# Heterogeneous Temporal Graph Neural Network

Yujie Fan* Mingxuan Ju† Chuxu Zhang‡ Yanfang Ye*†§

**Abstract**

Graph neural networks (GNNs) have been broadly studied on dynamic graphs for their representation learning, majority of which focus on graphs with homogeneous structures in the spatial domain. However, many real-world graphs - i.e., *heterogeneous temporal graphs* (HTGs) - evolve dynamically in the context of heterogeneous graph structures. The dynamics associated with heterogeneity have posed new challenges for HTG representation learning. To solve this problem, in this paper, we propose *heterogeneous temporal graph neural network* (HTGNN) to integrate both spatial and temporal dependencies while preserving the heterogeneity to learn node representations over HTGs. Specifically, in each layer of HTGNN, we propose a hierarchical aggregation mechanism, including intra-relation, inter-relation, and across-time aggregations, to jointly model heterogeneous spatial dependencies and temporal dimensions. To retain the heterogeneity, intra-relation aggregation is first performed over each slice of HTG to attentively aggregate information of neighbors with the same type of relation, and then intra-relation aggregation is exploited to gather information over different types of relations; to handle temporal dependencies, across-time aggregation is conducted to exchange information across different graph slices over the HTG. The proposed HTGNN is a holistic framework tailored heterogeneity with evolution in time and space for HTG representation learning. Extensive experiments are conducted on the HTGs built from different real-world datasets and promising results demonstrate the outstanding performance of HTGNN by comparison with state-of-the-art baselines. Our built HTGs and code have been made publicly accessible at: https://github.com/YesLab-Code/HTGNN.

## 1 Introduction

Many real-world data come in the form of graphs, such as academic networks [1, 2], social networks [3, 4], and epidemiological networks [5, 6]. The graph structure consists of a set of nodes interconnected by a set of edges. Learning node representations on graphs is essential for various downstream tasks, such as node classification, node clustering, link prediction, and personalized recommendation.

Recently, graph neural networks (GNNs) have been broadly studied and achieved state-of-the-art performance by taking both node features as well as graph structures into consideration. Despite their superior performance, most of the current research efforts concentrate on static graphs [7, 8, 9, 10] or dynamic/spatial-temporal graphs with homogeneous structures [11, 12, 5, 13]. However, many real-world graphs evolve dynamically in the context of heterogeneous graph structures. From the perspective of spatial domain, the graph is heterogeneous with multi-typed nodes connected by multi-typed relations; from the perspective of temporal domain, either the node features or graph structures evolve over time. We call this type of graph the *heterogeneous temporal graph* (HTG). A HTG could be described as an ordered list of heterogeneous graph slices with a set of temporal relations connecting them. It is a general concept for modeling heterogeneous and dynamically changing graph data. Typical examples include dynamic academic networks and epidemiological networks. For dynamic academic networks, the heterogeneous structures would evolve along with the authors' research directions and co-authorship. In contrast, the graph structures remains unchanged for epidemiological networks, but the node features inevitably change with increased/decreased patient numbers. It is worth noting that dynamic heterogeneous graphs [14, 15, 16, 17] can be treated as an instance of HTGs, where the dynamic nature comes from the evolving graph structures.

The dynamics associated with heterogeneity have posed new challenges for representation learning on HTGs. There exist some preliminary works. They could be roughly summarized into two categories: one first explores neural sequence models to process time-series features attached on each node, and then performs graph representation learning with the processed node features on the spatial domain [18, 5, 6, 19]; the other first applies GNNs on each graph slice of a HTG, and then employs sequence models on the outputs of each slice to obtain the final representations [20, 16, 15]. Although these works could achieve satisfactory results, they are still faced with the following limitations: (1) The existing models are graph-dependent. That is, the perfor-

*Case Western Reserve University, Cleveland, USA
†University of Notre Dame, Notre Dame, USA
‡Brandeis University, Waltham, USA
§Corresponding author: yye7@nd.edu

mance depends heavily on the characteristics of a graph. Specifically, for a HTG with dynamically evolving heterogeneous graph structures (e.g., academic networks), the second category approaches that emphasize more on spatial dependencies usually obtain better results. On the contrary, for a HTG with constantly changing node features (e.g., epidemiological networks), the first category methods that focus more on temporal dependencies would achieve superior performance. Apparently, selecting a model that best fits with a given HTG requires empirical knowledge. (2) The spatial and temporal dependencies are processed in a serialized way. Most existing models either analyze the temporal domain first and the spatial domain later or in reverse order, which weakens the spatial-temporal interactions as the information in these two domains is treated separately.

Currently, it is not yet well understood how to jointly integrate both spatial and temporal dependencies while preserving the heterogeneity for node representation learning over HTGs. To fill this gap, in this paper, we propose *heterogeneous temporal graph neural network* (HTGNN), a holistic framework tailored heterogeneity with evolution in time and space to learn node representations on HTGs. More specifically, to retain the *spatial heterogeneity*, we design intra-relation aggregation and inter-relation aggregation, which are performed purely on each graph slice of a HTG, to successively aggregate the information of a target node's neighbors within the same type of relation and over different types of relations. To handle the *temporal dependencies*, we introduce across-time aggregation, which is conducted across different graph slices, to gather the information of the target node's temporal neighbors. To capture the *spatial-temporal interactions*, we equip each layer of HTGNN with a hierarchical aggregation mechanism, including intra-relation, inter-relation, and across-time aggregation modules, to jointly, rather than serially, model heterogeneous spatial dependencies and temporal dimensions. With increased model depth, the information is iteratively propagated in these two domains, allowing HTGNN agnostic to graph characteristics. In sum, we make the following contributions:

- We study the representation learning problem on HTGs. HTG is a general concept to model graph data with heterogeneous spatial structures and temporal evolution patterns (i.e., dynamically evolving graph structures or constantly changing node features).
- We propose HTGNN to learn node representations on HTGs. HTGNN is a holistic framework, which is capable of jointly modeling heterogeneous spatial dependencies and temporal dimensions. This character differs from existing works that process these two types of dependencies in a serialized way.
- We establish two HTGs from different real-world datasets, one with dynamically evolving heterogeneous structures (i.e., OGBN-MAG) and another with constantly changing node features (i.e., COVID-19). Extensive experiments on these two HTGs demonstrate that HTGNN consistently achieves strong performance in comparison with state of the arts for different graph mining tasks.

