Lifelong Property Price Prediction: A Case Study for the Toronto Real Estate Market

Hao Peng, *Member, IEEE*, Jianxin Li, *Member, IEEE*, Zheng Wang, Renyu Yang, *Member, IEEE*, Mingsheng Liu, Mingming Zhang, Philip S. Yu, *Fellow, IEEE*, and Lifang He, *Member, IEEE*

Abstract—We present LUCE, the first life-long predictive model for automated property valuation. LUCE addresses two critical issues of property valuation: the lack of recent sold prices and the sparsity of house data. It is designed to operate on a limited volume of recent house transaction data. As a departure from prior work, LUCE organizes the house data in a heterogeneous information network (HIN) where graph nodes are house entities and attributes that are important for house price valuation. We employ a Graph Convolutional Network (GCN) to extract the spatial information from the HIN for house-related data like geographical locations, and then use a Long Short Term Memory (LSTM) network to model the temporal dependencies for house transaction data over time. Unlike prior work, LUCE can make effective use of the limited house transactions data in the past few months to update valuation information for all house entities within the HIN. By providing a complete and up-to-date house valuation dataset, LUCE thus massively simplifies the downstream valuation task for the targeting properties. We demonstrate the benefit of LUCE by applying it to large, real-life datasets obtained from the Toronto real estate market. Extensive experimental results show that LUCE not only significantly outperforms prior property valuation methods but also often reaches and sometimes exceeds the valuation accuracy given by independent experts when using the actual realization price as the ground truth.

Index Terms—Heterogeneous information network, graph neural network, LSTM, lifelong learning, house price prediction.

✦

1 INTRODUCTION

For many families, a house is their most valuable asset. Accurate and up-to-date house[1] valuation is vital for various real-estate stakeholders such as homeowners, buyers, mortgage lenders, agents, etc. House price estimation is traditionally performed by a real estate appraisal based on expert knowledge of target property, surrounding areas and historical data [1], though at a very coarse granularity. Substantial efforts – most notably regression-based methods [2], [3], [4], [5] – have been devoted to automate the house valuation by primarily examining the relationship between the house price and a range of quantified features like the property size, interior decoration, the number of bedrooms and facilities, the distance to a school catchment, etc.

Unfortunately, existing approaches for property valu-

- Hao Peng is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China, and is with the Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China. E-mail: penghao@act.buaa.edu.cn.
- Jianxin Li is with Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China. E-mail: lijx@act.buaa.edu.cn.
- Zheng Wang and Renyu Yang are with the School of Computing, University of Leeds, Leeds LS2 9JT, UK. E-mail: {z.wang5, r.yang1}@leeds.ac.uk.
- Mingsheng Liu is with the Shijiazhuang Institute of Railway Technology, Shijiazhuang 050041, China. E-mail: liums601001@sina.com.
- Mingming Zhang is with the UrBrain Technology, Canada. E-mail: zmm021@gmail.com.
- Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago 60607, USA. E-mail: psyu@uic.edu.
- Lifang He is with the Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA 18015 USA. E-mail: lih319@lehigh.edu.

1. In this work, a house is referred to as different types of residential properties, including the traditional house and apartments (or flats).

ation are inadequate in tackling two fundamental issues manifested by real-life property markets: data *freshness* and *sparsity*. The key challenge here is that house transaction data are rarely up-to-date and inherently sparse - there is typically a gap of years between two transactions of a property and only a small number of houses are on the market for any given time. For example, our analysis on the residential property transaction data of the Toronto Region in Canada between 2000 and 2019[2] shows that two consecutive transactions of a house typically spans over decades and only 0.1% to 0.5% of the residential properties within an administrative district (known as a neighborhood or community)[3] were traded within 12-month time frame. Moreover, the small number of freshly traded houses spread across a large demographic area across thousands of households, making it difficult to effectively model and reason about the relationships between traded houses. On top of that, transaction data before 2000 were often not in a digital form, which further reduces the availability of house transaction data. The lack of current house transaction data implicates much of the pricing information that prior approaches rely upon cannot accurately reflect the market values of the target houses. Given a complex and dynamic real estate market, the discontinuity and sparsity of house transactions make it extremely intricate to build an accurate predictor for house valuation.

To address the above limitations, we present LUCE[4], a novel learning framework for lifelong house price predic-

2. http://trreb.ca/
3. We consider 140 neighbourhoods (also often referred to as communities in the Canadian real estate market) officially recognized by the City of Toronto.
4. LUCE = **L**ifelong ho**u**se pri**ce** pr**e**diction.

tion. LUCE is designed to work on a limited set of current house transaction data. Our key insight is to use the most recent house transactions to estimate the value of all other properties of the target region (e.g., a metropolitan city). By periodically updating and estimating the house information, LUCE offers a life-long learning framework to estimate the current values for all houses with a metropolitan area. By so doing, LUCE enables the downstream house valuation model to utilize a significantly more substantial amount of house transaction data across many properties than what are available to prior methods. The completeness and up-to-date house information provided by LUCE thus enables one to build an accurate downstream valuation model using standard machine learning techniques.

Translating our high-level idea to build a practical system is, however, non-trivial. Since LUCE has to rely on a small number (i.e., data sparsity) of recent house transaction data across properties spreading across a large geographical area, it is important to make best use of all available data. However, doing so is challenging because houses are distributed over a large geographical region with many attributes that can affect the house valuation. To this end, we adopt the heterogeneous information network (HIN) to model the relationships - such as the location, facility, or floorplan - between houses entities. We then employ a Graph Convolutional Network (GCN) to learn the house data representation, which is fed into a property valuation model built upon a standard multilayer perceptron (MLP) model. Instead of directly performing learning on the entire, large HIN - which would be hard to generalize - we break down transactions into slices according to the geographical region of the traded house and when the transaction was taken place (on a monthly scale in this work). This allows us to partition a large HIN into smaller sub-graphs so that the learning of house data representation can be performed on smaller graphs in parallel. We go further by feeding the graph embeddings to a Long Short Term Memory (LSTM) network to improve the learned house representation by learning the temporal dependence of house data over time.

To address the discontinuity in house transaction data, we use the GCN-LSTM unit to perform house valuation of all house entities in the HIN over the last few months and then use the prediction and transaction history to estimate the price of the target house for the current month. We show that this lifelong learning framework can be achieved by simply stacking up a sequence of GCN-LSTM learning units. To overcome the gradient vanishing issue when performing learning over a long sequence of network layers, we introduce a sliding recursive parameter updating strategy to navigate the depth of gradient back-propagation and employ reinforcement learning to automate the parameter settings in the loss function calibration. Our evaluation shows that this approach is simple to implement but yields good prediction performance. We evaluate LUCE by applying it to a real-world dataset collected from the Toronto real estate market. We compare LUCE against 4 state-of-the-art automated house valuation methods and the valuation given by independent experts. When using the realization price as the ground truth, LUCE outperforms all prior methods and often expert valuations.

This paper makes the following contributions. It is the first to:

- adopt an HIN to model the house transaction data (§2);
- propose a novel lifelong learning framework to perform property valuation (§4 and §5);
- outperform prior automated house valuation methods and even expert property valuation on real-life datasets (§6).

To enable replication and foster research we make our LUCE publicly available at: `https://github.com/RingBDStack/LUCE`.

2 BACKGROUND AND MOTIVATION

In this section, we mainly outline the problem scope and motivation, followed by a detailed description of the data.

2.1 Problem Scope and Motivation

Problem definition. This work focuses on residential property valuation (price prediction). The prediction employs property specific information and transaction records to automatically estimate future property prices. Our work addresses two primary research challenges facing the house valuation – *data sparsity* and *data freshness* issues that manifest *spatially* and *temporally*. This is because only a small portion of houses is traded annually, while owner changing does not frequently manifest for most houses, resulting in a lack of up-to-date house transaction information temporally. Another challenge derives from the long-term learning wherein the vanishing gradients [6] and catastrophic forgetting [7] effects of neural networks inevitably exhibit. Therefore, the learning model continuously learns on short-term dependencies but be lifelong so that we can replace the missing transaction information in the property network with the estimated values. Formally, we aim to learn a model that takes as input house features $\mathcal{X}$ and its previous sold price $\mathcal{Y}$, and house-related properties to predict the current valuation, y_t, of the target house h at time t.

Motivation. Our work is motivated by the observation that prior regression-based work is insufficient to tackle data sparsity over time and stale transaction data cannot reflect realistic property prices. To illustrate this point, consider Table 1 that gives the precision of different prediction approaches, Decision Tree Regression (DT), Support Vector Regression (SVR), and LSTM against the realization price for the residential property transaction data of Toronto between 2000 and 2019. Regressors like SVR have a much higher Root Mean Square Error (RMSE) than the valuation given by human experts. Due to the intrinsic affinity among houses with similar terms of geographic location and floorplan design, we consider the price prediction as a regression problem based on nodes (houses) in a graph, where labels (i.e., house price) are only available for a small subset of nodes. We consider this problem as a life-long *semi-supervised* learning based on graph embeddings.

Table 1: RMSE Comparison.

#House Trans.	SVR	DT	LSTM-D	Appraiser-based
10,000	0.2677	0.2616	0.3055	0.1417
30,000	0.2792	0.2722	0.3047	0.1339
50,000	0.2885	0.2889	0.3076	0.1331
70,000	0.2987	0.2902	0.3021	0.1431
90,000	0.3011	0.2994	0.3015	0.1378

Table 2: Facility features and their types.

