Dynamic Heterogeneous Graph Attention Neural Architecture Search

Zeyang Zhang[1*], **Ziwei Zhang**[1], **Xin Wang**[1†], **Yijian Qin**[1], **Zhou Qin**[2], **Wenwu Zhu**[1†]

[1]Tsinghua University
[2]Alibaba Group
{zy-zhang20,qinyj19}@mails.tsinghua.edu.cn, qinzhou.qinzhou@alibaba-inc.com,
{zwzhang,xin_wang,wwzhu}@tsinghua.edu.cn

Abstract

Dynamic heterogeneous graph neural networks (DHGNNs) have been shown to be effective in handling the ubiquitous dynamic heterogeneous graphs. However, the existing DHGNNs are hand-designed, requiring extensive human efforts and failing to adapt to diverse dynamic heterogeneous graph scenarios. In this paper, we propose to automate the design of DHGNN, which faces two major challenges: 1) how to design the search space to jointly consider the spatial-temporal dependencies and heterogeneous interactions in graphs; 2) how to design an efficient search algorithm in the potentially large and complex search space. To tackle these challenges, we propose a novel Dynamic Heterogeneous Graph Attention Search (**DHGAS**) method. Our proposed method can automatically discover the optimal DHGNN architecture and adapt to various dynamic heterogeneous graph scenarios without human guidance. In particular, we first propose a unified dynamic heterogeneous graph attention (DHGA) framework, which enables each node to jointly attend its heterogeneous and dynamic neighbors. Based on the framework, we design a localization space to determine where the attention should be applied and a parameterization space to determine how the attention should be parameterized. Lastly, we design a multi-stage differentiable search algorithm to efficiently explore the search space. Extensive experiments on real-world dynamic heterogeneous graph datasets demonstrate that our proposed method significantly outperforms state-of-the-art baselines for tasks including link prediction, node classification and node regression. To the best of our knowledge, **DHGAS** is the first dynamic heterogeneous graph neural architecture search method.

1 Introduction

Dynamic heterogeneous graphs are ubiquitous in real-world applications, including social networks, e-commerce networks, academic citation networks, etc. Compared to static homogeneous graphs, dynamic heterogeneous graphs contain richer heterogeneous information represented as node and edge types, and dynamic information like evolving graph structures over time. The modeling of heterogeneity and temporal evolutionary patterns is critical for applications of dynamic heterogeneous graphs including the prediction of future links, node labels and properties.

Dynamic heterogeneous graph neural networks (DHGNNs) (Hu et al. 2020; Fan et al. 2022; Xue et al. 2020; Li et al. 2020) have recently achieved remarkable progress in mining graph dynamic and heterogeneous information (Huang et al. 2021; Fan et al. 2021; Luo et al. 2020). Despite their success, the existing DHGNNs are all manually designed and therefore suffer from the following problems: (1) The designs of DHGNN architectures require extensive human endeavors and expert knowledge. (2) Since the hand-designed models have a fixed architecture, they are unable to adapt to diverse dynamic heterogeneous graph scenarios. (3) The existing DHGNN architectures consider the heterogeneous and dynamic information rather separately and fail to optimally model the joint and complex interaction of heterogeneous and dynamic information.

In this paper, we propose to automate the design of DHGNN architectures on dynamic heterogeneous graphs using neural architecture search (NAS). NAS has attracted considerable attention in automated machine learning and has shown success in domains including computer vision (Wistuba, Rawat, and Pedapati 2019; Elsken, Metzen, and Hutter 2019). However, tailoring a NAS method for dynamic heterogeneous graphs is non-trivial and faces the following two challenges:

- How to design the suitable search space to jointly consider the complex spatial-temporal dependencies and heterogeneous interactions in graphs?
- How to design a tailored efficient search algorithm in the potentially large and complex search space for dynamic heterogeneous graphs?