## 2 Related Work

**Heterogeneous Graph Neural Networks.** Recently, various heterogeneous GNNs [21, 22, 10, 2, 9, 23] have been proposed for learning on heterogeneous graphs. RGCN [21] introduces relation-specific transformations for different relation types during the learning process. HGT [10] utilizes meta relations to model graph heterogeneity and further learns the mutual attention for each meta relation based on the Transformer architecture. By leveraging metapaths defined on the heterogeneous graphs, HAN [2] designs node-level attention and semantic-level attention to learn the importance of metapath-based neighbors and different metapaths, respectively. These models are built for static heterogeneous graphs and cannot deal with the dynamic properties of HTGs. It is worth noting that, HGT uses a relative temporal encoding to assign each node a timestamp to handle graph dynamics. However, this could partially address the problem as the node embedding in HGT is assumed to be time-invariant, which is not in line with many real-world scenarios.

**Dynamic Graph Learning.** Spatial-temporal graphs [11, 18, 6, 5] and dynamic graphs with homogeneous structures [12, 24, 13] have been widely studied in the literature. To further consider the graph heterogeneity, learning on dynamic heterogeneous graphs has drawn increasing attention, including dynamic heterogeneous graph embedding models [25, 26, 17, 14] that solely consider graph structures and dynamic heterogeneous GNNs [15, 20, 27] that take both graph structure and node features into consideration. DyHATR [15] first introduces node-level and edge-level attentions to learn heterogeneous information and then applies RNNs with temporal attention to capture temporal dependencies. HDGAN [28] combines heterogeneous attention and Hawkes process to model graph heterogeneity and dynamics. However, there are some limitations in current works. Firstly, these models are graph-dependent, which requires empirical knowledge; secondly, the spatial and temporal dependencies are processed serially, which leads to a weakened connection between these two domains. This paper addresses these issues by designing a holistic graph-agnostic framework to learn node representations on HTGs.

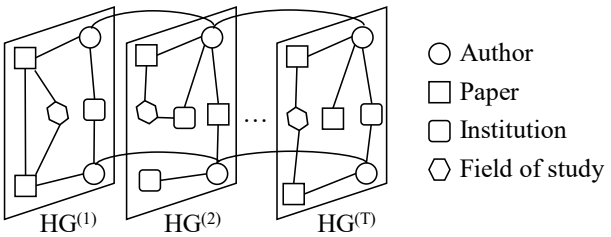

Figure 1: An graphical illustration of HTG.

DEFINITION 3.3. ***Relation-$r$-based Neighbors.*** *Given a relation type $r \in R$, a node $v$ at timestamp $t$, its relation-$r$-based neighbors at timestamp $t$ is defined as $\mathcal{N}_r^t(v) = \{u|(u,v) \in E^t, \psi(u,v) = r\}$.*

DEFINITION 3.4. ***Heterogeneous Temporal Graph Representation Learning.*** *Given a HTG $\mathcal{G} = \left(\{G^t\}_{t=1}^T, E'\right) = (\mathcal{V}, \mathcal{E})$ consisting of $T$ timestamps, the task of HTG representation learning is to learn a general $d$-dimensional node representation $\mathbf{h}_v \in \mathbb{R}^d$ for each node $v \in \mathcal{V}$ with $d \ll |\mathcal{V}|$. The node representations are able to capture both spatial heterogeneity and temporal dependencies of the HTG, and could be applied in various downstream tasks at timestamp $T+1$.*

## 4 Methodology

**4.1 Overview** The framework of HTGNN is shown in Figure 2. HTGNN takes a HTG as input and yields node representations as outputs for downstream tasks (see Figure 2 (a)). It is composed of multiple heterogeneous temporal aggregation layers. Each layer is equipped with a hierarchical aggregation mechanism (see Figure 2 (b)), including intra-relation, inter-relation, and across-time aggregation modules. Intra-relation and inter-relation aggregation modules are performed purely on each graph slice, aiming to depict the spatial heterogeneity, while the across-time aggregation module is conducted across different graph slices, aiming to capture the temporal dependencies. In one aggregation layer, each node successively receives messages from its spatial neighbors of the same relation type and different relation types; the nodes then start gathering messages from their temporal neighbors across graph slices. After this, another layer follows with node embeddings obtained from the previous layer. With increased model depth, the messages are iteratively propagated in spatial and temporal domains. Such design makes HTGNN be a holistic framework jointly modeling heterogeneous spatial dependencies and temporal dimensions.

**4.2 Intra-relation Aggregation** In a heterogeneous graph, each node type may have its own feature space. Take OGBN-MAG dataset as an example where only nodes with paper type are associated with input features. Metapath2vec [30] is a common approach to initiate features for those nodes without input features. Apparently, the feature spaces for paper type and other types are different as the former reflects the text content information while the latter represents the graph structural information. To handle this problem, before feeding node features into multiple heterogeneous temporal aggregation layers, we first adopt a type-specific projection on each node to map its distinct feature vector into a same feature space. Mathematically, given a node $v$ with node type $\phi(v)$ at timestamp $t$, we have:

$$\mathbf{h}_v^{t,0} = \mathbf{W}_{\phi(v)} \cdot \mathbf{x}_v^t, \tag{4.1}$$

where $\mathbf{x}_v^t \in \mathbb{R}^{d'}$ and $\mathbf{h}_v^{t,0} \in \mathbb{R}^d$ are $d'$-dimensional raw feature vector and $d$-dimensional projected embedding of node $v$, respectively; $\mathbf{W}_{\phi(v)} \in \mathbb{R}^{d \times d'}$ is the trainable type-specific transformation matrix.