Category	Type	Abbr.	Type	Abbr.
Building type	Townhouse	TH	Detach	DE
	Semi-detach	SDE	Duplex	DU
	Triplex	TR	Fourplex	FO
	Cottage	CO	Link	LI
	RuralResid	RR	Other	OT
Layout structure	Backsplit	BA	Bungalow	BU
	OneNHalfStorey	ONS	TwoNHalfStorey	TNS
	TwoStorey	TWS	ThreeStorey	THS
	Sidesplit	SS	Other	OT
Garage type	Attach	A	Builtin	B
	Carpor	C	Detach	D
Exterior wall	AlumSliding	AS	Brick	BR
	Concrete	CO	MetalNSide	MS
	Shingle	SH	Stone	ST
	VinylSling	VS	Wood	WO
Pool type	AboveGround	AG	Indoor	ID
	Inground	IG	None	NO
Heat source	Electricity	EL	Gas	GA
	Oil	OI	Other	OT
Heat equipment	Baseboard	BB	FanCoil	FC
	ForcedAir	FA	HeatPump	HP
	Radiant	RA	Water	WA
Basement	Crawl Space	CS	Step Entrance	SE
	Full	FU	Half	HA
	Finished	FI	Unfinished	UF
	Part Finished	PF	None	NO

2.2 Data Landscape

While generally applicable, to have a realistic use case, our approach uses the house transaction data of the Region of Toronto between 2000 and 2019. This dataset is owned by the Toronto Real Estate Board, an online information portal for real estate listings and services in the Greater Toronto area. The dataset consists of over a million transaction records of residential properties. Assumably, we can access the limited up-to-date information including the property size and floorplan because such information is often required to be supplied by the vendor to a real estate agent or lender. More information on the dataset can be found at §6.1. As depicted in Fig. 1, we categorize the features into four distinct aspects:

i) Geographical information: In our case study, we break down a valid house location by a series of address elements. These elements can be formatted as a top-down hierarchical tree that encompasses different levels of geographical units. As depicted in Fig 1, the top of the tree is the largest and root geographical unit – city-scale municipality (M) area. Further down the tree, we sequentially define the geographical units as: the community (C) (i.e., neighbour in the Canadian system), the forward (F) sortation area (FSA)[5], and the postal (P). Herein, the edge in the tree denotes the *belongs-to* relationship between layers. The geographical unit allows us to capture information like school catchments which are typically allocated through the geographical unit.

ii) Facilities information (enumerated): Key indicators of the property valuation typically encompass supporting facilities

5. A forward sortation area (FSA) is a geographical unit based on the first three characters in a Canadian postcode. All postcodes that start with the same three characters - for example, M4B - are together considered an FSA.
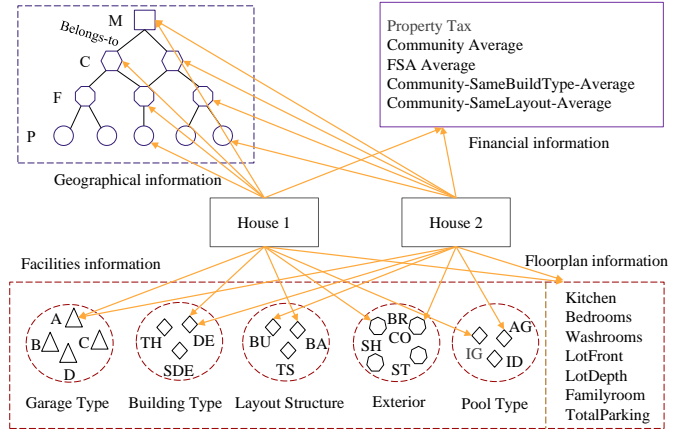

Figure 1: House data and its HIN representation.

(e.g., the type of garage, layout structure, etc.). Their types are *enumerated* and summarized in Table 2.

iii) Floorplan information (numerical): The number of various rooms (bedroom, washroom, family room, kitchen, basement), the house area, width and depth of house land, and the number of stove, air conditioning and parking slots.

iv) Financial information (numerical): the property tax, and other financial information (excluding the property price to be predicted) associated with the geographical information, including the average up-to-date property price of the community and the FSA, and the average price of properties of the same type within the community and the FSA.

3 OUR APPROACH OF LUCE

3.1 Overview

The first innovation of our approach, as a departure from prior work, is to encode the information of house data as a structured, heterogeneous information network (HIN)[6] wherein HIN nodes are different types of entities of houses and their characteristics, while edges represent different relationships between a pair of entities (e.g., a house *belongs-to* a community/neighborhood, or a house has detached garage). HIN has several advantages compared against the native approaches that only use raw features – (1) *higher accuracy*: By using HIN that contains abundant node and relationship information, the meta paths can provide rich semantics and indicate higher-order relationships. One can exploit such semantic information between nodes to embed the nodes into representation vectors, far more accurately than directly adopting house raw features. (2) *better interpretability:* Meaningful meta-paths and meta-graphs can indicate semantic-explainable similarities between two houses. (3) *Coherent connection with graph convolutions:* Since the problem of property price prediction is house entity oriented only, it is also effective enough to deduce the information from a self-contained HIN to a homogeneous graph that can be directly absorbed by the graph convolutions.

In this context, the fundamental requirements of graph embedding for the HIN consists of three crucial elements –

6. We refer to the same concept and definition about Heterogeneous Information Networks in previous work [8], [9], [10].
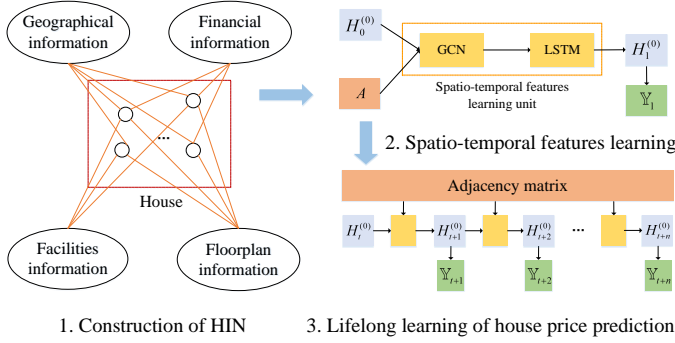
Figure 2: Overview of LUCE.

Table 3: Notations.

Symbol	Definition
$\mathcal{G}; \mathcal{V}; \mathcal{E}$	HIN and its node sets and edge sets
$\mathcal{X}$	House features set
$\mathcal{Y}$	Node label (house sold price) set
$t; T$	Time step number; Total time steps (months)
$Sim(v_i, v_j)$	The similarity of house v_i and house v_j
A	Adjacency matrix based on house similarity
$\bar{\omega}$	The weight of meta-path or meta-graph
$\mathcal{M}'$	The total number of meta-paths and meta-graphs
$\mathbb{L}_t$	Loss function of at time step t
y_t	The sold-price set of house at time step t
$H_{(i,t)}$	The house embedding generated by i-th GCN at time step t
$\mathbb{H}_t$	The house embedding at time step t
$\mathbb{Y}_t$	The predicted prices of all houses at time step t
$\mathcal{W}^{(l)}_{(i,t)}$	The parameter matrix of l-th layer of i-th GCN at time step t
$\theta_{(i,t)}$	The parameter of LSTM of i-th subgraph at time step t

obtaining graph structure and retaining node attributes (numerical features) and node labels. In fact, the graph structure reflects the *structural connectivity* between two house entities, based on the affinity in terms of geographic proximity and pertaining facilities. Meanwhile, numerical features of an individual house regarding the detailed floorplan information and financial information should be maintained as the initial node attribute while property price is regarded as node label. To simplify the lifelong house price prediction problem, especially at the metropolitan level, we did not discretize real values into buckets and transform them as graph nodes. In fact, transforming all numerical values into enumerated values would result in the degradation of prediction accuracy due to the loss of accuracy. Additionally, the financial information is more dynamic compared with other features such as layout structure, building type, etc., and thus more suitable to be numerical.

At the core of LUCE is a deep neural network that builds upon the GCN and LSTM. The network learns the most appropriate embedding for the structural and numerical features captured by the HIN and uses the learned representation and known labels to perform the price regression. Most notably, the semi-supervised GCN allows for feature learning for all houses through a limited number of available transaction sourcing from the sparse houses. Herein, we use the native GCN as the representative instance – due to its simplicity and general purpose use cases – while any other latest semi-supervised graph neural network models [11], [12] can be easily used as a substitute for the GCN in our scenario of house feature learning. Similarly, we also employ the widely-used recurrent neural network model LSTM, due to its general purpose use cases. Other recurrent neural feature learning models such as [13] can replace the LSTM and be integrated with the current framework for an improved prediction.

It is worth noting that, unlike prior work that directly operates on the discrete and sparse time-series data across years, LUCE splits available transactions into monthly slices and uses data within each month slice to train the GCN-LSTM continuum, the basic feature learning unit. This is because the house price is observably stable between two consecutive months and we can constantly update the prediction model on a monthly basis. Accordingly, the trained network manages to predict *all* house prices in the coming months. In addition, the contribution of different meta-

paths and meta-graphs in the HIN can be learned. To avoid out-of-memory, we can naturally divide the graph of houses into subgraphs, jointly connected by houses within the overlapping areas. Correspondingly, we break down the basic GCN-LSTM training unit into independent and parallel GCN-LSTM instances, each of which is exploited for feature embedding in each subgraph. Such parallelism will finally form an array of GCN-LSTM, thereby significantly speeding up the procedure of feature learning.

3.2 Architecture and Pipeline

Fig. 2 depicts the overall architecture of LUCE. From the constructed HIN, we firstly calculate the adjacency matrix – the best option to reflect the proximity and the node connectivity in the graph – and the attribute matrix to retain the residual numerical features of nodes. Due to the intrinsic fact that GCNs merely operate on homogeneous graph and induce embedding vectors for nodes based on the properties of their neighborhoods, we compute the similarity between every pair of houses and store it with adjacency matrix, to underpin the heterogeneous graph convolution (§4.1).

We feed adjacency matrix A – generated by both meta-paths instances and meta-graphs instances based similarity measurement – together with the house (or node) into a GCN and a graph LSTM. The learned embedding vectors delivered by the GCN-LSTM continuum are therefore concatenated to form the holistic representation of the original HIN at time t (§4.2).
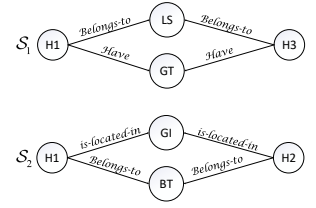
To build a lifelong prediction framework capable of estimating house price monthly, we design a multitask learning scheme where GCN-LSTM units are unfolded multiple times in a pipeline. A network obtained at month t can be inherited for predicting the house prices $\mathbb{Y}_t$ in the coming month $t + 1$. Meanwhile, the prediction will also update the embedding of houses $\mathbb{H}_{t+1}$ at time $t + 1$, which is used to train the follow-up GCN-LSTM units. We iterate this process until targeting the valuation of all houses at a certain month, $t + n$. To deal with the gradient vanishing problem, we introduce a sliding recursive strategy for parameter updating – limiting the depth the gradient can back-propagate and recognizing the different impact of the embedding within each time on the loss function calibration. We use reinforcement learning to auto-learn the parameters involved in the calibration (§5). To aid discussion, Table 3 depicts the notations used throughout the paper.
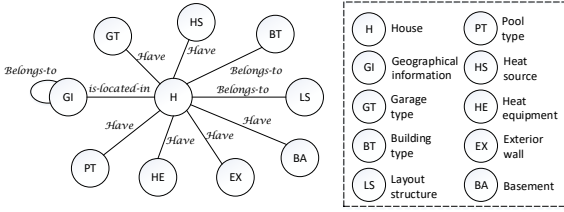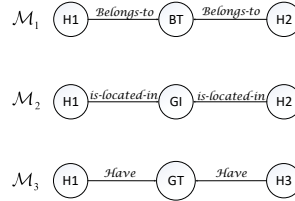
(a) Meta-schema of HIN. (b) Examples of meta-paths. (c) Examples of meta-graphs.

