Heterogeneous Graph Neural Network on Semantic Tree

Mingyu Guan[1], Jack W. Stokes[2], Qinlong Luo[2], Fuchen Liu[2], Purvanshi Mehta[3], Elnaz Nouri[2], Taesoo Kim[1]

[1]Georgia Institute of Technology,
[2]Microsoft Corporation,
[3]Lica World Inc
mingyu.guan@gatech.edu, {jstokes, qinlongluo, fuchen.liu}@microsoft.com, purvanshi@lica.world, elnaaz@gmail.com, taesoo@gatech.edu

Abstract

The recent past has seen an increasing interest in Heterogeneous Graph Neural Networks (HGNNs), since many real-world graphs are heterogeneous in nature, from citation graphs to email graphs. However, existing methods ignore a tree hierarchy among metapaths, naturally constituted by different node types and relation types. In this paper, we present HETTREE, a novel HGNN that models both the graph structure and heterogeneous aspects in a scalable and effective manner. Specifically, HETTREE builds a semantic tree data structure to capture the hierarchy among metapaths. To effectively encode the semantic tree, HETTREE uses a novel subtree attention mechanism to emphasize metapaths that are more helpful in encoding parent-child relationships. Moreover, HETTREE proposes carefully matching pre-computed features and labels correspondingly, constituting a complete metapath representation. Our evaluation of HETTREE on a variety of real-world datasets demonstrates that it outperforms all existing baselines on open benchmarks and efficiently scales to large real-world graphs with millions of nodes and edges.

Code — https://github.com/microsoft/HetTree

1 Introduction

Graph neural networks (GNNs) have been widely explored in a variety of domains from social networks to molecular properties (Pal et al. 2020; Park et al. 2019; Sun, Dai, and Yu 2022), where graphs are usually modeled as homogeneous graphs. However, real-world graphs are often heterogeneous in nature (Hu et al. 2020a; Lv et al. 2021). For example, as shown in Figure 1(a), a heterogeneous email graph can include multiple types of nodes - Domain, Sender, Recipient, Message, and IP Address - and the relations among them. Moreover, multiple relations can exist between two entities in complex heterogeneous graphs. For example, a *Sender* node can be a P1 sender and/or P2 sender of a *Message*, where the P1 sender denotes the entity that actually sent the message while an email application displays the P2 sender as the "From" address. For example, if Bob sends an email on behalf of Alice, the email appears to originate from Alice, making Alice the P2 sender and Bob the P1 sender. As a result, there are two relations between Sender and Message in Figure 1(a), *p1_sends* and *p2_sends*.

To better understand real-world heterogeneous graphs, various heterogeneous graph neural networks (HGNNs) have been proposed. The most well-known approaches are the metapath-based methods (Schlichtkrull et al. 2017; Wang et al. 2019; Fu et al. 2020), which first aggregate neighbor representations along each metapath at the node level and then aggregate these representations across metapaths at the metapath (semantic) level. However, metapath-based approaches often involve manual effort to select a subset of meaningful metapaths, because the node-level aggregation along each metapath is computationally expensive (Fu et al. 2020; Wang et al. 2019). Other models that do not apply the metapath method, such as HetGNN (Zhang et al. 2019) and HGT (Hu et al. 2020b), carefully encode representations for different node types and/or relation types in heterogeneous graphs. These fine-grained embedding methods often utilize multi-layer message-passing techniques as in traditional GNNs, thus facing scalability issues. To efficiently model real-world web-scale graphs, researchers and practitioners have explored various ways to scale HGNNs. Sampling-based methods sample sub-graphs with different strategies (Hu et al. 2020b; Zhang et al. 2019), while others use model simplification to execute feature propagation as a pre-processing stage before training (Yu et al. 2020; Yang et al. 2022).

However, existing HGNNs fail to account for a *tree hierarchy among the metapaths*. A metapath represents an ordered sequence of composite relationships connecting distinct or identical node types (Definition 3.2). For instance, as illustrated in Figure 1(a), the metapath $Sender \xrightarrow{p1\_sends} Message \xrightarrow{is\_sent\_from} IP$ is intuitively more closely associated with $Sender \xrightarrow{p1\_sends} Message$ than with $Sender \xrightarrow{s\_has\_domain\_of} Domain$ due to a greater overlap in node types and relationships. This overlap can be conceptualized as a parent-child relationship, where the parent metapath serves as a prefix to its child metapaths. Consequently, these parent-child relationships naturally form a tree hierarchy among the metapaths.

The exploration of tree structures in heterogeneous graphs is not a new topic, as several tree-based HGNNs have investigated this concept. However, existing tree-based HGNNs (Xu et al. 2021; Wu and Wang 2022; Qiao et al. 2020) pri-

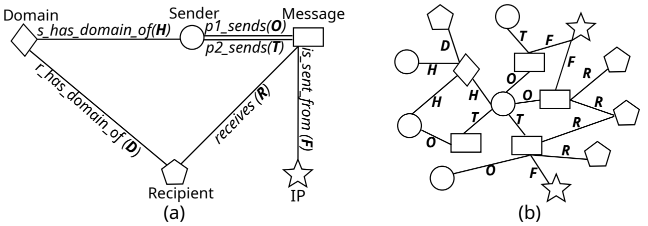

Figure 1: (a) Relational scheme of a heterogeneous email graph (b) An example of the email graph.

marily utilize tree structures to capture the *local topology of nodes*. These approaches subsequently employ attention mechanisms for semantic aggregation to integrate information across metapaths. Despite these efforts, they do not account for the parent-child relationships or hierarchical organization inherent among metapaths (Section 2).

