1, 1 (1996), 3–23.

[31] 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. *ACM Trans. Intell. Syst. Technol.* 10, 2 (2019).

[32] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. 2019. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Companion Proceedings of The 2019 World Wide Web Conference.* 310–316.

[33] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to estimate the travel time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 858–866.

[34] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning.* PMLR, 6861–6871.

[35] Fan Wu and Lixia Wu. 2019. Deepeta: A spatial-temporal sequential neural network model for estimating time of arrival in package delivery system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 774–781.

[36] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 7370–7377.

[37] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *Advances in Neural Information Processing Systems* 32 (2019), 11983–11993.

[38] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 401–411.

[39] Xin Zhou. 2022. A Tale of Two Graphs: Freezing and Denoising Graph Structures for Multimodal Recommendation. *arXiv preprint arXiv:2211.06924* (2022).

[40] Xin Zhou, Donghui Lin, Yong Liu, and Chunyan Miao. 2022. Layer-refined Graph Convolutional Networks for Recommendation. *arXiv preprint arXiv:2207.11088* (2022).

[41] Xin Zhou, Aixin Sun, Yong Liu, Jie Zhang, and Chunyan Miao. 2021. SelfCF: A Simple Framework for Self-supervised Collaborative Filtering. *arXiv preprint arXiv:2107.03019* (2021).

[42] Xin Zhou, Hongyu Zhou, Yong Liu, Zhiwei Zeng, Chunyan Miao, Pengwei Wang, Yuan You, and Feijun Jiang. 2022. Bootstrap Latent Representations for Multimodal Recommendation. *arXiv preprint arXiv:2207.05969* (2022).

[43] Lin Zhu, Wei Yu, Kairong Zhou, Xing Wang, Wenxing Feng, Pengyu Wang, Ning Chen, and Pei Lee. 2020. Order Fulfillment Cycle Time Estimation for On-Demand Food Delivery. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2571–2580.

[44] Zhihua Zhu, Xinxin Fan, Xiaokai Chu, and Jingping Bi. 2020. Hgcn: A heterogeneous graph convolutional network-based deep learning model toward collective classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1161–1171.