The intra-relation aggregation module is performed separately on each relation type in each graph slice. It takes the node embeddings of last layer as inputs and outputs multiple relation embeddings for each node at each timestamp. Given a target node $v$ at timestamp $t$ and a relation type $r \in R$, the intra-relation aggregation can be described as:

$$\mathbf{h}_{v,r}^{t,l} = AGG_{intra}\Big(\big\{\mathbf{h}_u^{t,l-1} | u \in \mathcal{N}_r^t(v)\big\}; \mathbf{\Theta}_{intra}\Big), \tag{4.2}$$

where $\mathcal{N}_r^t(v)$ represents relation-$r$-based neighbors of node $v$ at timestamp $t$, $\mathbf{h}_u^{t,l-1}$ is node $u$'s embedding at timestamp $t$ in layer $l-1$, $\mathbf{h}_{v,r}^{t,l}$ denotes the relation $r$'s embedding with respect to node $v$ at timestamp $t$ in layer $l$, and $\mathbf{\Theta}_{intra}$ is the trainable parameters and is non-shareable for relation type, timestamp and aggregation layer. Noted that when $l = 1$, $AGG_{intra}$ takes the outputs of type-specific projection module introduced above as inputs.

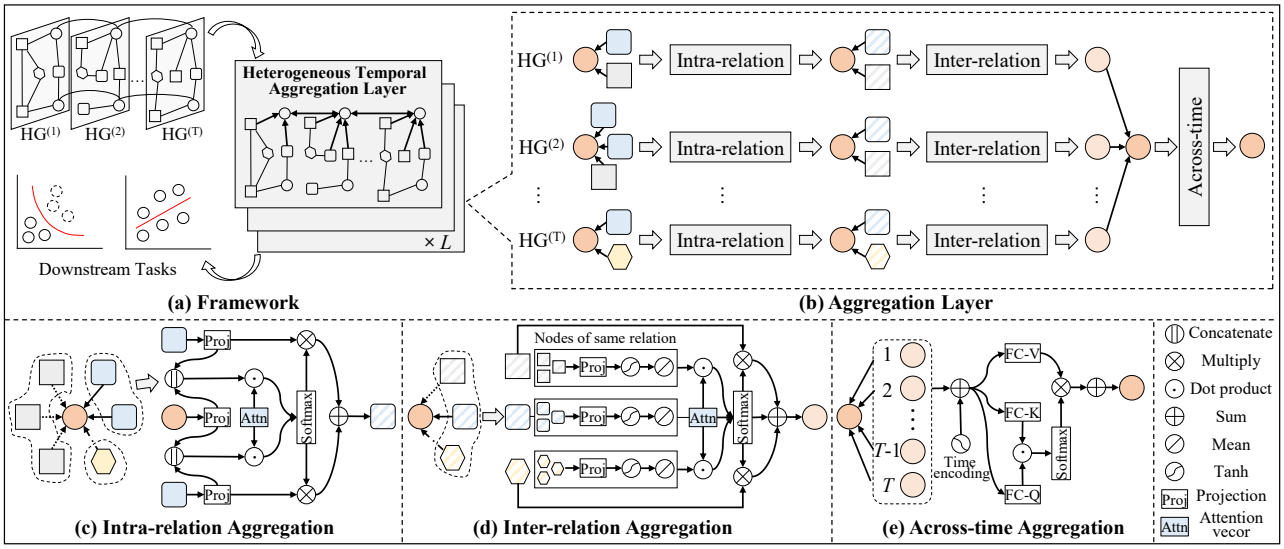


Figure 2: The overall framework of heterogeneous temporal graph neural network.

For a target node, its different neighbors, even within the same type of relation, would also contribute differently in the learning process, and thus we adopt the self-attention mechanism to assign each neighbor a weight reflecting its importance. Formally, given a target node $v$ and one of its relation-$r$-based neighbors at timestamp $t$, i.e., $u \in \mathcal{N}_r^t(v)$, their attention coefficient can be computed by:

$$e_{(u,v),r}^{t,l} = \sigma\Big(\big[\mathbf{a}_r^{t,l}\big]^\top \cdot \big[\mathbf{W}_r^{t,l} \cdot \mathbf{h}_u^{t,l-1} \| \mathbf{W}_r^{t,l} \cdot \mathbf{h}_v^{t,l-1}\big]\Big), \tag{4.3}$$

where $\sigma(\cdot)$ is LeakyReLU, $\mathbf{W}_r^{t,l} \in \mathbb{R}^{d \times d}$ and $\mathbf{a}_r^{t,l} \in \mathbb{R}^{2d}$ are trainable transformation matrix and attention vector, respectively, and $\|$ concatenates the vector. We then normalize the attention coefficient across all relation-$r$-based neighbors via softmax function:

$$\alpha_{(u,v),r}^{t,l} = \frac{\exp\left(e_{(u,v),r}^{t,l}\right)}{\sum_{u' \in \mathcal{N}_r^t(v)} \exp\left(e_{(u',v),r}^{t,l}\right)}. \tag{4.4}$$

After obtaining the normalized attention values of node $v$'s neighbors, we perform weighted combination to compute relation $r$'s embedding with respect to $v$:

$$\mathbf{h}_{v,r}^{t,l} = \sigma\Bigg(\sum_{u \in \mathcal{N}_r^t(v)} \big[\alpha_{(u,v),r}^{t,l}\big] \cdot \big[\mathbf{W}_r^{t,l} \cdot \mathbf{h}_u^{t,l-1}\big]\Bigg). \tag{4.5}$$