Moreover, to augment the data usage, label utilization has been widely adopted in GNNs (Sun, Gu, and Hu 2021; Shi et al. 2020; Zhang et al. 2022; Yu et al. 2022). These methods leverage ground truth labels from the training set and propagate them through the graph structure as inputs to the model. However, existing approaches either completely separate feature learning from label learning, combining them only in the final stage, or treat feature and label vectors equivalently by projecting them to the same latent space. While such strategies may be effective for homogeneous graphs, they are less suitable for heterogeneous graphs, where features and labels can propagate along distinct metapaths, and *features and labels through the same metapath are more related*.

In this paper, we present HetTree, a scalable HGNN that extracts a unified tree representation on metapaths, i.e., *semantic tree*, from the input graph and proposes a novel tree aggregation with subtree attention to encode the resulting semantic tree, in which propagated labels are carefully matched with corresponding features based on metapaths.

To scale efficiently on web-scale graphs, HetTree follows the model simplification approach that simplifies heterogeneous feature aggregation as a pre-processing stage. Meanwhile, label aggregation of the target node types for each metapath is also executed. HetTree then builds a semantic tree to capture the hierarchy among metapaths. Instead of separating feature learning from label learning (Sun, Gu, and Hu 2021; Zhang et al. 2022) or treating features and labels equally by projecting them to the same latent space (Yang et al. 2022), HetTree proposes to match them carefully on corresponding metapaths, which provides more accurate and richer information between node features and labels.

To encode the resulting semantic tree, HetTree uses a novel *subtree attention* mechanism to emphasize children nodes that are more helpful in encoding parent-child relationships. Existing tree encoding techniques (Xu et al. 2021; Wu and Wang 2022; Qiao et al. 2020) aggregate children nodes by weighting the contribution of children nodes based on similarity to the parent node. However, in the semantic tree, this tree encoding fails to capture the entire parent-child hierarchy by only considering the parent node. Hence, subtree attention in HetTree models a broader parent-child structure while enhancing correlations, bringing a better representation for each metapath.

In summary, we make the following contributions.

- We observe that existing HGNNs ignore the hierarchy of the metapath features. HetTree takes a radically new approach to encoding metapath hierarchy by building a *semantic tree* for both pre-computed features and labels.
- HetTree proposes a novel tree aggregation with *subtree attention* to encode the semantic tree structure. For better label usage, HetTree matches pre-computed features and labels correspondingly, which constitutes a complete representation of a metapath.
- We conduct extensive experiments on five open graph datasets as well as a real-world commercial email dataset. The results demonstrate that HetTree can outperform the state-of-the-art architectures on all datasets with low computation and memory overhead.

2 Related Work

Graph Neural Networks. Graph Neural Networks (GNNs) are neural networks that take input structured as graphs. The fundamental task in a GNN is to generate the representation of graph entities, such as nodes and edges, in a d-dimensional space, referred to as the embedding of the entity. To generate the embedding, GNNs usually use a multi-layer feature propagation followed by a neural network to combine structural information from the graph structure and the input features. Various GNN architectures exist today, but they differ in how the information is aggregated and transformed (Hamilton, Ying, and Leskovec 2017; Defferrard, Bresson, and Vandergheynst 2016; Veličković et al. 2018; Hamilton, Ying, and Leskovec 2017). Nevertheless, a main problem with these classic GNNs is that they are hard to scale due to the feature propagation performed at each layer of the neural networks. Hence, many sampling methods (Hamilton, Ying, and Leskovec 2017; Chen, Ma, and Xiao 2018; Zou et al. 2019a,b) have been proposed to reduce both computation and memory complexity by using only a subset of nodes and edges. Besides sampling, other methods (Wu et al. 2019; Frasca et al. 2020; Zhu and Koniusz 2020) simplify models by making feature propagation an offline stage, so that this computation-intensive process only needs to be executed once and is not involved during the training.

Heterogeneous Graph Neural Networks. To extend GNN from homogeneous graphs to heterogeneous graphs, various heterogeneous GNN architectures (HGNNs) have been explored. The most general approach in HGNNs is the so-called metapath-based method (Fu et al. 2020; Wang et al. 2019), where the feature propagation is performed based on semantic patterns and an attention mechanism is usually applied at both the node level for each metapath and the semantic level across metapaths. Other models (Schlichtkrull et al. 2017; Zhang et al. 2019; Hu et al. 2020b; Lv et al. 2021) encode graph heterogeneity at a more fine-grained level using the multi-layer, message-passing framework common in GNNs, where different weights are learned for distinct entity types.

However, HGNNs also inherit the scalability problem from traditional homogeneous GNNs. Hence, sampling (Hu et al.

2020b) and model simplification (Yu et al. 2020) have also been explored in the heterogeneous graph learning domain. NARS (Yu et al. 2020) first applies the scaling approach proposed by SIGN (Frasca et al. 2020) on heterogeneous graphs, which samples multiple relational subgraphs using different sets of relations and then treats them as homogeneous graphs. SeHGNN (Yang et al. 2022) computes averaged metapath features separately and applies Transformer-like attention to learn metapath features. However, simply applying the Transformer ignores the hierarchy among metapath features and thus results in sub-optimal results.