Figure 3: (a) Meta-schema denotes basic relationships among entities in HIN; (b) a meta-path encodes a common relationship/feature shared by two entities; (c) a meta-graph encodes multiple relationships shared by a pair of entities.

4 TEMPORAL-AWARE NETWORK

In this section, we first give the detailed construction process of the graph-of-house and then introduce the temporal-aware feature learning for the house price prediction.

4.1 Graph-of-house Construction from HIN

Calculating meta-path and meta-graph. Meta-schema is a meta-level template that defines the relationship and type constraints of nodes and edges in the HIN. As shown in Fig. 3a, we obtain a meta-schema that encodes all possible relationships between the house entity and other types of entities. *Meta-path* is a path that connects a pair of network nodes with a semantically meaningful relationship between nodes (exemplified in Fig. 3b). We can enumerate all existing relationships among each pair of house entities as the pre-defined meta-paths. In fact, a meta-path can be used to encode common features shared by two houses, e.g., two houses belong to the same community. As a pair of houses could have an arbitrary number of meta-paths, *meta-graph*, in the form of directed acyclic graph (DAG), can be used as a template to capture the arbitrary but meaningful combination of existing meta relationships between a pair of nodes. For instance, the meta-graphs described in Fig. 3c define two templates – houses have the same layout and garage type (above) and houses located in the same area have the same building type (below).

Retrieving structural information. Meta-paths and meta-graphs over types and structures indicate semantic-explainable similarities between two houses – houses have more meta-path instances and meta-graph instances tend to have closer valuation. In addition, different meta-graphs should be arguably differentiated when computing the similarity according to the differed semantic implications in meta-paths. For example, a house within the same postal area with the same layout structure as house x is far more likely to have similar valuation compared against another house that has the same building and pool style as house x but in a different community.

Following similar methodology presented in [14], [15], we compute the similarity of house valuation between two houses h_i and h_j as follows:

$$S(h_i, h_j) = \sum_{m=1}^{M'} \omega_m \frac{2 \times Count_{C_m}(h_i, h_j)}{Count_{C_m}(h_i, h_i) + Count_{C_m}(h_j, h_j)}, \quad (1)$$

where C denotes the collection of meta-paths and meta-graphs, and $Count_{C_m}(h_i, h_j)$ counts the number of m-th

element (path/graph) C_m between two house instances h_i and h_j. Similarly, $Count_{C_m}(h_i, h_i)$ and $Count_{C_m}(h_j, h_j)$ compute the number of meta-paths and meta-graphs instances between h_i and h_i, and between h_j and h_j, respectively. Intuitively, the more houses are connected via the meta-paths or meta-graphs, the higher the similarity. The number of meta-graph instances is counted by the *Hadamard product* between matrixes counted by sub-string meta-paths. At the core of the similarity function, $S(h_i, h_j)$, is to normalize the importance of meta-paths and meta-graphs between h_i and h_j by applying different weights to different structural relationships. $2 \times Count_{C_m}(h_i, h_j)$, counts the number of shared meta-path instances and meta-graph instances between house instances h_i and h_j for computing the semantically overlapped information. We multiply the number by two because the meta-paths and meta-graphs are bi-directional. $Count_{C_m}(h_i, h_i) + Count_{C_m}(h_j, h_j)$ counts the total number of meta-paths and meta-graphs among the two house instances themselves. Notably, we use a *learnable* parameter vector $\vec{\omega} = [\omega_1, \omega_2, \ldots, \omega_{M'}]$ to denote weights of all meta-paths and meta-graphs.

We can then use the calculated similarity to indicate the connectivity between any pair of house instances. Accordingly, we construct an $N \times N$ weighted adjacency matrix A to store the semantic similarity among N houses. The adjacency matrix A is also referred to as the graph-of-house. The weighted mechanism can ensure that the aggregation of GCN is not susceptible to the skewed distribution of entity links, as the weights are learnable and hence adaptive to changes in the graph.

Retrieving house attribute matrix through PCA. There are several house-related numerical attributes. We use one-hot representation to encode each of these numerical attributes, after which we concatenate them as a single vector of numerical values. An attribute vector is therefore associated with a house entity in the HIN. We further apply principal component analysis (PCA) to reduce the dimensionality of the vector to D (e.g., to 100 elements). In this manner, we eventually form the house attribute matrix X, which is of shape $N \times D$.

4.2 Temporal-Aware Feature Learning

Fig. 4 illustrates the flowchart of LUCE learning framework. The trunk of LUCE is to take as input the weighted adjacency matrix A and the house attribute matrix X, and constantly update the house embedding $\mathbb{H}_t$ at the t-th month. The house price $\mathbb{Y}_t$ can be then predicted based on the embedded features with known *price label*.
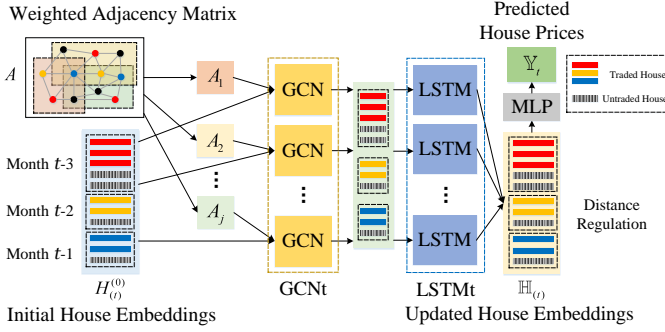
Figure 4: Basic unit of feature learning.

At the core of the graph embedding is learning the representation of house features. We leverage GCN to aggregate the neighborhood information in A when measuring the relationship between an entity pair. Primarily exploiting the weighted adjacency matrix and attribute matrix, GCN learns and feeds the feature representation to a LSTM to involve temporal dependency within the HIN and further calibrate the effectiveness of the graph embedding. We introduce parallelism to accelerate the model training and reduce the memory overhead when processing a large graph. As houses have been intrinsically divided into adjacent geographic areas, we split the holistic graph into several overlapping subgraphs, and conduct feature learning for each subgraph in parallel. Specifically, we divide the weighted adjacency matrix A into several overlapping subgraphs (see Fig. 4) and it can be formalized as:

$$A = A_1 \cup A_2 \cdots \cup A_j, \tag{2}$$

where j is the total number of divided subgraphs.

Feature learning. For the i-th subgraph A_i, we employ a GCN model [16] to learn the numerical feature embedding on a monthly basis, by formalizing a layer-wise propagation rule at the t-th month:

$$H_{(i,t)}^{(l)} = GCN(A_i, H_{(i,t)}^{(0)}, W_{(i,t)}^{(l)}), \tag{3}$$

where $H_{(i,t)}^{(0)}$ is the afferent feature matrix of the i-th subgraph, and $W_{(i,t)}^{(l)}$ is the parameter matrix of the i-th subgraph at l-th layer. $H_{(i,1)}^{(0)}$ is set to be the initial house attribute of the i-th sub-graphs from X. After training, we record the l-th layer embedding of the i-th subgraph at t-th month as $\mathcal{H}_{(i,t)} = H_{(i,t)}^{(l)}$. In fact, the adjacency matrix A_i contains the parameter $\vec{\omega}$ pertaining to each meta path/graph in the similarity measurement. Notably, the parameter will be continuously and globally updated in the training procedure.

This semi-supervised GCN technology allows for learning embedded features of all houses using a limited number of available house transactions. Meanwhile, it enables us to learn the contribution of different meta-paths and meta-graphs, which can facilitate to divide house instances into independent and parallel GCNs. Still, it is intractable how to leverage house transactions at different time (e.g., different months) to calibrate the house embedding more precisely. To tackle this, we will work on the temporal features at different time periods.

Temporal dependency. House embedding obtained from the GCN cannot guarantee an up-to-date price information due to the ignorance of time difference in the price label. We therefore add an additional LSTM layer to learn and update the valuation for each house. Specifically, we feed the learned house embedding $\mathcal{H}_{(i,t)}$ of the i-th subgraph at t-th month as the input of LSTM units. The output of the LSTM can be formalized as:

$$\mathbb{H}_{(i,t)} = LSTM(\mathcal{H}_{(i,t)}, \theta_{(i,t)}), \quad H_{(i,t+1)}^{(0)} = \mathbb{H}_{(i,t)} \tag{4}$$

where $\theta_{(i,t)}$ means parameters in LSTM unit and the output of LSTM unit will be passed to next GCN unit as the initial house attribute. Consequently, the feature embedding of houses are transformed into time series according to the transaction times, significantly alleviating the discontinuity of house transactions in the short run. We then concatenate all individual embeddings $\mathbb{H}_{(i,t)}$ into $\mathbb{H}_{(t)}$ before making fine-grained calibration. Hence, for the month t, by using the transaction prices of houses in the HIN, we can train evolving house embeddings, which integrate both spatial and temporal features. Eventually, we add a multi-layer perceptron (MLP) between the delivered embedding by LSTM and the price label to decode and thus predict any house prices $\mathbb{Y}_{(t)}$, at t: $\mathbb{Y}_t = MLP(\mathbb{H}_{(t)})$.

Calibration through distance regulation. We instantiate an independent GCN for each subgraph to obtain its own feature embedding in parallel. To coordinate those embedding results and form a holistic picture, we use *distance regulation* to calibrate the embedding, ensuring the house across different subgraphs has close embedding scheme in different GCNs:

$$\epsilon(P_t) = \sum_{\alpha=1}^{L} ||\mathbb{H}_{(1,t)}(p_\alpha) - \sum_{\beta=2}^{g(p_\alpha)} \frac{1}{g(p_\alpha) - 1} \mathbb{H}_{(\beta,t)}(p_\alpha)||, \tag{5}$$

where $\mathcal{P}$ denotes the set of those overlapping houses, i.e., $\mathcal{P} = \{p_\alpha, \alpha = 1, \ldots, L\}$. $g(p_\alpha)$ means the number of subgraphs that contain the p_α house, and $\mathbb{H}_{(\beta,t)}(p_\alpha)$ refers to the embedding of the p_α house in the β-th GCN at the t-th month.

Loss Function. We use the following loss function to optimize model parameters:

$$\mathcal{L}_t = \mathcal{R}_t + \epsilon(P_t). \tag{6}$$

The loss function $\mathcal{L}_t$ comprises two parts – the accuracy of prediction – the Root Mean Square Error (RMSE) $\mathcal{R}_t$ – and the unified embedding of overlapping houses among GCNs. The widely-used stochastic gradient descent (SGD) method is used to update all parameters including $W_t, \theta_t, \vec{\omega}$, etc.

However, the proposed feature embedding is effective in case of short term transactions but is extremely susceptible to longer series spanning many years. In addition, the model training would become very slow and difficult to achieve convergence, resulting in the failure of feature embedding.