Figure 2 (c) illustrates the implementation of intra-relation aggregation module. Without loss of generality, multi-head attention mechanism can be employed. Specifically, $K$ independent attention heads are executed in a parallel fashion, and then their representations are concatenated, as following:

$$\mathbf{h}_{v,r}^{t,l} = \Big\|_{k=1}^{K} \sigma\Bigg(\sum_{u \in \mathcal{N}_r^t(v)} \big[\alpha_{(u,v),r}^{t,l}\big]^k \cdot \big[\mathbf{W}_r^{t,l} \cdot \mathbf{h}_u^{t,l-1}\big]^k\Bigg), \tag{4.6}$$

where $\big[\alpha_{(u,v),r}^{t,l}\big]^k$ denotes the attention value at $k$-th attention head.

**4.3 Inter-relation Aggregation** Through intra-relation aggregation, the target node would gather multiple relation embeddings. Based on this, the inter-relation aggregation module aims to learn a spatial embedding for the target node summarizing the information of its spatial neighbors over all relation types. Formally, this process is denoted as:

$$\mathbf{h}_{v,R}^{t,l} = AGG_{inter}\Big(\big\{\mathbf{h}_{v,r}^{t,l} | r \in R(v)\big\}; \mathbf{\Theta}_{inter}\Big), \tag{4.7}$$

where $R(v)$ denotes the set of relations connected to node $v$, $\mathbf{h}_{v,r}^{t,l}$ is relation $r$'s embedding with respect to node $v$ from previous module, $\mathbf{h}_{v,R}^{t,l}$ denotes the spatial embedding of node $v$ that will be learned in this module, $\mathbf{\Theta}_{inter}$ is the trainable parameters and is non-shareable for timestamp and aggregation layer.

A straightforward way to implement $AGG_{inter}(\cdot)$ is treating each relation embedding equally by conducting element-wise sum/mean operation or concatenating them followed by a linear transformation. However, each relation type preserves a unique semantic meaning and thus should not be treated identically. Therefore, we manage to learn a importance weight for each relation type and explore the attention mechanism for implementation. Specifically, for relation type $r$, we use a three-step process to learn its importance: (1) we first

retrieve its embeddings of all related nodes and feed them into a non-linear transformation; (2) we generate its summarized embedding by averaging the transformed relation embeddings; (3) finally, we calculate its attention coefficient by measuring the similarity between its summarized embedding with a relation attention vector. This learning process is formalized as:

$$e_r^{t,l} = \left[\mathbf{c}_R^{t,l}\right]^\top \cdot \frac{1}{|V_r^t|} \sum_{v' \in V_r^t} \left[\tanh\left(\mathbf{W}_R^{t,l} \cdot \mathbf{h}_{v',r}^{t,l} + \mathbf{b}\right)\right], \tag{4.8}$$

where $V_r^t$ denotes the set of nodes connected by relation $r$ at timestamp $t$, $\mathbf{b} \in \mathbb{R}^d$ is the bias vector, $\mathbf{W}_R^{t,l} \in \mathbb{R}^{d \times d}$ and $\mathbf{c}_R^{t,l} \in \mathbb{R}^d$ are trainable transformation matrix and attention vector, respectively. The normalized importance of $r$ with regard to $v$ is computed as:

$$\beta_{v,r}^{t,l} = \frac{\exp\left(e_r^{t,l}\right)}{\sum_{r' \in R(v)} \exp\left(e_{r'}^{t,l}\right)}. \tag{4.9}$$

With the importance of different relations, we generate the spatial embedding for $v$ via linear combination:

$$\mathbf{h}_{v,R}^{t,l} = \sum_{r \in R(v)} \left[\beta_{v,r}^{t,l}\right] \cdot \left[\mathbf{h}_{v,r}^{t,l}\right]. \tag{4.10}$$

An intuitive explanation is shown in Figure 2 (d). We could also extend it to multi-head mechanism.

**4.4 Across-time Aggregation** Intra-relation and inter-relation aggregation modules that aggregate information from the target node's spatial neighbors are performed purely on each graph slice. Across-time aggregation aims to capture the interactions among the target node's temporal neighbors. We define the temporal neighbors as the same nodes in different graph slices (including itself). This module takes in the spatial embeddings of the target node's temporal neighbors and outputs a spatial-temporal embedding for this target node. This process is formalized as follow:

$$\mathbf{h}_{v,ST}^{t,l} = AGG_{across}\Big(\big\{\mathbf{h}_{v,R}^{t',l} | 1 \le t' \le T\big\}; \mathbf{\Theta}_{across}\Big), \tag{4.11}$$

where $\mathbf{h}_{v,R}^{t',l}$ is the spatial embedding of node $v$'s temporal neighbor at timestamp $t'$ in layer $l$, $\mathbf{h}_{v,ST}^{t,l}$ is node $v$'s spatial-temporal embedding at timestamp $t$ in layer $l$, $\mathbf{\Theta}_{across}$ is the trainable parameters and is non-shareable for node type and aggregation layer.