Tree-based HGNNs. Recent work (Zhang, Ying, and Lauw 2023) utilizes a topic tree as a regularizer, i.e., log-likelihood terms, for text decoding on document graphs. For general heterogeneous graphs, a few HGNNs have explored tree structure based on the local topology of nodes. T-GNN (Qiao et al. 2020) and SHGNN (Xu et al. 2021) construct hierarchical tree structures at the node level, where each tree represents a metapath and each level of the tree contains nodes of a certain type. Similarly, HetGTCN (Wu and Wang 2022) also constructs a tree hierarchy for each node, where tree nodes at the k^{th} level are *k-hop* neighbors of the root node. To encode tree-structured data, i.e. parent-child relationships among tree nodes, they either use a weighted sum aggregator (Qiao et al. 2020; Wu and Wang 2022) or compute weights using an attention mechanism for each parent-child pair (Xu et al. 2021; Wu and Wang 2022). These methods utilize the tree structure to aggregate neighbor information for each metapath first and then use semantic attention (Wang et al. 2019) to aggregate metapath representations to obtain the final node representation. However, *the tree hierarchy among metapaths* has not been explored.

Label Utilization. Label utilization has been commonly applied in graph representation learning. In general, partially observed labels in the training set are propagated through the network structure to generate label representations, combined with the feature representations to generate the final representations of graph entities (ZhuΓ́ and GhahramaniΓ́H 2002; Wang et al. 2021; Sun, Gu, and Hu 2021). To avoid label leakage and overfitting, UniMP (Shi et al. 2020) randomly masks the training nodes for each epoch. GAMLP (Zhang et al. 2022) modifies label propagation with residual connections to each hop to alleviate the label leakage issue.

However, these methods either completely separate feature learning and label learning and only combine them at the end, or they simply add the features and label vectors together as propagation information, which may result in good performance in homogeneous graphs but not in the case of heterogeneous graphs, since the features and labels are related by the corresponding metapaths.

3 Preliminary

In this section, we provide formal definitions of important terminologies related to HETTREE.

Definition 3.1. Heterogeneous Graph. A heterogeneous graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{R})$, where each node $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ are associated with a node mapping function $\tau(v) : \mathcal{V} \rightarrow \mathcal{O}$ from node set $\mathcal{V}$ to node type set $\mathcal{O}$, and a edge mapping function $\phi(e) : \mathcal{E} \rightarrow \mathcal{R}$ from edge set $\mathcal{E}$ to relation set $\mathcal{R}$, respectively.

Example 3.1. Figure 1(a) shows the relational scheme of a heterogeneous email graph while Figure 1(b) shows an illustrative example. It is composed of five types of nodes: $Domain$,$Sender$, $Message$, $Recipient$, IP, and six types of relations: *s_has_domain_of(**H**)*, *r_has_domain_of(**D**)*, *p1_sends(**O**)*, *p2_sends (**T**)*, *receives(**R**)*, *is_sent_from(**F**)*.

Definition 3.2. Metapath. A metapath P is a path that describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between node types O_1 and O_{l+1}, where $\circ$ denotes the composition operator on relations. P is denoted as $O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} O_{l+1}$, which is abbreviated as $R_1 R_2 \cdots R_l$. A special metapath where no relation is present but includes only a node type O is simply denoted as O^{init} (abbreviated as $init$ when O is specified). The set of metapaths ending with node type O excluding $P = O^{init}$ is denoted as $\mathcal{P}_O$. The set of metapaths up to hop k is denoted as $\mathcal{P}^k$, where $l \leq k, \forall P = R_1 R_2 \cdots R_l \in \mathcal{P}^k$. Metapath-based neighbors $\mathcal{N}_P^v$ of node v along metapath P are the set of nodes that are connected with node v via metapath P. Note that when $\tau(v) = O$ and P ends with O, $\mathcal{N}_P^v$ can include v itself.

Remark 3.3. Most of metapath-based HGNNs denote a metapath $P = O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} O_{l+1}$ as $O_1 O_2 \cdots O_{l+1}$ for short. We note that this notation fails to differentiate metapaths that have different relation compositions but the same node types along the metapaths, as multiple relations can be present between two node types in a heterogeneous graph.

Example 3.3. In Figure 1, a sender can be connected to a recipient through three 2-hop metapaths: $Sender \xrightarrow{O} Message \xrightarrow{R} Recipient$ (OR), $Sender \xrightarrow{T} Message \xrightarrow{R} Recipient$ (TR), $Sender \xrightarrow{H} Domain \xrightarrow{D} Recipient$ (HD). Moreover, let $S = \{OR, TR, HD\}$, then we have $S \subset \mathcal{P}^k$, for any $k \geq 2$.

Definition 3.4. Semantic Tree. A semantic tree T_O with depth of k, for node type O, is composed of tree nodes $\mathcal{C} = \{C_P, \forall P \in \mathcal{P}^k\}$ and relation edges $\mathcal{R}$. The root node represents the metapath $P = O^{init}$, denoted as $C_{O^{init}}$ (abbreviated as C_{init} when O is specified). A non-root node $C_{R_1 R_2 \cdots R_l}$ represents the metapath from the root node C_{init} to them via relation edges $R_1 R_2 \cdots R_l$. The root node C_{init} is the parent of all nodes C_{R_1} at depth 1 of the semantic tree T_O. A node $C_{R_1 R_2 \cdots R_l}$ with depth ≥ 2 has a parent node $C_{R_1 R_2 \cdots R_{l-1}}$, and they are connected by edge R_l.

Example 3.4. A semantic tree with depth of 2 for *Sender* nodes in the heterogeneous email graph is shown in Figure 2, where the root node C_{init} represents metapath *init* and other nodes represent the metapath from the root node to them. For example, node C_{OF} is connected with the root node C_{init} by relation edges O and F in order.