5 LIFELONG LEARNING NETWORK

In this section, we first present the loss function of the lifelong learning model and then how reinforcement learning is used for optimizing the loss function.
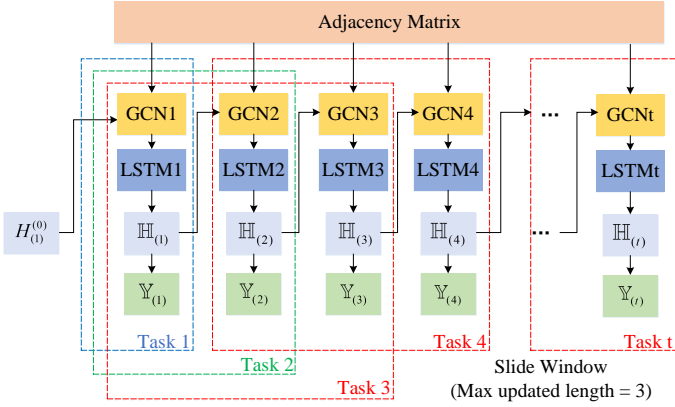
Figure 5: An instance of the structure of LUCE.

5.1 Basic Model

Unlike previous mainstream models of time series prediction that calculate the loss function and update parameters until last time step, our proposed lifetime learning approach will update parameters at each month. Therefore, we divide house price set according to monthly-based subsets. y_t is the t-th set corresponding to embedded representation $\mathbb{H}_t$ of the house, and the overall time series is $\mathcal{Y} = \cup_{t=1}^T y_t$. We record the number of traded houses with price labels each month as N_t. The RMSE can be formalized as:

$$\mathcal{R}_t = \sqrt{\frac{1}{N_t} \sum_{j=1}^{N_t} (\hat{y}_t(j) - y_t(j))^2}, \quad (7)$$

where $\hat{y}_t(j)$ and $y_t(j)$ refer to the predicted price and sold price of j-th house, respectively.

In the lifelong learning network, we introduce a sliding recursive parameter updating strategy to navigate the depth of gradient back-propagation, thereby mitigating the gradient vanishing problem. We take the evolving embedding $\mathbb{H}_t$ of the house learned every month as the initial house attribute of the next month's GCN-LSTM unit. As shown in Fig. 5, we use "$task$" to describe the training goal at various time steps in the lifelong model. At time step t, task t indicates the training procedure based on the n-months ahead of t for the price label prediction at time step $t+1$, to confine the parameter updating for at most n back-propagation depths. For example, if n is 3, within $Task\ 4$, the gradients generated by the objective function $\mathcal{R}_4$ will not only affect W_4 and θ_4, but back-propagate and affect the parameters W_3, θ_3 and W_2, θ_2.

Loss Function Calibration. We differentiate the impacts of tasks on the effectiveness of feature learning. Within each task, the loss function comprises the original loss function with the accumulation stemming from the back-propagation:

$$\mathbb{L}_t = \frac{1}{n} \left(\mathcal{L}_t + \sum_{i=1}^{n-1} \lambda_i \Theta_{t-i} \right), \quad (8)$$

where λ_i denotes the penalty coefficient to depict the impact of prior tasks to the task $t-1$ while Θ_{t-i} indicates the individual propagating loss. To further accelerate the model

training, we adopt *parameter inheritance* in sequential GCN-LSTM units, which ensure the previously delivered house embedding can be initialized in the follow-up task.

As only a small fraction of houses are traded every month and have price labels, when we calculate the monthly RMSE in Eq. 7, only a part of the house is calculated, meaning $\hat{y}_t(j) \in \mathbb{Y}_t, j = 1, 2, \ldots, N_t$. Nevertheless, it is enough to train the proposed lifelong learning model.

5.2 Reinforcement Learning Based Optimization

Since the lifelong framework involves multi-tasks in the training procedure, it is indispensable to fine-tune the relevant coefficients as reasonably as possible. By following works [17], [18], we also employ the reinforcement learning (RL) technique to facilitate the model optimization.

In Eq. 8, we collect the penalty coefficients as $\vec{\lambda} = [\lambda_1, \lambda_2 \ldots \lambda_{n-1}]$. As instinctively transactions long time ago tend to have a decayed impact on the up-to-date price prediction, we believe the λ_i should become smaller when $task\ t-i$ moves away from the current $task\ t$. We therefore use RL wherein the process of finding the minimum training loss is formalized as a Markov Decision Process (MDP) problem, i.e., $\mathcal{M}(\mathcal{S}, \mathcal{A}, P, r)$, where $\mathcal{S}$, $\mathcal{A}$, and P are the state space, action space and state transition model, respectively. r represents the reward function:

- **State space:** The state $s \in \mathcal{S}$ is directly defined as the element in $\vec{\lambda}$: $s = \{\lambda_0, \lambda_1, \cdots, \lambda_m\}$, and $\lambda_i \in [0, 1]$.
- **Action space:** The action $a \in \mathcal{A}$ is used to update the value of $\vec{\lambda}$. Since $\lambda_i \in [0, 1]$, the action $a = (m, \epsilon)$ is to increase or decrease a small value ϵ for the m-th dimension in $\vec{\lambda}$, i.e., $\epsilon \in \{-0.01, 0.01\}$.
- **Reward:** We determine whether $\vec{\lambda}$ is good enough by examining if the training loss calculated by the current $\vec{\lambda}$ can make the model achieve a smaller error on the test set, leading to a large delay in calculating reward r according to the action a. To this end, we design it as a piece-wise function of discrete values, and directly use the discrete reward to update $\vec{\lambda}$ to simplify the RL process. In particular, we formalize the reward $r(s, a)$ as follows:

$$r(s, a) = \begin{cases} +0.01, & if\ RMSE(\hat{y}, y|s) > RMSE(\hat{y}', y|s') \\ -0.01, & if\ RMSE(\hat{y}, y|s) < RMSE(\hat{y}', y|s') \end{cases} \quad (9)$$

where $RMSE(\hat{y}, y|s)$ represents the predicted price error of the model trained based on state s on the test set, and s' indicates the state after s is updated according to action a.

- **State transition:** we determine the next state according to the reward $r(s, a)$ and the current action $a(m, \epsilon)$. If r is positive, then the state will be transitted from state s to s' according to a; otherwise, state s will remain the same.
- **Termination:** We make action a randomly select the dimension m to be updated, then $|\mathcal{A}| = 2m$. If reward pertaining to each action is constantly negative within a range of certain steps, the loss coefficients are considered to be *optimal*.

To facilitate the training of optimal loss coefficients, we, in practice, obtain the predicted price first through training on a small-scale dataset with fixed loss weight, before launching the RL. Once the model turns stable after repeated training iterations, we apply the loss weights directly to a larger dataset.
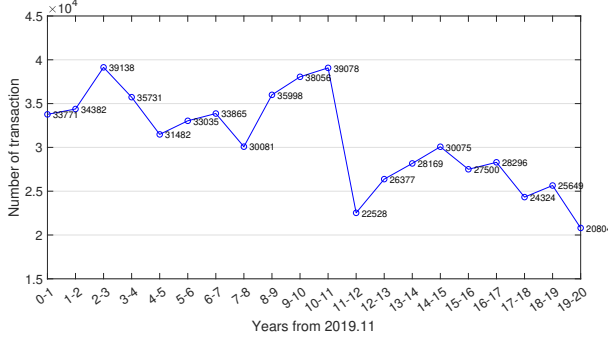
Figure 6: The number of house transactions.

Table 4: Statistics of the three datasets.

Dataset	TorC-H	TorC-A	Tor-H
#House	31,000	37,500	618,339
#Nodes in HIN	155,834	127,955	3,093,180
Time span (months)	31	75	246

6 EVALUATION

6.1 Experiment Setup

Platforms. Evaluation is conducted on a multi-core server with a 64-core Intel Xeon CPU @2.40GHz with 512GB RAM and 8x NVIDIA Tesla P100 GPUs. The server runs Ubuntu 20.04 LTS with Linux kernel 5.4.0. Our model is implemented using Python 3.5.2.

Model parameters. We set the house embedding dimension to $d = 100$. The numbers of enumerated meta-paths and meta-graphs participating in the calculation of house similarity are 30 and 80. We use month as the basic length of time steps to construct the time series. We use Pytorch to implement the networks of the lifelong framework in LUCE. We set 2 layers in GCN unit and we use ReLU as the activation function. The size of hidden layers of LSTM is 128. We adopt Adam optimizer in the back propagation of the network, and the learning rate is set to be 0.001 by default.

Dataset. We sample houses in Toronto Region and construct multiple datasets. We mainly use the following three datasets to evaluate our model:

• **TorC-H:** Sampled house transactions, spanning over 31 months, in a district in Toronto city.

• **TorC-A:** Sampled apartment transactions, spanning over 75 months, in a district in Toronto city. Apartment data has fewer features against houses.

• **Tor-H:** Sampled house transactions in the entire Toronto Region, which cover a wider area and larger-scale data spanning over 63 months compared to the above two datasets. Fig. 6 depicts transaction numbers at different time intervals from November 2019.

Note that our sampling will make each house only have 1 transaction record. Table 4 shows the statistic information of the three datasets. Since TorC-H and TorC-A are relatively small-scale datasets, we mainly use them to learn penalty coefficients $\vec{\lambda}$, the weights of meta-paths and meta-graphs ω_m, and evaluate the effectiveness of LUCE, particularly comparing the predicted price and actual transaction price from 2019-06 to 2019-11. We use the larger Tor-H to evaluate its capability of processing large-scale data.

Evaluation metrics and baselines. We mainly use the following two metrics in the experiments: (i) we use **RMSE** and **MAE** to evaluate the error between the predicted price and the actual sold price of the house; (ii) we define **General Error Rate (GER)** R_{pre} as $R_{pre} = \frac{\hat{y}-y}{y}$, where $\hat{y}$ is the predicted price and y is the actual transaction price. In fact, the GER can better satisfy homeowners, buyers, agents and bank valuation agencies to understand the predicted prices more intuitively.

To evaluate the performance of the proposed LUCE, we consider various of baseline methods as follows:

• **Support vector regression (SVR)**: A classic machine learning algorithm that uses support vector machine (SVM) to fit the dataset for regression analysis. We set the input of the SVR as the initial feature of the houses, and utilize different SVR models to train the houses in different areas.

• **Decision tree (DT)**: A supervised machine learning method. Decision tree has a tree structure wherein each node represents the judgment of the attributes, and each branch represents the output of the judgment result. This is a non-time-series model.

• **Ensemble method (ENS)** [3]: A typical ensemble machine learning method that uses C4.5, RIPPER, Naïve Bayesian, and AdaBoost methods. We set the input of the ENS as the initial feature of the houses.

• **Regression-Particle Swarm Optimization (Reg-PSO)** [5]: A hybrid linear regression prediction model. PSO is used for selection of affected variables and regression analysis is used to determine the optimal coefficient in prediction. We set the input of the Reg-PSO as the initial feature of the houses.