As the transformer [29] has shown great performance in natural language processing domain, we explore its attention mechanism to model our across-time aggregation process. Before calculating attentions for different timestamps, we define a time encoding function $PE(\cdot)$ for $\mathbf{h}_{v,R}^{t,l}$ to incorporate time-related factors:

$$PE\Big(\mathbf{h}_{v,R}^{t,l}\Big) = \|_{i=1}^{d}\Big(\mathbf{h}_{v,R}^{t,l}(i) + p(t,i)\Big), \tag{4.12}$$

where $i$ is the index of each element and $p(\cdot)$ is a frequency encoding function that characterizes a time-dependent sinusoid, where $p(t,i) = \sin(t/10000^{2i/d})$ if $i$ is even, or $\cos(t/10000^{2i/d})$ if odd. By feeding the embeddings at different timestamps into this function, they become discriminative with regard to time. We then transform the target node's spatial embedding into Query vector, its temporal neighbor's spatial embedding into Key vector, and calculate their dot product as the attention coefficient to measure the importance of this temporal neighbor. Accordingly, we have:

$$\begin{aligned} \mathbf{q}_v^{t,l} &= \mathbf{W}_{\phi(v),q}^{l} \cdot PE\Big(\mathbf{h}_{v,R}^{t,l}\Big), \\ \mathbf{k}_v^{t',l} &= \mathbf{W}_{\phi(v),k}^{l} \cdot PE\Big(\mathbf{h}_{v,R}^{t',l}\Big), \end{aligned} \tag{4.13}$$

where $\mathbf{W}_{\phi(v),q}^{l}, \mathbf{W}_{\phi(v),k}^{l} \in \mathbb{R}^{d \times d}$ denote the trainable transformation matrices for query, and key, respectively. The normalized attention value is then calculated by:

$$\gamma_v^{t',l} = \frac{\exp\Big(\left[\mathbf{q}_v^{t,l}\right]^\top \cdot \left[\mathbf{k}_v^{t',l}\right]\Big)}{\sum_{t''=1}^{T} \exp\Big(\left[\mathbf{q}_v^{t,l}\right]^\top \cdot \left[\mathbf{k}_v^{t'',l}\right]\Big)}. \tag{4.14}$$

Finally, node $v$'s spatial-temporal embedding is computed via a linear combination of its temporal neighbors' transformed embeddings and the calculated attention values, formulated as:

$$\begin{aligned} \mathbf{v}_v^{t',l} &= \mathbf{W}_{\phi(v),v}^{l} \cdot PE\Big(\mathbf{h}_{v,R}^{t',l}\Big), \\ \mathbf{h}_{v,ST}^{t,l} &= \sigma\Big(\sum_{t'=1}^{T} \left[\gamma_v^{t',l}\right] \cdot \left[\mathbf{v}_v^{t',l}\right]\Big). \end{aligned} \tag{4.15}$$

A graphical explanation of across-time aggregation is shown in Figure 2 (e). Similarly, multi-head attention mechanism could also be applied in this module.

Using the hierarchical aggregation mechanism above, we can obtain the spatial-temporal embedding for each node. Despite this, the feature vector of node itself also plays an essential role in learning its representation. Instead of directly summing them, we design a gate mechanism to control how much information the node and its neighbors should contribute. Given the node feature vector at timestamp $t$ in layer $l-1$ and its spatial-temporal embedding at the same timestamp in layer $l$, their combination is formulated as:

$$\mathbf{h}_v^{t,l} = \delta_{\phi(v)} \cdot \left[\mathbf{h}_{v,ST}^{t,l}\right] + (1 - \delta_{\phi(v)}) \cdot \left[\mathbf{W}_{\phi(v)} \cdot \mathbf{h}_v^{t,l-1}\right], \tag{4.16}$$

where $\delta_{\phi(v)} \in \mathbb{R}^1$ and $\mathbf{W}_{\phi(v)} \in \mathbb{R}^{d \times d}$ are the trainable weight and transformation matrix, respectively.

Table 1: Statistics of datasets.

| Dataset | Graph | Time Span | Node | Relation | Feature | Data Split |
|---|---|---|---|---|---|---|
| OGBN-MAG | # Graph: 10<br>Granularity: year | 2010-2019 | # Author: 17,764<br># Paper: 282,039<br># Field: 34,601<br># Institution: 2,276 | # Author-Paper: 2,061,677<br># Paper-Paper:2,377,564<br># Paper-Field: 289,376<br># Author-Institution: 40,307 | Paper: 128<br>Author: 128<br>Institution: 128<br>Field: 128 | Training: 8<br>Validation: 1<br>Testing: 1 |
| COVID-19 | # Graph: 304<br>Granularity: day | 05/01/2020-<br>02/28/2021 | # State: 54<br># County: 3223 | # State-State: 269<br># State-County: 3,141<br># County-County: 22,176 | State: 1<br>County: 1 | Training: 244<br>Validation: 30<br>Testing: 30 |

**4.5 Learning Algorithm** By stacking $L$ heterogeneous temporal aggregation layers, we could derive the embedding for each node at each timestamp, denoted as $\mathbf{h}_v^{t,L}$. We then simply sum the node embedding of all timestamps as its final embedding: $\mathbf{h}_v = \sum_{t=1}^{T} \mathbf{h}_v^{t,L}$. HTGNN could be trained in an end-to-end manner with the labeled data at timestamp $T+1$, as following:

$$\mathcal{L} = \sum_{v \in \mathcal{V}_L} J(y_v, \hat{y}_v) + \lambda \|\mathbf{\Theta}\|_2^2, \hat{y}_v = \sigma(MLP(\mathbf{h}_v)). \tag{4.17}$$

where $J(\cdot)$ measures the loss between ground $y_v$ and the predicted score $\hat{y}_v$, $\|\mathbf{\Theta}\|_2^2$ is the L2-regularizer to prevent over-fitting. Depending on the goals of different tasks, $J(\cdot)$ could be set as cross-entropy loss for node classification and link prediction problems, or mean absolute error for regression problem.

## 5 Experiments

In this section, we conduct four sets of experiments to evaluate the performance of the proposed HTGNN.

**5.1 Datasets** We construct two HTGs from two different domains with distinct characteristics. OGBN-MAG dataset, an academic network with dynamically evolving heterogeneous structures, is used for link prediction task. COVID-19 dataset, an epidemiological network with constantly changing node features, is used for node regression task.