4 Methodology

In this section, we describe a novel heterogeneous tree graph neural network, HETTREE, for scalable and effective heterogeneous graph learning. HETTREE consists of three major

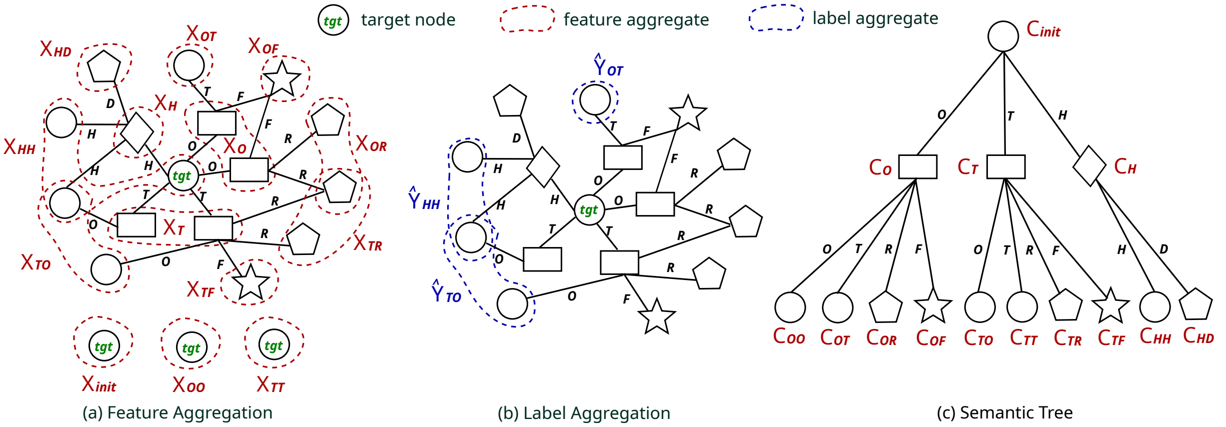

Figure 2: (a) The offline process of feature aggregation. The center node is the target $Sender$ node and features are aggregated for all metapaths $\mathcal{P}^k$ up to hop k, where $k = 2$ in this example. (b) The offline process of label aggregation on partially observed labels in the training set. (c) Semantic tree with height of k for *Sender* nodes in the email graph. A tree node C_P represents metapath P, where $P \in \mathcal{P}^k$.

components: offline feature aggregation and semantic tree construction (Section 4.1), metapath feature transformation (Section 4.2), and semantic tree aggregation (Section 4.3).

4.1 Offline Aggregation and Semantic Tree Construction

As a pre-processing stage, node features and labels are aggregated to prepare for training. In node classification tasks, initial node features are normally associated with every node (if not, we can use external graph embedding algorithms like ComplEx (Trouillon et al. 2017) to generate them), while labels are often only associated with nodes of target node types that need to be classified. For example, in the email dataset we collect, the task is to classify *Sender* nodes as a compromised email account or not, and only *Sender* nodes are associated with labels. Moreover, as mentioned in Section 2, only labels of training nodes can be used as part of the input features, and the problem of label leakage in label utilization should be well addressed. We also construct a novel semantic tree structure to organize the aggregation results and capture the hierarchy of metapaths, which can be leveraged in semantic tree aggregation (Section 4.3).

Feature Aggregation Figure 2(a) shows the process of offline feature aggregation. Unlike existing metapath-based methods (Wang et al. 2019; Fu et al. 2020), where feature aggregation is involved with model learning such as projection and attention, HETTREE performs feature aggregation as a pre-processing step. Existing methods manually select metapaths with domain knowledge. The choice saves computation complexity but also results in information loss. As the feature aggregation happens offline and involves no parameter learning, which is much less expensive than existing approaches, we use all metapaths up to hop k, where k is a user-defined parameter. For example, when $k = 2$ as shown in Figure 2(a), feature aggregation is conducted on 14 metapaths ($init$, O, T, H, OO, OT, OR, OF, TT, TO, TR, TF, HH, HD). Possible aggregators include but are not limited to $mean$, sum, max, and min, and we use the $mean$ aggregator in this paper for both feature and label aggregation. For each node v, we compute a set of aggregated features $\mathcal{X}^v$:

$$\mathcal{X}^v = \{X_P^v = agg(\{x^u, \forall u \in \mathcal{N}_P^v\}), \forall P \in \mathcal{P}^k\} \quad (1)$$

where X_P^v represents the aggregated feature for node v along metapath P, and agg is the aggregation function ($mean$ by default).

Label Aggregation Figure 2(b) shows the process of offline label aggregation. The label aggregation process is very similar to the feature aggregation process described in Section 4.1, except for two differences: first, since only labels from nodes with target node type O_{tgt} (node type to be classified) in the training set can be used, the label aggregation is only conducted for metapaths $\mathcal{P}^k_{O_{tgt}}$ where they end at node type O_{tgt}; second, feature aggregation applies to all nodes in $\mathcal{N}_P^v$ including v itself, while v is excluded during label aggregation to avoid label leakage. For example, when $k = 2$ as shown in Figure 2(b), label aggregation is conducted on 3 metapaths (OT, TO, HH) excluding the center target node, respectively. Note that only labels from nodes in the training set are used for label aggregation, and zero vectors are used for nodes in the non-training set. Specifically, for each node v, we compute a set of aggregated features $\hat{\mathcal{Y}}^v$:

$$\hat{y^v} = \begin{cases} y^v, & \text{if } v \in training\_set \\ \mathbf{0}, & \text{otherwise} \end{cases}$$
$$\hat{\mathcal{Y}}^v = \{\hat{Y}_P^v = agg(\{\hat{y}^u, \forall u \in \mathcal{N}_v^P \setminus \{v\}\}), \forall P \in \mathcal{P}^k_{O_{tgt}}\} \quad (2)$$

where y^v is the ground truth label for node v, $\hat{y}^v$ is the label used in label aggregation for node v, and $\hat{Y}_P^v$ is the aggregated label along metapath P for node v.