• **Discrete time series based LSTM (LSTM-D)**: LSTM [19] is a time-recurrent neural network model, which is widely used in temporal prediction. Since our housing transactions are not continuous in time, we organize the discrete housing initial features into time series and use them as input to the LSTM.

• **HIN based temporal GCN (HT-GCN)**: An adaptive version from T-GCN [20] where is a spatio-temporal prediction model combining GCN and GRU to extract spatial-temporal features. In our experiment, we use the adjacency matrix A generated by HIN and the initial house attribute matrix X as the inputs to T-GCN.

• **Heterogeneous GCN (H-GCN)**: H-GCN is consistent with the popular GCN unit [16], but only the spatial features of houses in HIN are considered. This is a model of static semi-supervised learning on the constructed HIN.

• **LSTM with H-GCN (HG-LSTM)**: HG-LSTM, consistent with the spatio-temporal feature learning unit mentioned in § 4.2, is used for mining spatio-temporal features based on the constructed HIN. It uses LSTM to explore temporal features of house prices without any lifelong framework. H-CGN provision input features for LSTM.

• **Independent expert valuation (Appraiser)**: We give the estimated price based on the expertise of professional appraisers listed on the Toronto Real Estate Board.

Methodology. We mainly evaluate LUCE in terms of the overall effectiveness and its breakdown stemming from system components and training optimizations.

We firstly evaluate the effectiveness of LUCE on house price prediction by examining the general error rate and the
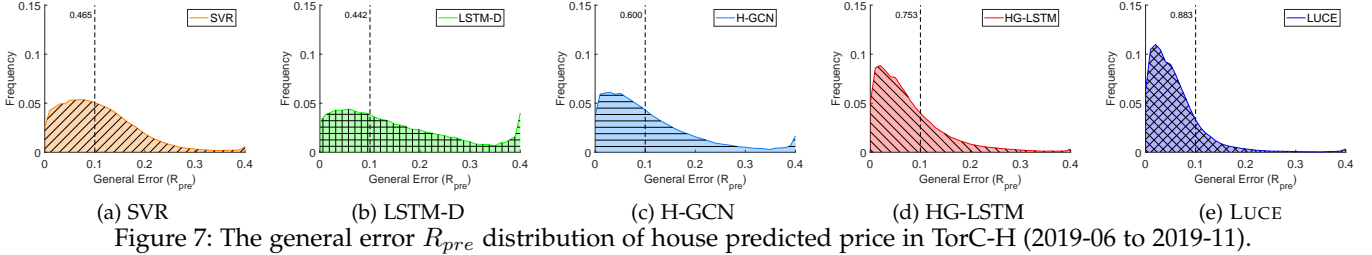
Figure 7: The general error R_{pre} distribution of house predicted price in TorC-H (2019-06 to 2019-11).

average RMSE and MAE of different comparative methods against LUCE (§6.2). We further validate the effectiveness gained from our lifelong learning framework (§6.3). Afterwards, we conduct in-depth investigations into the performance gain harvested from our design optimization and impact of parameters used in the lifelong learning component (§6.4). In addition to these overall evaluation on effectiveness, we evaluate how regularization of training loss and parameter inheritance can boost the training effectiveness and accelerate the model training (§6.5). We also outline our findings of digging and differentiating the importance in various meta-paths and meta-graphs and how they fundamentally underpin the effective feature embedding in GCNs (§6.6). Finally, we evaluate the time consumption of LUCE on three datasets (§6.7).

As some baselines are difficult to deal with large-scale data, we split the large-scale dataset and train multiple identical models on the sub-datasets before merging their prediction results. For LUCE, we make each GCN unit in the model process about 10,000 houses, an empirical node number normally allowed by most GCNs. One can tune the number depending upon the memory capacity on the test server. When utilizing the above baseline methods to predict the transaction price of a month, all transaction prices of its previous months are used as training set to ensure the holistic test fairness.

6.2 Prediction Effectiveness

General error rate. To illustrate the prediction effectiveness using transactions across multiple months, we firstly plot the frequency distribution histogram of GER over a 6-months range from 2019-06 to 2019-11 in dataset TorC-H (containing a total of 6000 house transaction records) under different representative methods[7] in Fig. 7. We set the bin size to 0.01. Obviously, the shape with a left tendency indicates a lower error rate and better prediction effectiveness.

It is observable that time series based approaches including HG-LSTM and LUCE have much lower general error rate against other non-time-series approaches, e.g., H-GCN. Although HG-LSTM takes heterogeneous graph modeling and temporal features into account, its effectiveness is still inferior to LUCE. This is because we merely update parameters that have a more apparent and direct impact in limited depth, while other methods simply update all their parameters. LUCE is also efficient in dealing with the prediction problem in case of long-term sparse transactions. To more precisely, we count the number of houses with GER

7. Since the techniques of some baseline methods used are similar, and the different between their prediction results is small, we select SVR, LSTM-D, H-GCN, HG-LSTM, and LUCE as the representatives.

Table 5: General Error Rate (R_{pre}) Cumulative Probability in Different Methods (2019-11).

| | Methods | GER Value | | | |
		$< 5\%$	$< 10\%$	$< 15\%$	$< 20\%$
TorC-H	SVR	23.72	46.47	66.75	78.50
	DT	29.71	55.60	72.84	83.34
	ENS	26.84	50.61	70.38	80.11
	Reg-PSO	27.91	52.59	71.55	83.56
	LSTM-D	22.95	44.18	61.32	75.89
	Appraiser	46.84	68.61	83.78	91.56
	H-GCN	33.28	60.00	76.48	86.90
	HT-GCN	43.17	73.34	87.20	92.73
	HG-LSTM	47.93	75.28	88.38	94.13
	LUCE	**60.44**	**88.28**	**95.44**	**98.06**
TorC-A	SVR	23.47	44.29	61.89	77.18
	DT	25.83	48.47	66.10	77.67
	ENS	23.67	49.01	63.75	78.90
	Reg-PSO	25.72	50.92	64.61	79.24
	LSTM-D	22.95	43.27	61.64	75.58
	Appraiser	37.80	54.41	70.60	83.66
	H-GCN	31.61	51.20	68.60	84.46
	HT-GCN	42.40	61.83	79.74	93.25
	HG-LSTM	45.62	66.42	84.22	95.47
	LUCE	**52.84**	**70.45**	**90.81**	**97.69**
Tor-H	SVR	19.60	38.36	54.91	67.05
	DT	24.42	44.05	62.48	74.36
	ENS	20.58	39.52	58.71	73.45
	Reg-PSO	23.17	42.46	60.05	75.62
	LSTM-D	27.85	48.58	64.83	76.87
	Appraiser	**48.56**	67.27	87.13	96.60
	H-GCN	27.21	51.16	62.34	75.58
	HT-GCN	41.83	68.92	84.48	93.10
	HG-LSTM	36.04	65.75	83.09	93.01
	LUCE	47.16	**68.07**	**91.62**	**97.38**

lower than 10% under various approaches. LUCE has the highest number – 47.13% and 14.73% more than H-GCN and HG-LSTM, respectively.

To have an in-depth understanding of the effectiveness, we dive into the experiment result of month 2019-11 and Table 5 illustrates the corresponding cumulative proportion of houses with *good prediction* (e.g., with less than 10% GER) in all houses – an important indicator of prediction precision used in real estate field. The specific number indicates the percentage of houses that have GER less than a given threshold. For instance, by using LUCE on TorC-H dataset, 88.28% houses can be *perfectly* predicted the price with less than 10% error rate. The number is far better than conventional SVR regressor (46.47%) and appraiser-based estimation (68.61%).

RMSE and MAE. Fig. 8 shows the RMSE and MAE in representative baseline methods, i.e., SVR, H-GCN and HG-LSTM against LUCE on the TorC-H dataset. As shown in Fig. 8, LUCE has much lower RMSE and MAE for house prediction in every month, with the lowest error fluctuation compared with other methods. Specifically, compared with H-GCN and HG-LSTM, the average RMSE in LUCE has decreased by up to 84.84% and 38.11%, respectively.
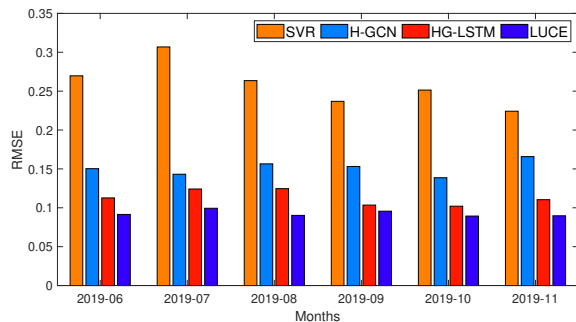
(a) The RMSE of the prediction results on TorC-H.
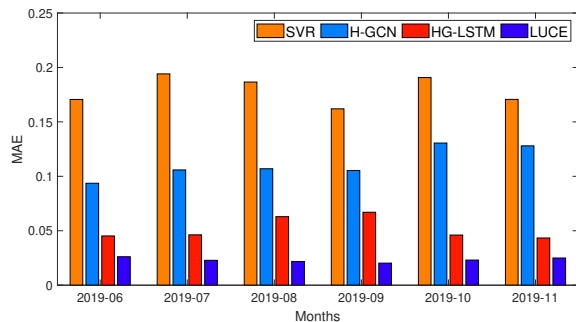
(b) The MAE of the prediction results on TorC-H.

Figure 8: The RMSE and MAE of the prediction results on TorC-H in the last 6 continuous months (2019-06 to 2019-11).

Table 6: RMSE and MAE comparison (2019-11).

	Methods	RMSE	MAE
TorC-H	SVR	0.2852	0.1881
	DT	0.2794	0.1694
	ENS	0.2805	0.1814
	Reg-PSO	0.2744	0.1738
	LSTM-D	0.3051	0.2031
	Appraiser	0.1158	0.0409
	H-GCN	0.1580	0.0958
	HT-GCN	0.2018	0.1349
	HG-LSTM	0.1333	0.0504
	LUCE	**0.0950**	**0.0297**
TorC-A	SVR	0.3012	0.1964
	DT	0.3604	0.2230
	ENS	0.2979	0.1905
	Reg-PSO	0.2866	0.1901
	LSTM-D	0.3223	0.2255
	Appraiser	0.1459	0.0941
	H-GCN	0.1984	0.1456
	HT-GCN	0.2575	0.1799
	HG-LSTM	0.1627	0.1068
	LUCE	**0.1336**	**0.0742**
Tor-H	SVR	0.3185	0.2185
	DT	0.3001	0.1967
	ENS	0.3045	0.1942
	Reg-PSO	0.2946	0.1841
	LSTM-D	0.2893	0.1847
	Appraiser	0.1677	0.1014
	H-GCN	0.2105	0.1322
	HT-GCN	0.1735	0.1081
	HG-LSTM	0.1812	0.1104
	LUCE	**0.1581**	**0.0982**

Similarly, the MAE of monthly house prices predicted by LUCE has decreased to only 0.029. Table 6 demonstrates the detailed comparison if we only extract the data of 2019-11, also revealing the fact that LUCE significantly outperforms other baseline methods.