**OGBN-MAG:** The original OGBN-MAG dataset [1] is a static heterogeneous network composed of a subset of the Microsoft Academic Graph (MAG). We extract a HTG from OGBN-MAG consisting of 10 graph slices spanning from 2010 to 2019. We first select authors that consecutively publish at least one paper every year. We further collect these authors' affiliated institutions, published papers, and the papers' field of studies in each year to construct this HTG. Each graph slice is a heterogeneous graph that contains four types of nodes (paper, author, institution, and fields of study), and four types of relations among them (author-*affiliated with*-institution, author-*writes*-paper, paper-*cites*-paper, and paper-*has a topic of*-field of study).

**COVID-19:** The COVID-19 data is obtained from 1point3acres[1], which contains both state and county level daily case reports (e.g., confirmed cases, new cases, deaths, and recovered cases). We use the daily new COVID-19 cases as the time-series data for each state and county. We then build a HTG including 304 graph slices spanning from 05/01/2020 to 02/28/2021. Each graph slice is also a heterogeneous graph consisting of two types of nodes (state and county) and three types of relations between them, i.e., one administrative affiliation relation (state-*includes*-county) and two geospatial relations (state-*near*-state, county-*near*-county).

In OGBN-MAG, each paper comes with a 128-dimensional feature vector obtained by averaging the embeddings of words in its title and abstract. For other nodes, we run the metapath2vec [30] on each graph slice to generate the 128-dimensional node embeddings as their input features. For COVID-19, we attach each node in each graph slice with its daily new cases as the node feature. We split each dataset into training, validation, and testing sets with a ratio of 8:1:1. Statistics of these datasets are summarized in Table 1.

**5.2 Baselines** We compare HTGNN with three classes of state-of-the-art baselines.

**Neural Sequence Models**: This class of baselines is capable of capturing temporal dependencies. LSTM [31] is a type of recurrent neural network that learns order dependence of sequences. Transformer [29] handles sequences with global message routing, weighing the influence of different parts of the input.

**Static Graph Models**: We consider several static homogeneous/heterogeneous GNNs that depict spatial dependencies. GCN [7] and GAT [8] work on homogeneous graphs, where the neighbor information is aggregated through a mean function and a self-attention mechanism, respectively. For the heterogeneous GNNs, we choose RGCN [21] and HGT [10] that do not rely on metapaths. RGCN considers specialized transformation matrices for different types of relations. HGT applies the Transformer architecture to learn the mutual attention for each meta relation. In the experiment, we treat HGT as a static heterogeneous GNN as the relative temporal encoding is not applicable for HTGs.

[1] https://coronavirus.1point3acres.com/en

Table 2: Experimental results of different methods on OGBN-MAG (%) and COVID-19 datasets.

| Dataset | Metric | | Sequence Model | | Static Graph Model | | | | Dynamic Graph Model | | | | Ours |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | LSTM | TRFM | GCN | GAT | RGCN | HGT | CoGNN | DySAT | HDGAN | DyHATR | HTGNN |
| OGBN-MAG | AUC | Mean | 81.37 | 82.89 | 78.81 | 80.23 | 80.34 | 85.30 | 85.87 | 86.36 | 88.88 | 89.49 | **91.01** |
| | | SD | 0.42 | 0.36 | 1.53 | 2.07 | 2.21 | 1.20 | 1.52 | 0.24 | 0.73 | 0.65 | 0.77 |
| | AP | Mean | 78.56 | 79.81 | 76.46 | 77.58 | 78.11 | 82.68 | 83.11 | 83.83 | 86.18 | 86.24 | **89.18** |
| | | SD | 0.29 | 0.25 | 1.95 | 1.74 | 1.65 | 1.20 | 0.92 | 0.29 | 0.82 | 0.91 | 1.24 |
| COVID-19 | MAE | Mean | 720 | 691 | 846 | 821 | 833 | 805 | 653 | 672 | 656 | 643 | **555** |
| | | SD | 115 | 54 | 101 | 91 | 95 | 88 | 45 | 74 | 66 | 36 | 34 |
| | RMSE | Mean | 1504 | 1458 | 1674 | 1612 | 1640 | 1598 | 1298 | 1373 | 1303 | 1282 | **1136** |
| | | SD | 258 | 182 | 204 | 211 | 198 | 200 | 60 | 96 | 81 | 59 | 65 |

**Dynamic Graph Models**: We select one spatial-temporal GNN, one dynamic homogeneous GNN, and two dynamic heterogeneous GNNs as baselines. CoGNN [6] applies a multilayer perceptron to process time-series node features and uses GCN [7] with skip connections for spatial information aggregation. DySAT [12] employs self-attention to aggregate structural neighborhood and temporal dynamics for node representation learning. HDGAN [28] combines heterogeneous attention and Hawkes process to model graph heterogeneity and dynamics. We replace the heterogeneous attention module with HGT [10] to avoid incorporating metapaths. DyHATR [15] uses hierarchical attention to learn heterogeneous information and incorporates RNNs with temporal attention to capture temporal dependencies.

**5.3 Implementation Details** For those models designed for homogeneous graphs, we ignore the graph heterogeneity and directly feed the whole graph into the learning algorithms. We employ Adam optimizer with learning rate set to 5e-3, and weight decay set to 5e-4. For other parameters, we set dropout rate to 0.2, GNN layer to 2, hidden embedding dimension to 32 for OGBN-MAG and 8 for COVID-19, respectively; and we also use ReLU as the activation function. We train all the models with a fixed 500 epochs and use an early stopping strategy with a patience of 50. All models are trained for five times, and the mean and standard deviation of test performance are reported. All baselines and the proposed HTGNN are implemented with Python 3.7.10, PyTorch 1.8.1 and Deep Graph Library (DGL) 0.6.0. Experiments are conducted on a machine equipped with i9-9900K processor, two RTX 2080Ti graphic cards, and 64 GB of RAM.