Semantic Tree Construction We can construct a semantic tree T_O for node type O with tree nodes $\{C_P, \forall P \in \mathcal{P}^k\}$. C_{init} is the root node and a non-root node $C_{R_1 R_2 \cdots R_l}$ represents the metapath from the root node C_{init} to itself via relation edges $R_1 R_2 \cdots R_l$. The parent of all 1-hop tree nodes

C_{R_1} is C_{init}, which are at depth 1 of T_O. Starting from depth 2, $C_{R_1 R_2 \cdots R_l}$ is connected with its parent node $C_{R_1 R_2 \cdots R_{l-1}}$ via edge R_l. By constructing the semantic tree, the hierarchy among metapaths can be captured, which provides the model structural information of the metapaths. Moreover, the semantic tree is also used as the underlying data structure for semantic tree aggregation discussed in Section 4.3, where the metapath features are aggregated following the tree structure in a bottom-up way. Figure 2(c) shows an illustration of the semantic tree for the *Sender* node type.

4.2 Metapath Feature Transformation

After obtaining aggregated features and labels for metapaths, we transform them to the same latent space. This is due to the aggregated features for metapaths being generated from raw features of metapath-based neighbors with different node types, which may have different initial spaces. Instead of separating the transformation of features and labels as in existing methods (Sun, Gu, and Hu 2021; Zhang et al. 2022), HETTREE automatically matches and concatenates ($\|$) the aggregated features and labels of the same metapath P for $P \in \mathcal{P}_{O_{tgt}}$. This gives the model more accurate label information of its metapath-based neighbors by designating the aggregated labels with corresponding metapaths. Specifically, for all $P \in \mathcal{P}^k$, we compute the metapath features $\mathcal{M}$ as

$$\mathcal{M} = \{M_P = \begin{cases} MLP(X_P \parallel \hat{Y}_P), & \text{if } P \in \mathcal{P}^k_{O_{tgt}} \\ MLP(X_P), & \text{otherwise} \end{cases}\}. \quad (3)$$

4.3 Semantic Tree Aggregation

As discussed in Section 4.1, we construct a semantic tree T, and each tree node C_P represents a metapath P, where $P \in \mathcal{P}^k$. Since we also obtain metapath features $\mathcal{M}$, we can associate each tree node C_P with M_P correspondingly. Note that the semantic tree structure is the same for all nodes with the same node type in a heterogeneous graph, so the target nodes (to be classified) can easily be batched. We now have tree-structured metapath features, and the hierarchical relationship between metapath features needs to be well modeled when aggregating them.

The tree aggregation in HETTREE is conducted in a bottom-up fashion. As it gets closer and closer to the target node as the process proceeds, the semantic tree aggregation can gradually emphasize those metapaths that contribute more to the local subtree structure, i.e., the parent-child relationship. To calculate the encoded representation Z_P for each metapath node in the semantic tree, HETTREE applies a novel *subtree attention* mechanism to aggregate the children nodes thus encoding the local subtree structure. Unlike existing tree encoding methods (Tai, Socher, and Manning 2015; Qiao et al. 2020; Xu et al. 2021; Wu and Wang 2022) that use either a simple weighted-sum aggregator or attention mechanism to emphasize parent tree nodes, HETTREE proposes the subtree attention to encode both the parent and children representation and uses it to emphasize the hierarchical correlation between metapaths. Specifically, let $\mathcal{P}_P^{child}$ be the set of metapaths that $\{C_Q, Q \in \mathcal{P}_P^{child}\}$ are the set of children nodes of C_P in the semantic tree, HETTREE computes a *subtree reference* as $S_P = MLP(M_P \parallel \sum_{Q \in \mathcal{P}_P^{child}} M_Q)$, for each metapath P in the semantic tree. Then, the weight coefficient a_Q of each children node C_Q can be calculated as:

$$\alpha_Q = \frac{exp(\delta(W_P \cdot [S_P \parallel Z_Q]))}{\sum_{B \in \mathcal{P}_P^{child}} exp(\delta(W_P \cdot [S_P \parallel Z_B]))}. \quad (4)$$

where δ is the activation function, W_P is a learnable projection vector for metapath P and $\|$ stands for concatenation. Then, we can finally compute the encoded representation Z_P for parent node C_P by aggregating encoded representation of children nodes as:

$$Z_P = M_P + \delta(\sum_{Q \in \mathcal{P}_P^{child}} \alpha_Q \cdot Z_Q). \quad (5)$$

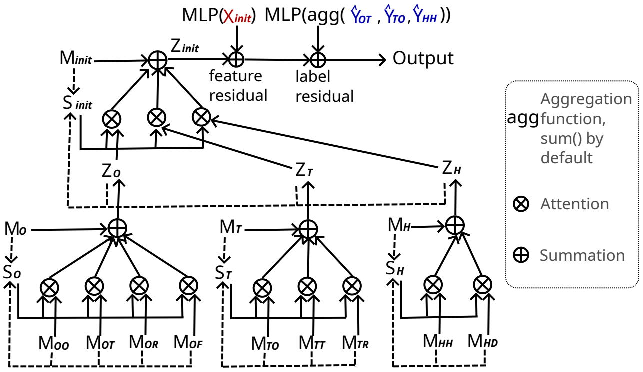

Figure 3: Semantic tree aggregation in HETTREE.