This is because simple machine learning methods such as SVR, DT, ENS and Reg-PSO, only utilize the original features of houses, without employing heterogeneous information modeling to capture the intrinsic relationship and connections among houses. Due to the non-time-series techniques used in transactions modeling, their prediction turns out to be the worst in most cases. Regarding static graph neural network approaches such as H-GCN that are based on heterogeneous modeling, graph neural network can obtain a fusion representation based on the relationships between houses and thus facilitate to overcome the freshness and sparsity problem to some degree. Nevertheless, H-GCN model neglects the temporal dependencies in the transaction data, resulting in a performance discrepancy compared with other improvements via heterogeneous graph modeling, such as HT-GCN and HG-LSTM.

Due to the ignorance of heterogeneous characteristics, LSTM-D delivers inferior results compared against other time-series methods that can capture and model heterogeneous data. The results are even worse than SVR, DT, ENS and Reg-PSO on both TorC-H and TorC-A datasets. This is primarily because LSTM usually requires a large amount of high-quality data to underpin the feature learning. In the case of sparse transactions, the prediction of LSTM-D barely outperforms the general regression models; when the number of reference houses is insufficient, LSTM-D is even inferior to general regression models in some scenarios.

By contrast, such approaches as HT-GCN, HG-LSTM and LUCE elaborately consider the data sparsity by adopting heterogeneous graph embedding that can fully leverage house similarity and temporal dependency in the feature learning. In fact, HT-GCN and HG-LSTM are able to deliver competitive results on both TorC-H and TorC-A datasets – they can output equivalent or even better estimation compared to appraisers. Nevertheless, when dealing with long-term prediction problem, the lack of up-to-date house prices and valuation has much more negative impact on the prediction results. Furthermore, we observe that noise data stemming from the distant past house transactions has non-negligible impact on model performance and cause catastrophic forgetting problem. In comparision, LUCE employs the evolving multitask embedding to achieve constant parameter updates, resulting in a better effectiveness than HT-GCN and HG-LSTM. Our solution is even superior to the estimation by appraisers in most cases; the improvement can reach up to 20.21% on prediction error rate.

6.3 Lifelong Prediction Effectiveness

To further analyze the performance of handling large-scale and continuous prediction, we simulate LUCE's parameter update and prediction process in multi-month, on the occasion of new transaction data arrival from Tor-H. In this context, new prices will be predicted by the model; once a house is traded, the transaction price will be added to the train set so that model parameter will be updated. We record the loss of LUCE on the train set and test set during the simulation, where the train set loss is calculated by the optimized training loss in Eq. 8, and the test set loss is calculated by RMSE. As shown in Fig. 9, the training procedure in LUCE's is very stable across all stages – the performance on the train set and test set is generally consistent, indicating that continuous house price prediction can be effectively conducted in LUCE on large-scale datasets.
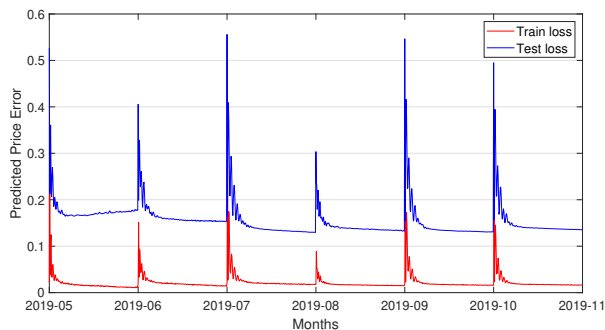
Figure 9: The train and test loss of LUCE in simulating lifelong continuous prediction.
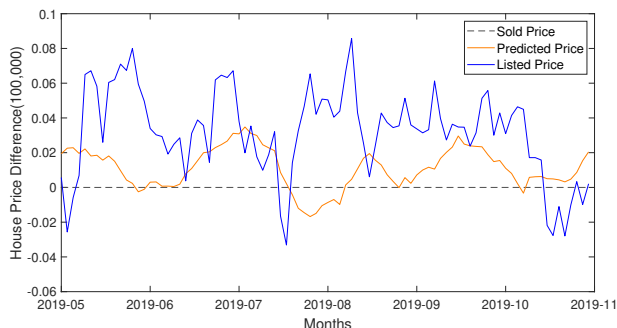

Figure 10: Differences between predicted price and appraiser's price compared to actual sold price in Tor-H.

To explicitly validate the effectiveness of lifelong prediction for the traded houses in the last six months of the dataset Tor-H, we plot in Fig. 10 the numerical discrepancies between the predicted price and the appraiser estimated price, together with the actual trading price as a baseline (in gray dotted line). Herein, we aggregate the average price of traded houses within an area. Although the house prices vary over time in dataset Tor-H without any strong regularity, LUCE is still able to deliver prices in closer proximity to the actual trading price compared with the appraiser's price.

To summarize, all the aforementioned experiments demonstrate that LUCE has qualified learning capability of spatio-temporal features, and thus overcomes the data freshness and sparsity manifesting in house transaction records, against other baselines including simple regression methods and conventional spatio-temporal mining methods.

6.4 Micro-benchmarking

In this section, we aim to demonstrate the individual contribution of different learning components to the holistic prediction effectiveness and performance.

Ablation study: LUCE-G, LUCE-L vs. LUCE. In light of the methodology of variable controlling, the main steps in this evaluation is to remain only one single component – whilst removing others – and examine how it affects the effectiveness. As depicted in Fig. 5 in §5, LUCE manages to continuously evolve the embedding by adaptively updating the parameters. The updates mainly depend on GCN layers and LSTM layers while lifelong learning relies upon the combinations of such GCN-LSTM units and the limited-depth recursive parameter updating strategy.

Hence, we identify two comparable tailored subsystems – LUCE-G (LUCE without LSTM layers) and LUCE-L (LUCE without GCN layers) – and compare them with the complete LUCE. We leverage the RMSE of the house prices prediction on the aforementioned 6-months transactions on TorC-H (6,000 houses) and TorC-A (3,000 houses) dataset, respectively. As shown in Fig. 11a and Fig. 11b, there is an RMSE increase in the LUCE-G and LUCE-L compared against LUCE, and the removal of LSTM layers has a greater impact on the performance than removal of GCN layers. This indicates the proposed lifelong learning framework can effectively and constantly tolerate the deficiency in up-to-date transaction data and alleviate the issue of time discontinuity.

Impact of parameter in lifelong learning. We further investigate how the lifelong learning parameter impact the overall performance of LUCE. Specifically, the key variable is the number of spatio-temporal feature learning units, i.e, the maximum updated length n. We retain the same adoption of RL based training optimization and examine the RMSE of predicting house prices in 2019-11 based on the dataset TorC-H. As shown in Fig. 11c, the optional configuration of the maximum update length is 6. With the increment of length, the back-propagation tends to experience vanished gradient increasingly and more out-of-date transaction data will be involved in the learning. Overall, the model's prediction error could be acceptable when the maximum update length is between 5-12 months. A smaller parameter will give rise to the surging RMSE because our model cannot completely explore the data of adjacent areas and months – the model fails to learn the spatio-temporal features sufficiently.

This study implies it is extremely imperative to carry out the lifelong learning framework within LUCE– in each of the prediction tasks, we desire to train LUCE with a moderate recursive length thereby effectively evolving the house embedding and minimizing the prediction error.

6.5 Effectiveness of Training Optimization

In §4.2, we regularize the house embedding in overlapping areas to optimize the training process of LUCE. In this section, we present some optimization details during the training process and test the effectiveness of these optimizations during training process.

Regularization of training loss. The optimization of training loss encompasses several portions including Eq. 5, Eq. 6 and Eq. 8. By contrast, the unoptimized training loss will be conducted without distance regulation, i.e., $\epsilon(P_t) = 0$[8].

We therefore evaluate two cases where LUCE is attached with and without such optimization based on 6-months TorC-H dataset, whilst using RMSE as the main indicator. As depicted in Fig. 12a, LUCE with regularization can significantly lower the prediction error against LUCE without regularization; the RMSE value can be reduced by 11.15% at most. This phenomenon is because distance regularization can integrate the features learned by the same house in different graph neural networks, thereby better coordinating and calibrating the feature embedding. In comparison,

---

8. For this reason, we also use a separate multi-layer perception for each graph neural network to perform price prediction.
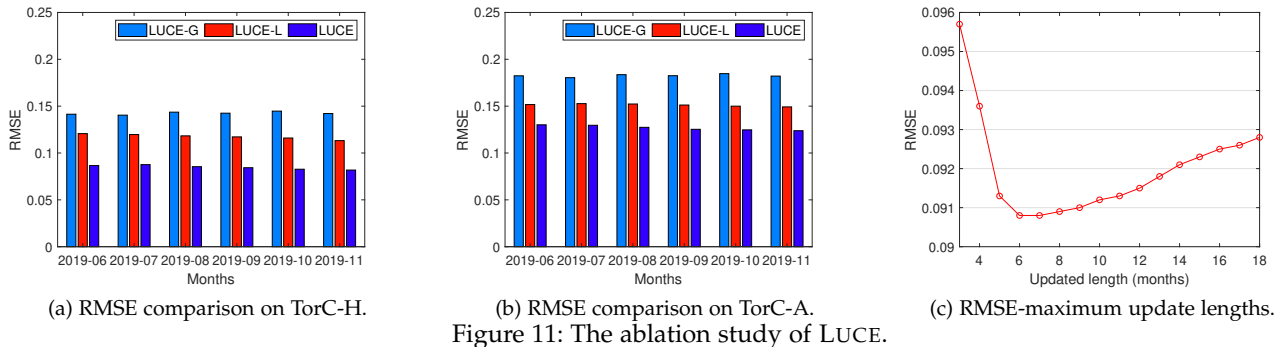
(a) RMSE comparison on TorC-H.

(b) RMSE comparison on TorC-A.

(c) RMSE-maximum update lengths.

Figure 11: The ablation study of LUCE.

(a) Comparison of optimized loss and unoptimized loss on TorC-H.