**5.4 Link Prediction** We perform link prediction on OGBN-MAG dataset to evaluate different methods. We split the dataset into three sets with a ratio of 8:1:1. Specifically, the graphs of 2010-2017 are used for training, the graph of 2018 for validation, and the graph of 2019 for testing. We frame our task to new co-author link prediction. The new co-author relation is defined as the co-author link that exists in year $T+1$ but not in year $T$. We randomly select 10% new co-author links as positive samples. Following the standard manner of learning-based link prediction, we randomly sample the same number of nonexistent co-author links as negative samples. We set the time window size to 3, which means, to predict the co-author relation in next year, we consider the HTG of the past three years. Note that, for static graph models without considering temporal dependency, we simply set the time window to 1. For a pair of authors, after obtaining the embeddings via HTGNN, we feed their concatenation into Eq. (4.17) for training with the cross-entropy loss. Similar to [32, 33], we adopt the widely used AUC score and AP score (Average Precision) to measure the co-author link prediction performance.

The experimental results with mean performance and their standard deviations are reported in Table 2. We have the following conclusions by analyzing the results: (1) Both sequence and static graph models could achieve satisfactory results, which indicates that the temporal and spatial dependencies depicted by these two types of methods contribute to the co-author link prediction problem. (2) Dynamic graph models improve the performance by taking the information in both spatial and temporal domains into consideration. (3) GNNs designed for heterogeneous graphs perform better than homogeneous GNNs, which demonstrates the advantage of incorporating graph heterogeneity. (4) Our proposed HTGNN that could jointly model the heterogeneous spatial dependencies and temporal dimensions consistently outperforms all baselines.

**5.5 Node Regression** The node regression task is conducted on COVID-19 dataset. We aim to perform state-level daily new case forecasting. We also split the dataset into training, validation, and testing sets with a ratio of 8:1:1. Specifically, 05/01/2020-12/30/2020 are used for training, 12/31/2020-01/29/2021 for validation, and 01/30/2021-02/28/2021 for testing. In this task, we set the time window to 7 (using the past one-week historical data for forecasting). Similarly, we set it to 1 for static graph models. As suggested in [11, 5], MAE (Mean Absolute Errors) and RMSE (Root Mean

Squared Errors) are adopted to measure the performance. We report the average MAE and RMSE of 54 states in the US. The experimental results with mean and standard deviation reported are demonstrated in Table 2. Besides some similar conclusions drawn from the link prediction task, we notice that sequence models yield better performances than static graph models. This phenomenon indicates that the temporal domain contributes more to the COVID-19 forecasting task compared to the spatial domain. This is because the graph structures remain unchanged in this case; however, the node features (i.e., daily new cases) have a remarkable change over time.

**5.6 Ablation Study** In HTGNN model heterogeneous spatial dependen dimensions jointly. To evaluate this lish two HTGNN variants for compa these two types of dependencies ser processes spatial dependencies first pendencies later by first performing and inter-relation aggregations in eac applying across-time aggregation acro $\text{HTGNN}_{TS}$ analyzes these two doma der by first employing across-time a temporal domain and then conductin tial aggregation on the last graph s tal results shown in Table 3 demo $\text{HTGNN}_{ST}$ outperforms $\text{HTGNN}_{TS}$ dataset but is less effective on Covid is because $\text{HTGNN}_{ST}$ emphasizing m main fits better with OGBN-MAG evolving graph structures. On the co paying more attention to the tempor suitable for COVID-19 with changing HTGNN achieves better performance than these two variants in both datasets, proving that our proposed holistic model is agnostic to graph characteristics and delivers superior performance.

We then perform additional ablation studies to evaluate the three major components in HTGNN: intra-relation, inter-relation, and across-time aggregation modules. Accordingly, we prepare three variants to examine the effect of each component. HTGNN w/o Intra replaces the intra-relation aggregation module in each layer with a mean pooling mechanism. HTGNN w/o Inter replaces the inter-relation aggregation module in each layer with a mean pooling mechanism. HTGNN w/o Across replaces the across-time aggregation module in each layer with a mean pooling mechanism. Experimental results are shown in Figure 3. From Figure 3, we observe that HTGNN equipped with three components achieves the best performance, which proves that each component makes its contribution to the final performance. It is also worth noting that HTGNN w/o Intra and HTGNN w/o Inter work better than HTGNN w/o across on COVID-19 dataset but yield worse results on OGBN-MAG dataset. We owe this to the distinct characteristics of different datasets. In particular, for OGBN-MAG, the heterogeneous graph structures evolve dynamically, increasing the difficulty in capturing spatial dependencies. In contrast, for COVID-19, the temporal dependencies are relatively harder to capture as the node features change constantly, while the graph structures remain unchanged.

Table 3: Evaluation of joint modeling on HTGs.

| Dataset | OGBN-MAG | | COVID-19 | |
|---|---|---|---|---|

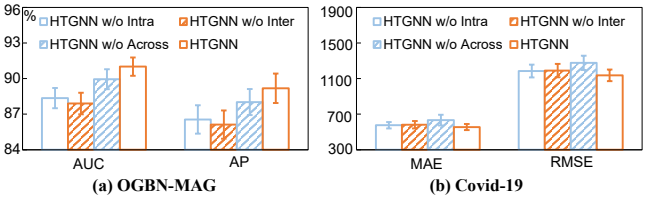


Figure 3: Evaluation of each component in HTGNN.

**5.7 Parameter Sensitivity** In this section, we investigate HTGNN's sensitivity to key hyper-parameters.

**Model depth.** We vary the model depth (the number of aggregation layers) from 1 to 5 to examine the model's performance on two datasets. The experimental results with mean and standard deviation reported are shown in Figure 4 (a)-(b). We can see that with the increase of model depth, the performance of HTGNN first improves then starts to decrease gradually. This phenomenon is attributed to the oversmoothing problem.