After the semantic tree aggregation has finished from bottom to top, the sum of metapath representations will be used as the final representation of the semantic tree. Moreover, we add a feature residual and a label residual to further emphasize the initial features and labels aggregated from the metapath-based neighbors. Specifically,

$$\begin{aligned} Y_{pred} \quad &= \quad MLP(\sum_{P \in \mathcal{P}^k} Z_P) + MLP(X_{init}) \\ &\quad + MLP(agg.(\{\hat{Y}_P, \forall P \in \mathcal{P}^k_{O_{tgt}}\})). \end{aligned} \quad (6)$$

where $agg.$ is an aggregation function, which can be $mean$, sum, max, min, etc. An illustration of the semantic tree aggregation process is shown in Figure 3, following the same example in Figure 2.

5 Experiments

We conduct extensive experiments on six heterogeneous graphs to answer the following questions.

Q1. How does HETTREE compare to the state-of-the-art overall on open benchmarks?

Q2. How does HETTREE perform in a practical compromised account detection on a noisy email graph?

Q3. How does each component of HETTREE contribute to the performance gain?

Q4. Is HETTREE practical w.r.t. running time and memory usage?

	DBLP		IMDB		ACM		Freebase	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
RGCN	91.52±0.50	92.07±0.50	58.85±0.26	62.05±0.15	91.55±0.74	91.41±0.75	46.78±0.77	58.33±1.57
HAN	91.67±0.49	92.05±0.62	57.74±0.96	64.63±0.58	90.89±0.43	90.79±0.43	21.31±1.68	54.77±1.40
HetGNN	91.76±0.43	92.33±0.41	48.25±0.67	51.16±0.65	85.91±0.25	86.05±0.25	-	-
MAGNN	93.28±0.51	93.76±0.45	56.49±3.20	64.67±1.67	90.88±0.64	90.77±0.65	-	-
HGT	93.01±0.23	93.49±0.25	63.00±1.19	67.20±0.57	91.12±0.76	91.00±0.76	29.28±2.52	60.51±1.16
HGB	94.01±0.24	94.46±0.22	63.53±1.36	67.36±0.57	93.42±0.44	93.35±0.45	47.72±1.48	66.29±0.45
SeHGNN	95.06±0.17	95.42±0.17	67.11±0.25	69.17±0.43	94.05±0.35	93.98±0.36	51.87±0.86	65.08±0.66
HetTree	**95.34±0.17**	**95.64±0.15**	**68.43±0.31**	**70.92±0.29**	**94.26±0.20**	**94.19±0.20**	**52.35±0.96**	**66.39±0.40**

Table 1: Experimental Results of HetTree and baselines over four graphs in the HGB benchmark. ”-” means that the model runs out of memory on the corresponding graph.

Graph	Nodes	Edges	Node Types	Relation Types	Classes
DBLP	26.1K	239.5K	4	6	4
IMDB	21.4K	86.6K	4	6	5
ACM	10.9K	547.8K	4	8	3
Freebase	180.0K	1.0M	8	36	7
Mag	1.9M	21.1M	4	5	349
Email	7.8M	34.9M	5	6	2

Table 2: Statistics of datasets.

Experimental Setup. All of the experiments were conducted on a machine with dual 12-core Intel Xeon Gold 6226 CPU, 384 GB of RAM, and one NVIDIA Tesla A100 80GB GPU. The server runs 64-bit Red Hat Enterprise Linux 7.6 with CUDA library v11.8, PyTorch v1.12.0, and DGL v0.9.

Datasets. We evaluate HetTree on four graphs from the HGB (Lv et al. 2021) benchmark: DBLP, IMDB, ACM, and Freebase, a citation graph *Ogbn-Mag* from the OGB benchmark (Hu et al. 2020a) and a real-world email dataset collected from a commercial email platform. We summarize the six graphs in Table 2.

Baselines. For the four graphs from the HGB benchmark, we compare the HetTree results to the results reported in the HGB paper (Lv et al. 2021) as well as a state-of-the-art work SeHGNN (Yang et al. 2022). For *Ogbn-Mag*, we compare the HetTree with top-performing methods from either the baseline paper or the leaderboard of OGB (Hu et al. 2020a). We use the unified metrics chosen by the benchmarks for a fair comparison, i.e., the HGB benchmark uses F1 scores while the OGB benchmark uses accuracy as metrics. For the email dataset, we compare HetTree with the best-performing baseline SeHGNN. All experimental results reported are averaged over five random seeds.

Ethics and Broader Impacts. This work was reviewed and approved by independent experts in Ethics, Privacy, and Security. For the email dataset, all users' identities were anonymized twice, and the map from the second anonymized user IDs to the first anonymized user IDs was deleted. Furthermore, the data was handled according to GDPR regulations.

5.1 Experiments on Open Benchmarks

To answer **Q1**, we compare the performance of the proposed HetTree model to state-of-the-art models on five heterogeneous graphs from two open benchmarks - HGB (Lv et al. 2021) and OGB (Hu et al. 2020a).

Performance on HGB Benchmark. Table 1 shows results that compare HetTree with the best-performing baselines on four datasets from the HGB benchmark. HetTree outperforms the baselines on all graphs in terms of both Macro-F1 and Micro-F1 scores. For datasets in the HGB benchmark, which share similar medium-scale sizes and have uniformly preprocessed input node features, we observe that HetTree derives greater benefits from its semantic tree aggregation mechanism on more complex tasks involving a larger number of classes. As shown in Table 1, HetTree has more performance gain on IMDB and Freebase with 5 and 7 classes, respectively, compared with DBLP and ACM with 3 and 4 classes, respectively. This can be attributed to HetTree's semantic tree aggregation that learns more information, i.e., the hierarchy among metapaths, which is ignored by the other baselines.