(b) Impact of parameter inheritance on TorC-H.
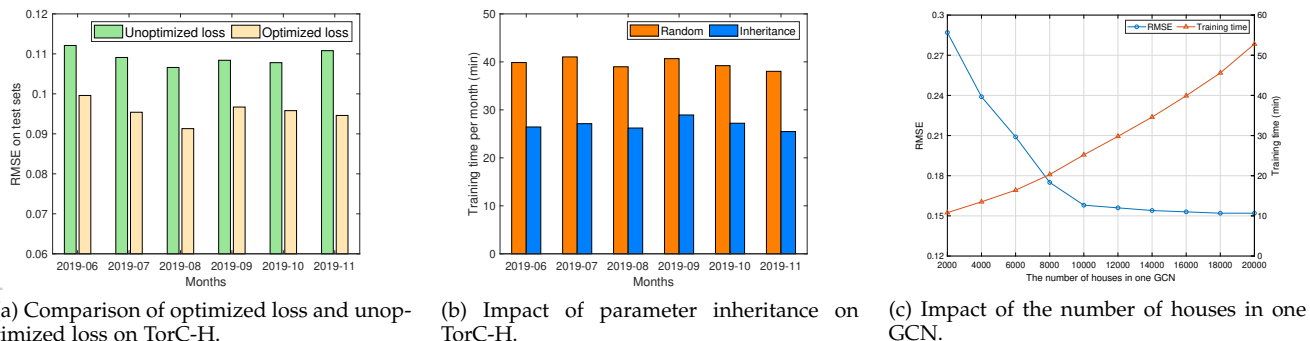
(c) Impact of the number of houses in one GCN.

Figure 12: LUCE's performance about training optimization.

models without regularization have to learn the house's own features without strong connections and fusions from external embedding results that can be reused.

Parameters inheritance. In order to shorten the time required for convergence during LUCE training, we adopt the strategy of parameters inheritance and examine its efficiency. This inheritance signifies the initial parameters W_t and θ_t of a new time step t can be possessed directly from the parameter W_{t-1} and θ_{t-1} of its prior time step, without learning from the scratch. Intuitively, the inheritance takes advantage of similarities of evolving house embeddings in adjacent months, which is beneficial to the initialization of model parameters when new data arrives.

At the other extreme, parameters will be randomly initialized – when the data of new time step t arrives, the initial values of parameter W_t and θ_t are given arbitrarily. To evaluate the convergence time, we run LUCE models by using parameter inheritance (Inheritance) and random parameter initialization (Random), separately. we test their training time per month to achieve convergence on the dataset TorC-H in the last 6 month. Fig. 12b indicates that parameter inheritance can facilitate to reduce the training time; the training time required to achieve convergence can be reduced by up to 33.90%.

Scalability: impact of house number. We conduct experiments to examine the impact of varying the number of houses within a GCN on RMSE and training time required to reach convergence, by ranging the number from 2,000 to 20,000. As depicted in Fig. 12c, when only a few houses available for learning in a GCN, it is inadequate for the GCN to effectively learn features of spatial information, due to the limited house overlap across different GCN units. Taking the dataset Tor-H as an example: the overlapping houses account for merely 4.73% of all houses on average in a single GCN. By contrast, the increment of the total num-

ber of houses results in a soaring number of overlapping, thereby improving the effectiveness of distance regulation. However, the training overhead will grow drastically when dealing with a vast number of house transactions – the training time to convergence increases significantly, susceptible to memory overflow in some worse-case scenarios. Hence, we leverage a proper number of houses (i.e. 10,000 in the experiments) in building the graph and GCN, to ultimately balance the training time and precision requirement under memory constraints.

6.6 Importance of Meta-paths and Meta-graphs

Fig. 13 shows the learnable weights $\vec{\omega}$ of different meta-paths and meta-graphs after training on the dataset TorC-H. We display the top 15 weights of the delivered meta-paths and meta-graphs and observably there are non-negligible differences in weights between different meta-paths and meta-graphs. We can find House-Spatial Information-House (H-SI-H) is the meta-path with the largest weight, indicating that the space information (SI) has the greatest impact on house prices among the various attributes in HIN, followed by building type (BT), layout structure (LS), garage type (GT), and so forth. This finding is coherent with our common understanding of property valuation. Meanwhile, the disparity among meta-paths and meta-graphs make it reasonable to calculate the inherent similarity whilst recognizing the most crucial factors that have heavy impact on the real estate market.

6.7 Time Consumption

We evaluate the time consumption to demonstrate the proposed LUCE is practical and scalable for the prediction tasks. Table 7 shows the results of the average time consumption and standard deviation under three datasets when
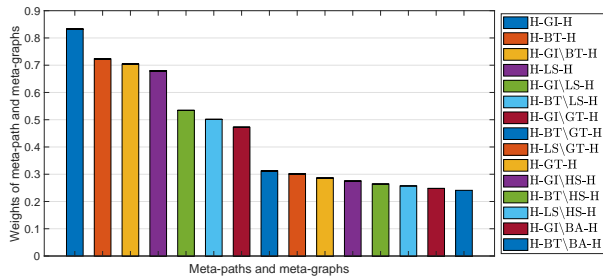
Figure 13: Top 15 weights of meta-paths and meta-graphs.

conducting monthly average price prediction. For example, LUCE can estimate the value of $618,339$ houses in the entire Toronto Region after a training period with 45.16 minutes. The observable minute-level solution stems from the adoption of a series of modules such as GCN, LSTM, distance regularization, and reinforcement learning that are usually time-consuming. The time complexity is acceptable, considering the achieved prediction accuracy which is competitive to the expertise of professional appraisers. In addition, our approach is very scalable when tackling a growing number of houses, since we split all houses into multiple pieces that can be processed by parallel GCN units.

Table 7: Statistics of the training time consumption in a monthly average setting (minute).

Model	TorC-H	TorC-A	Tor-H
LUCE	15.28±1.37	17.50±1.06	45.16±2.57

7 RELATED WORK

House price prediction. The prediction of house prices attracts researchers' attention because it can be regarded as a regression problem when there is sufficient transaction and characteristic information of houses. There have been a lot of studies to predict housing prices through simple machine learning techniques (such as decision tree and hedonic model) [3], [21], [22], [23], [1], [5] or deep learning neural networks with relatively simple structures [24], [25]. These techniques can generally take into account the spatial characteristics of the houses, encode the characteristics and send them to the model for training. In recent years, some researches [26], [27] have considered the impact of temporal features on house prices, while fully considering other houses characteristics, and using time series models to predict housing prices.

Heterogeneous graph learning. Here we mainly refer to the representation learning of heterogeneous graphs and dynamic graphs. For heterogeneous graph representation learning, it mainly includes two types of unsupervised heterogeneous information network embeddings [28], [29], [30], [31], [32], [33], [34] and semi-supervised heterogeneous graph neural networks [35], [36], [14], [37], [38], [39]. In terms of unsupervised heterogeneous information network embeddings, most of the approaches are based on meta-path [28], [29] or meta-graph [30] guided random walk on heterogeneous network to learn the embedding of nodes

with negative sampling technologies. In terms of semi-supervised heterogeneous graph neural networks, most existing researches are based on homogeneous graph neural networks, fusing different types node information [35], [36] or converting heterogeneous graphs into parameterized homogeneous graphs [14], and then learning node embedding through graph neural networks. Besides, some recent works [17], [18], [40] use RL to guide the neighborhood selection in graph neural networks through learning the optimized filtering thresholds. In this work, we also employ the Markov Decision Process to learn the penalty coefficients in our task.

For dynamic graph representation learning, there are a huge body of research on unsupervised, semi-supervised and supervised graph embedding models [41], [42], [43], [44], [45], [46]. Mainly through modeling dynamic processes, they have a focus on effective or efficient representation learning of dynamic structures. By contrast, in this work we assumed the information of geography, facilities and floorplan is static, while the financial information and transaction prices can change over time, i.e., the node attribute and node label are varying. Hence, it would be sufficient to use the LSTM units to learn the temporal house features. Overall, LUCE innovate an end-to-end learning framework by considering spatio-temporal characteristics and managing to tackle cope with the data freshness and sparsity.

Spatio-temporal data mining. Recent studies on spatio-temporal data prediction have combined models that extract spatial and temporal features. For example, ConvLSTM [47] is a combination of CNN and LSTM. In terms of spatial features, CNN is usually used for images, maps or data that can be modeled as grids [48], [49], [50], and graph neural network is usually used for data that can be modeled as graphs and networks [51], [52], [53], [54], [55]. In terms of temporal features, most researches utilize RNN to learn temporal features, including LSTM [19], GRU [20], Seq2Seq [56], Irregular Convolutional Residual LSTM [57], Transformer [58], etc. They can be used to replace the native LSTM modules and integrated with the current framework to further improve the prediction accuracy.

Lifelong learning. Lifelong learning is a relatively new research domain proposed in recent years, aiming to propose a method that can accumulate past knowledge and apply it to future learning [59]. In recent years, [60] retained the useful parameters for new tasks by changing the gradient update strategy, while ignoring those useless parameters [61] expanded the models and combines the trained models with the new model to train new tasks. [62] designed the gate to determine which past task the new task is more like to initialize the model of the new task. There are still a lot of works to study in the field of lifelong learning.

8 CONCLUSION

We have presented LUCE, a novel learning framework for automated property valuation. LUCE is designed to address the spatial and temporal sparsity of house transaction data. To extract useful information, LUCE organizes the house-related data in a heterogeneous information network (HIN). It then employs the GCN and LSTM to extract the spatial and temporal information from the HIN. LUCE uses GCN and LSTM to develop a lifelong learning framework for

house valuation for the first time. LUCE makes use of the limited recent house transactions data to update the valuation for all house entities in the HIN to provide a complete and update-to-date dataset to improve the accuracy of the downstream price prediction task. We evaluate LUCE by applying it to large-scale, real-world house transaction data of Toronto between 2000 and 2019. Experimental results show that LUCE consistently outperforms prior automated house valuation methods. It reaches and often exceeds the accuracy of valuation given by independent experts when using the actual sold price as the ground truth.

ACKNOWLEDGMENTS

REFERENCES

[1] S. C. Bourassa, E. Cantoni, and M. Hoesli, "Spatial dependence, housing submarkets, and house price prediction," *The Journal of Real Estate Finance and Economics*, vol. 35, no. 2, pp. 143–160, 2007.

[2] S. Basu and T. G. Thibodeau, "Analysis of spatial autocorrelation in house prices," *The Journal of Real Estate Finance and Economics*, vol. 17, no. 1, pp. 61–85, 1998.

[3] B. Park and J. K. Bae, "Using machine learning algorithms for housing price prediction: The case of fairfax county, virginia housing data," *Expert Systems with Applications*, no. 6, pp. 2928–2934, 2015.

[4] S. Stevenson, "New empirical evidence on heteroscedasticity in hedonic housing models," *Journal of Housing Economics*, vol. 13, no. 2, pp. 136–153, 2004.

[5] A. N. Alfiyatin, R. E. Febrita, H. Taufiq, and W. F. Mahmudy, "Modeling house price prediction using regression analysis and particle swarm optimization," *International Journal of Advanced Computer Science and Applications*, vol. 8, 2017.

[6] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[7] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, pp. 109–165.