**Embedding dimension.** We vary the embedding dimension from 4 to 64 for OGBN-MAG and 2 to 32 for COVID-19 to investigate its influence. Comparison results are illustrated in Figure 4 (c)-(d). We observe that increasing the embedding dimension initially improves the performance since a larger dimension can preserve more information. However, when using a too large dimension, the model would suffer the overfitting problem, which results in reduced performance.

**Time window size.** We validate the effect of time window size by ranging it from 2 to 6 for OGBN-MAG and 5 to 13 for COVID-19, respectively. The results are shown in Figure 4 (e)-(f). We can see that a large time window size boosts the performance as more historical information is included. However, further enlarging the window size yields a fluctuating performance.

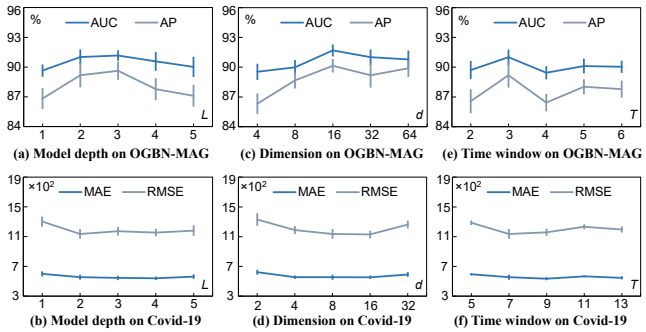


Figure 4: Parameter sensitivity analysis.

## 6 Conclusion

In this paper, we study the representation learning problem on heterogeneous temporal graphs (HTGs), a general concept for modeling heterogeneous and constantly evolving graph data. We further propose heterogeneous temporal graph neural network (HTGNN), a holistic framework tailored heterogeneity with evolution in time and space for HTG representation learning. In particular, HTGNN consists of several heterogeneous temporal aggregation layers, each of which employs a hierarchical aggregation mechanism, including intra-relation, inter-relation and across-time aggregation modules, to jointly model heterogeneous spatial dependencies and temporal dimensions. Extensive experiments are conducted on two built HTGs: OGBN-MAG with dynamically evolving heterogeneous structures, and COVID-19 with constantly changing node features. Promising results demonstrate the great performance of HTGNN in comparison with state-of-the-art baselines.

## Acknowledgment

This work is partially supported by the NSF under grants IIS-2107172, IIS-2140785, CNS-1940859, CNS-1814825, IIS-2027127, IIS-2040144, IIS-1951504 and OAC-1940855, the NIJ 2018-75-CX-0032.

## References

[1] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[2] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019.

[3] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *TKDE*, vol. 30, no. 12, 2018.

[4] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *WWW*, 2019, pp. 417–426.

[5] S. Deng, S. Wang, H. Rangwala, L. Wang, and Y. Ning, "Cola-gnn: Cross-location attention based graph neural networks for long-term ili prediction," in *CIKM*, 2020, pp. 245–254.

[6] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O'Banion, "Examining covid-19 forecasting using spatio-temporal graph neural networks," *arXiv preprint arXiv:2007.03113*, 2020.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[9] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW*, 2020, pp. 2331–2341.

[10] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, 2020, pp. 2704–2710.

[11] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[12] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *WSDM*, 2020, pp. 519–527.

[13] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *WWW*, 2018, pp. 969–976.

[14] Y. Yin, L.-X. Ji, J.-P. Zhang, and Y.-L. Pei, "Dhne: Network representation learning method for dynamic heterogeneous networks," *IEEE Access*, vol. 7, pp. 134 782–134 792, 2019.

[15] H. Xue, L. Yang, W. Jiang, Y. Wei, Y. Hu, and Y. Lin, "Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn," *arXiv preprint arXiv:2004.01024*, 2020.

[16] W. Luo, H. Zhang, X. Yang, L. Bo, X. Yang, Z. Li, X. Qie, and J. Ye, "Dynamic heterogeneous graph neural network for real-time event prediction," in *KDD*, 2020, pp. 3213–3223.

[17] Y. Xie, Z. Ou, L. Chen, Y. Liu, K. Xu, C. Yang, and Z. Zheng, "Learning and updating node embedding on dynamic heterogeneous information network," in *WSDM*, 2021.

[18] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv preprint arXiv:1906.00121*, 2019.

[19] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, and J. Ye, "Heteta: Heterogeneous information network embedding for estimating time of arrival," in *KDD*, 2020, pp. 2444–2454.

[20] L. Yang, Z. Xiao, W. Jiang, Y. Wei, Y. Hu, and H. Wang, "Dynamic heterogeneous graph embedding using hierarchical attentions," in *ECIR*, 2020, pp. 425–432.

[21] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018, pp. 593–607.

[22] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *KDD*, 2019.

[23] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *CIKM*, 2018, pp. 2077–2085.

[24] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[25] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, and S. Mou, "Dynamic heterogeneous information network embedding with meta-path based proximity," *TKDE*, 2020.

[26] 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.

[27] Q. Li, Y. Shang, X. Qiao, and W. Dai, "Heterogeneous dynamic graph attention network," in *ICKG*, 2020, pp. 404–411.

[28] Y. Ji, T. Jia, Y. Fang, and C. Shi, "Dynamic heterogeneous graph embedding via heterogeneous hawkes process," in *ECML-PKDD*, 2021, pp. 388–403.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[30] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017, pp. 135–144.

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

[32] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *arXiv preprint arXiv:1802.09691*, 2018.

[33] H. Fan, F. Zhang, Y. Wei, Z. Li, C. Zou, Y. Gao, and Q. Dai, "Heterogeneous hypergraph variational autoencoder for link prediction," *TPAMI*, 2021.