Performance on Ogbn-Mag Dataset. We also evaluate HetTree on a large-scale citation graph, Ogbn-Mag (Hu et al. 2020a), with millions of nodes in Table 3. We report results *without* self-enhanced techniques like multi-stage training (Li, Han, and Wu 2018; Sun, Lin, and Zhu 2020; Yu et al. 2022), which are orthogonal to HetTree and can be incorporated for additional benefits. The results show that HetTree maintains its benefits on large graphs and outperforms all baselines.

Methods	Validation Accuracy	Test Accuracy
RGCN	48.35 ± 0.36	47.37 ± 0.48
HGT	51.24 ± 0.46	49.82 ± 0.13
NARS	53.72 ± 0.09	52.40 ± 0.16
LEGNN	54.43 ± 0.09	52.76 ± 0.14
GAMLP	55.48 ± 0.08	53.96 ± 0.18
SeHGNN	56.56 ± 0.07	54.78 ± 0.17
HetTree	**57.31 ± 0.15**	**55.54 ± 0.17**

Table 3: Detection accuracy of the HetTree model and other baselines for the Ogbn-Mag dataset.

5.2 Experiments on Commercial Email Graph

Besides the open benchmark datasets, we also collect a large email dataset from a commercial email platform to answer **Q2**, for lack of public alternatives. In this experiment, a subsample of real-world email data is used, which contains five types of entities - senders, recipients, domains, IP addresses, and messages. The task is to predict if the sender is legitimate or compromised, given its domain, messages, recipients of the message, recipients' domains, and message IP addresses. Compromised accounts may send various types of malicious emails, such as phishing emails, malware attachments, and spam. The email dataset has highly imbalanced classes where legitimate email accounts are much more than compromised accounts, as in the real-world scenario.

Since the email dataset contains binary labels, we can construct a Receiver Operating Characteristic (ROC) curve for the models. The ROC curve for the detection of compromised emails is presented in Figure 4 for the email dataset, and the accuracies are shown in Table 4. These results demonstrate that while both the best-performing baseline SeHGNN (Yang et al. 2022) and HetTree can achieve high accuracy in classification, HetTree can distinguish better between positive and negative classes.

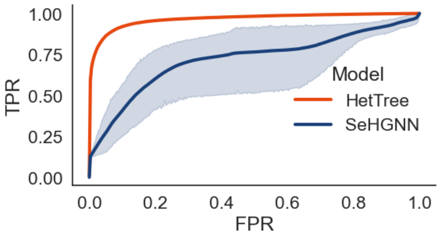

Figure 4: ROC Curves for the email dataset. The error bars for HetTree are tiny.

Methods	Val Accuracy	Test Accuracy
SeHGNN	97.22 ±0.02	97.26 ± 0.02
HetTree	**98.48±0.01**	**98.48 ± 0.02**

Table 4: Detection accuracy for the email dataset.

5.3 Ablation Study

We next evaluate whether adding the subtree attention component and using labels from the training set really help or not to answer **Q3**. The test F1 scores or accuracy of HetTree are evaluated on IMDB, ACM, and Ogbn-Mag compared with its three variants: ”*weighted-sum*”, ”*parent-att*” and ”*no-label*”. Variant *weighted-sum* removes the proposed subtree attention but uses a weighted child-sum like TreeLSTM (Tai, Socher, and Manning 2015). Variant *parent-att* removes the proposed subtree attention but computes weights of children nodes using attention on the parent node, which is the tree-encoding method used in SHGNN (Xu et al. 2021). Variant *no-label* does not use labels as extra inputs. We note that substituting the entire semantic tree aggregation with conventional metapath aggregation based on attention mechanism is what the baseline SeHGNN (Yang et al. 2022) does. Since we compare HetTree with SeHGNN in all benchmarks, we do not include it in the ablation study.

The results in Table 5 show that each component is effective for HetTree. We notice that datasets exhibit distinct sensitivities to individual components. The subtree attention results in more performance gain on IMDB, which could be attributed to the sparsity of the graph. It demonstrates that subtree attention can capture the metapath hierarchy, compared to other tree encoding mechanisms. Ogbn-Mag is more sensitive to label utilization, which could be attributed to the large number of classes of labels that provide richer information through propagation.

	IMDB	**ACM**	**Ogbn-Mag**
	Micro-F1	Micro-F1	Accuracy
HetTree	**70.92±0.29**	**94.19±0.20**	**55.54±0.17**
weighted-sum	69.70±0.24	93.54±0.23	55.38±0.21
parent-att	69.69±0.41	93.83±0.18	54.87±0.26
no-label	70.11±0.25	93.41±0.23	52.93±0.12

Table 5: Effectiveness of each component of HetTree.

5.4 Computation Cost Comparison

We next investigate the computational cost of HetTree in terms of epoch time and memory footprint to answer **Q4**. We select three performant models - HAN (Wang et al. 2019), HGB (Lv et al. 2021), and SeHGNN (Yang et al. 2022) - to compare with HetTree on four graphs in the HGB benchmark. For fair comparison, we use a 2-layer structure for HAN and HGN, and 2-hop feature propagation for SeHGNN and HetTree. The result in Figure 5 shows that HetTree incurs the lowest computational cost in terms of both running time and memory usage across three datasets from the HGB benchmark.

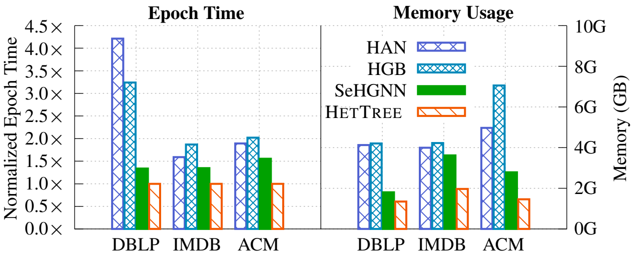

Figure 5: Epoch time and memory usage on HGB datasets.