[8] Y. Sun, J. Han, X. Yan, and P. S. Yu, "Mining knowledge from interconnected data: a heterogeneous information network analysis approach," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2022–2023, 2012.

[9] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[10] Q. Sun, H. Peng, J. Li, S. Wang, X. Dong, L. Zhao, S. Y. Philip, and L. He, "Pairwise learning for name disambiguation in large-scale heterogeneous academic networks," in *Proceedings of the IEEE ICDM*, 2020, pp. 511–520.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the ICLR*, 2018.

[12] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the NIPS*. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[14] H. Peng, J. Li, Q. Gong, Y. Song, Y. Ning, K. Lai, and P. S. Yu, "Fine-grained event categorization with heterogeneous graph convolutional networks," in *Proceedings of the IJCAI*. AAAI Press, 2019, pp. 3238–3245.

[15] H. Peng, J. Li, Y. Song, R. Yang, R. Ranjan, P. S. Yu, and L. He, "Streaming social event detection and evolution discovery in heterogeneous information networks," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 5, May 2021.

[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the ICLR*, 2017.

[17] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the ACM CIKM*, 2020, pp. 315–324.

[18] H. Peng, R. Zhang, Y. Dou, R. Yang, J. Zhang, and P. S. Yu, "Reinforced neighborhood selection guided multi-relational graph neural networks," *arXiv preprint arXiv:2104.07886*, 2021.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.

[21] O. Bin, "A prediction comparison of housing sales prices by parametric versus semi-parametric regressions," *Journal of Housing Economics*, vol. 13, no. 1, pp. 68–84, 2004.

[22] J. Chica-Olmo, "Prediction of housing location price by a multivariate spatial method: Cokriging," *Journal of Real Estate Research*, vol. 29, no. 1, pp. 91–114, 2007.

[23] L. Osland, "An application of spatial econometrics in relation to hedonic house price modeling," *Journal of Real Estate Research*, vol. 32, no. 3, pp. 289–320, 2010.

[24] V. Limsombunchai, "House price prediction: hedonic price model vs. artificial neural network," in *Proceedings of the New Zealand agricultural and resource economics society conference*, 2004, pp. 25–26.

[25] H. Selim, "Determinants of house prices in turkey: Hedonic regression versus artificial neural network," *Expert systems with Applications*, vol. 36, no. 2, pp. 2843–2852, 2009.

[26] T. Bollerslev, A. J. Patton, and W. Wang, "Daily house price indices: Construction, modeling, and longer-run predictions," *Journal of Applied Econometrics*, vol. 31, no. 6, pp. 1005–1025, 2016.

[27] X. Liu, "Spatial and temporal dependence in house price prediction," *The Journal of Real Estate Finance and Economics*, vol. 47, no. 2, pp. 341–369, 2013.

[28] T. Y. Fu, W. C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proceedings of the ACM CIKM*, 2017, p. 1797–1806.

[29] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the ACM SIGKDD*, 2017, p. 135–144.

[30] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Metagraph2vec: Complex semantic path augmented heterogeneous network embedding," in *Proceedings of the PAKDD*. Springer, 2018, pp. 196–208.

[31] Y. He, Y. Song, J. Li, C. Ji, J. Peng, and H. Peng, "Hetespaceywalk: A heterogeneous spacey random walk for heterogeneous information network embedding," in *Proceedings of the ACM CIKM*, 2019, p. 639–648.

[32] H. Ji, C. Shi, and B. Wang, "Attention based meta path fusion for heterogeneous information network embedding," in *Proceedings of the PRICAI*. Springer, 2018, pp. 348–360.

[33] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *Proceedings of the ACM SIGKDD*, New York, NY, USA, 2019, p. 1358–1368.

[34] L. Dos Santos, B. Piwowarski, and P. Gallinari, "Multilabel classification on heterogeneous graphs with gaussian embeddings," in *Proceedings of the ECMLKDD*, 2016, pp. 606–622.

[35] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *Proceedings of the WWW*. ACM, 2019, p. 2022–2032.

[36] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the ACM SIGKDD*. ACM, 2019, p. 793–803.

[37] J. Li, H. Peng, Y. Cao, Y. Dou, H. Zhang, P. Yu, and L. He, "Higher-order attribute-enhancing heterogeneous graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[38] Y. Hei, R. Yang, H. Peng, L. Wang, X. Xu, J. Liu, H. Liu, J. Xu, and L. Sun, "Hawk: Rapid android malware detection through heterogeneous graph attention networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[39] J. Gong, S. Wang, J. Wang, W. Feng, H. Peng, J. Tang, and P. S. Yu, "Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view," in *Proceedings of the ACM SIGIR*, 2020, pp. 79–88.

[40] Z. Zhong, C.-T. Li, and J. Pang, "Reinforcement learning enhanced heterogeneous graph neural network," *arXiv preprint arXiv:2010.13735*, 2020.

[41] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI*, vol. 34, no. 04, 2020, pp. 5363–5370.

[42] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proceedings of the ICONIP*. Springer, 2018, pp. 362–373.

[43] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognition*, vol. 97, p. 107000, 2020.

[44] H. Peng, R. Yang, Z. Wang, J. Li, L. He, P. Yu, A. Zomaya, and R. Ranjan, "Lime: Low-cost incremental learning for dynamic heterogeneous information networks," *IEEE Transactions on Computers*, 2021.

[45] H. Peng, J. Li, H. Yan, Q. Gong, S. Wang, L. Liu, L. Wang, and X. Ren, "Dynamic network embedding via incremental skip-gram with negative sampling," *Science China Information Sciences*, vol. 63, no. 10, pp. 1–19, 2020.

[46] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey." *Journal of Machine Learning Research*, vol. 21, no. 70, pp. 1–73, 2020.

[47] X. SHI, Z. Chen, H. Wang, D. Y. Yeung, W.-k. Wong, and W. C. WOO, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Proceedings of the NIPS*, 2015, pp. 802–810.

[48] S. Dabiri and K. Heaslip, "Inferring transportation modes from gps trajectories using a convolutional neural network," *Transportation research part C: emerging technologies*, vol. 86, pp. 360–371, 2018.

[49] C. Chen, K. Li, S. G. Teo, G. Chen, X. Zou, X. Yang, R. C. Vijay, J. Feng, and Z. Zeng, "Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction," in *Proceedings of the IEEE ICDM*, 2018, pp. 893–898.

[50] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE ICCV*, 2015, pp. 4489–4497.

[51] D. Chai, L. Wang, and Q. Yang, "Bike flow prediction with multi-graph convolutional networks," in *Proceedings of the ACM SIGSPATIAL*, 2018, p. 397–400.

[52] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proceedings of the AAAI*, vol. 33, 2019, pp. 3656–3663.

[53] Y. Lin, N. Mago, Y. Gao, Y. Li, Y.-Y. Chiang, C. Shahabi, and J. L. Ambite, "Exploiting spatiotemporal patterns for accurate air quality forecasting using deep learning," in *Proceedings of the ACM SIGSPATIAL*, 2018, pp. 359–368.

[54] H. Peng, H. Wang, B. Du, M. Z. A. Bhuiyan, H. Ma, J. Liu, L. Wang, Z. Yang, L. Du, S. Wang *et al.*, "Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting," *Information Sciences*, vol. 521, pp. 277–290, 2020.

[55] H. Peng, B. Du, M. Liu, M. Liu, S. Ji, S. Wang, X. Zhang, and L. He, "Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning," *Information Sciences*, vol. 578, pp. 401–416, 2021.

[56] B. Liao, J. Zhang, M. Cai, S. Tang, Y. Gao, C. Wu, S. Yang, W. Zhu, Y. Guo, and F. Wu, "Dest-resnet: A deep spatiotemporal residual network for hotspot traffic speed prediction," in *Proceedings of the ACM MM*, 2018, pp. 1883–1891.

[57] B. Du, H. Peng, S. Wang, M. Z. A. Bhuiyan, L. Wang, Q. Gong, L. Liu, and J. Li, "Deep irregular convolutional residual lstm for urban traffic passenger flows prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 972–985, 2019.

[58] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *arXiv preprint arXiv:2106.13008*, 2021.

[59] Z. Chen, E. R. Hruschka Jr, and B. Liu, "Lifelong machine learning and computer reading the web," in *Proceedings of the ACM SIGKDD*, 2016, pp. 2117–2118.

[60] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov, "Rotate your networks: Better weight consolidation and less catastrophic forgetting," in *Proceedings of the IEEE ICPR*, 2018, pp. 2262–2268.

[61] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[62] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proceedings of the IEEE CVPR*, 2017, pp. 3366–3375.

Hao Peng is currently an Assistant Professor at the School of Cyber Science and Technology, and Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His research interests include representation learning, social network mining and reinforcement learning. To date, Dr Peng has published over 60 research papers in top-tier journals and conferences, including the IEEE TKDE, TC, TPDS, TITS, ACM TKDD, TIST, TSAS and Web Conference.

Jianxin Li is currently a Professor with the State Key Laboratory of Software Development Environment, and Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His current research interests include social network, machine learning, big data and trustworthy computing.

Zheng Wang is an Associate Professor with the University of Leeds, UK. His research cuts across the boundaries of parallel program optimisation, systems security, and applied machine learning.

Renyu Yang is an EPSRC-funded Research Fellow with the University of Leeds, UK. He has industrial experience in building large-scale distributed systems with ML and co-authored/co-led many research grants including UK EPSRC, Innovate UK, EU Horizon 2020, China 973 and 863, etc. His research interests include reliable distributed systems, resource management and applied machine learning.

Mingsheng Liu is a Professor in Shijiazhuang Institute of Railway Technology, Shijiazhuang, China. His research interests include urban computing, big data and deep learning.

Lifang He is currently an Assistant Professor in the Department of Computer Science and Engineering at Lehigh University. Before her current position, Dr. He worked as a postdoctoral researcher in the Department of Biostatistics and Epidemiology at the University of Pennsylvania. Her current research interests include machine learning, data mining, tensor analysis, with major applications in biomedical data, neuroscience, or multimodal data.
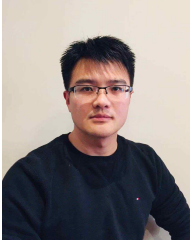
Mingming Zhang is a senior engineer in Ur-Brain Technology, Toronto, Canada. His research interests include urban computing and big data mining.

Philip S. Yu is a Distinguished Professor and the Wexler Chair in Information Technology at the Department of Computer Science, University of Illinois at Chicago. Before joining UIC, he was at the IBM Watson Research Center, where he built a world-renowned data mining and database department. He is a Fellow of the ACM and IEEE. Dr. Yu was the Editor-in-Chiefs of ACM Transactions on Knowledge Discovery from Data (2011-2017) and IEEE Transactions on Knowledge and Data Engineering (2001-2004).