6 Conclusion

In this paper, we present a novel HGNN, HetTree, based on the observation that existing HGNNs ignore a tree hierarchy among metapaths, which is naturally constituted by different node types and relation types. HetTree builds a semantic tree structure to capture the hierarchy among metapaths and proposes a novel subtree attention mechanism to encode the semantic tree. Compared with existing tree-encoding techniques that weight the contribution of children nodes based on similarity to the parent node, subtree attention in HetTree can model the broader local structure of parent nodes and children nodes. The evaluation shows that HetTree can outperform state-of-the-art baselines on open benchmarks and efficiently scale to large real-world graphs.

References

Chen, J.; Ma, T.; and Xiao, C. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *International Conference on Learning Representations*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29, 3844–3852. Curran Associates, Inc.

Frasca, F.; Rossi, E.; Eynard, D.; Chamberlain, B.; Bronstein, M.; and Monti, F. 2020. SIGN: Scalable Inception Graph Neural Networks.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*, WWW '20, 2331–2341. New York, NY, USA: Association for Computing Machinery. ISBN 9781450370233.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30, 1024–1034. Curran Associates, Inc.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin*, arXiv:1709.05584.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020a. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020b. Heterogeneous Graph Transformer. In *Proceedings of The Web Conference 2020*, WWW '20, 2704–2710. New York, NY, USA: Association for Computing Machinery. ISBN 9781450370233.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; and Tang, J. 2021. Are we really making much progress? Revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1150–1160.

Pal, A.; Eksombatchai, C.; Zhou, Y.; Zhao, B.; Rosenberg, C.; and Leskovec, J. 2020. PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, 2311–2320. New York, NY, USA: Association for Computing Machinery. ISBN 9781450379984.

Park, N.; Kan, A.; Dong, X. L.; Zhao, T.; and Faloutsos, C. 2019. Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 596–606. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362016.

Qiao, Z.; Wang, P.; Fu, Y.; Du, Y.; Wang, P.; and Zhou, Y. 2020. Tree Structure-Aware Graph Representation Learning via Integrated Hierarchical Aggregation and Relational Metric Learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, 432–441.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Berg, R. v. d.; Titov, I.; and Welling, M. 2017. Modeling Relational Data with Graph Convolutional Networks.

Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; and Sun, Y. 2020. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification.

Sun, C.; Gu, H.; and Hu, J. 2021. Scalable and Adaptive Graph Neural Networks with Self-Label-Enhanced training.

Sun, K.; Lin, Z.; and Zhu, Z. 2020. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 5892–5899.

Sun, R.; Dai, H.; and Yu, A. W. 2022. Does GNN Pretraining Help Molecular Representation? In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Trouillon, T.; Dance, C. R.; Welbl, J.; Riedel, S.; Éric Gaussier; and Bouchard, G. 2017. Knowledge Graph Completion via Complex Tensor Factorization.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous Graph Attention Network. In *The World Wide Web Conference*, WWW '19, 2022–2032. New York, NY, USA: Association for Computing Machinery. ISBN 9781450366748.

Wang, Y.; Jin, J.; Zhang, W.; Yu, Y.; Zhang, Z.; and Wipf, D. 2021. Bag of Tricks for Node Classification with Graph Neural Networks.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, 6861–6871. PMLR.

Wu, N.; and Wang, C. 2022. Heterogeneous Graph Tree Networks. arXiv:2209.00610.

Xu, W.; Xia, Y.; Liu, W.; Bian, J.; Yin, J.; and Liu, T.-Y. 2021. SHGNN: Structure-Aware Heterogeneous Graph Neural Network. arXiv:2112.06244.

Yang, X.; Yan, M.; Pan, S.; Ye, X.; and Fan, D. 2022. Simple and Efficient Heterogeneous Graph Neural Network.

Yu, L.; Shen, J.; Li, J.; and Lerer, A. 2020. Scalable Graph Neural Networks for Heterogeneous Graphs. *CoRR*, abs/2011.09679.

Yu, L.; Sun, L.; Du, B.; Zhu, T.; and Lv, W. 2022. Label-Enhanced Graph Neural Network for Semi-supervised Node Classification. arXiv:2205.15653.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous Graph Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &; Data Mining*, KDD '19, 793–803. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362016.

Zhang, D. C.; Ying, R.; and Lauw, H. W. 2023. Hyperbolic Graph Topic Modeling Network with Continuously Updated Topic Tree. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, 3206–3216. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701030.

Zhang, W.; Yin, Z.; Sheng, Z.; Li, Y.; Ouyang, W.; Li, X.; Tao, Y.; Yang, Z.; and Cui, B. 2022. Graph Attention Multi-Layer Perceptron. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, 4560–4570. New York, NY, USA: Association for Computing Machinery. ISBN 9781450393850.

Zhu, H.; and Koniusz, P. 2020. Simple spectral graph convolution. In *International Conference on Learning Representations*.

ZhuЃ, X.; and GhahramaniЃн, Z. 2002. Learning from labeled and unlabeled data with label propagation.

Zou, D.; Hu, Z.; Wang, Y.; Jiang, S.; Sun, Y.; and Gu, Q. 2019a. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32, 11249–11259. Curran Associates, Inc.

Zou, D.; Hu, Z.; Wang, Y.; Jiang, S.; Sun, Y.; and Gu, Q. 2019b. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems*, 32.