To tackle these challenges, we propose a novel Dynamic Heterogeneous Graph Attention Search (**DHGAS**) method. Our proposed method can automatically tailor an optimal DHGNN architecture and adapt to various dynamic heterogeneous graph scenarios. In particular, we first propose a unified dynamic heterogeneous graph attention (DHGA) framework. We enable the model to simultaneously consider heterogeneous neighbors across different time stamps by extending the classic neighborhood to dynamic heterogeneous neighborhood and applying attention to the neighbor-

[*]This work was done during author's internship at Alibaba Group
[†]Corresponding authors

hood with node and edge type-aware parameterizations. Second, we propose a localization and parameterization search space based on the DHGA framework. Localization space determines what relation types and time stamps should we apply attentions to, which can customize the connections in dynamic heterogeneous neighborhood. Parameterization space further determines the functions for calculating attentions and what types of nodes and edges and time stamps should share the same parameterization, which can customize the mapping functions in dynamic heterogeneous neighborhood. We show that our proposed search space is general and flexible, and can cover many classic DHGNN architectures as special cases. Besides, it also permits the search algorithm to make a trade-off between performance and computational resources. Lastly, we propose a multi-stage differentiable search algorithm to efficiently explore the search space. By relaxing the discrete choices to continuous ones in the localization and parameterization space, our proposed method can jointly optimize the architecture choices and the supernet weights in a differentiable manner, providing fast and accurate performance estimation of architecture candidates. To stabilize the training process, we further propose to train the supernet in a multi-stage manner and search the spaces sequentially based on the stage.

Extensive experiments on 5 real-world dynamic heterogeneous graph datasets demonstrate that our proposed method significantly outperforms manually designed and automated state-of-the-art baselines for tasks including link prediction, node classification, and node regression. Detailed ablation studies further verify the effectiveness of our proposed search space and search strategy. The codes are publicly available[1].

In summary, the contributions of our work are as follows:

- We propose Dynamic Heterogeneous Graph Attention Search (**DHGAS**) method for dynamic heterogeneous graphs. To the best of our knowledge, **DHGAS** is the first dynamic heterogeneous graph neural architecture search method.
- We design a localization space and a parameterization space for dynamic heterogeneous graphs based on our unified dynamic heterogeneous graph attention framework. We show that our proposed space covers representative manually designed architectures as special cases.
- We propose a multi-stage differentiable search algorithm for dynamic heterogeneous graphs which can explore our proposed search space effectively and efficiently.
- Extensive experiments on real-world datasets demonstrate the superiority of our method over state-of-the-art manually designed and automated baselines.

2 Notations and Preliminaries

2.1 Dynamic Heterogeneous Graphs

Consider a graph $\mathcal{G}$ with the node set $\mathcal{V}$ and the edge set $\mathcal{E}$. Nodes are associated with a type mapping function $\phi_n : \mathcal{V} \to \mathcal{C}_n$ and edges are associated with a type mapping function $\phi_e : \mathcal{E} \to \mathcal{C}_e$, where $\mathcal{C}_n$ and $\mathcal{C}_e$ denote the node type set and the edge type set, respectively. We give a formal definition for dynamic heterogeneous graphs as follows:

Definition 1 *A dynamic heterogeneous graph is defined as* $\mathcal{G} = (\{\mathcal{G}^t\}_{t=1}^T, \phi_n, \phi_e)$*, where* T *is the number of time stamps,* $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ *is the graph slice at time stamp* t*,* $\mathcal{V} = \bigcup_{t=1}^T \mathcal{V}^t$*,* $\mathcal{E} = \bigcup_{t=1}^T \mathcal{E}^t$*, and* $|\mathcal{C}_n| + |\mathcal{C}_e| \geq 2$*.*

Dynamic heterogeneous graphs are rather general data formats to represent relational data in real-world applications. For example, static graphs and homogeneous graphs can be considered special cases of dynamic heterogeneous graphs by setting $T = 1$ and $|\mathcal{C}_n| + |\mathcal{C}_e| = 2$, respectively.

2.2 Dynamic Heterogeneous Graph Neural Networks

Generally, GNNs follow a message-passing mechanism (Gilmer et al. 2017; Hamilton, Ying, and Leskovec 2017) that each node aggregates information from its neighbors. Specifically, let $\mathbf{h}_u$ be the representation of node u. Message-passing GNNs update the node representation by[2]

$$\mathbf{h}_u \leftarrow \text{Update}\left(\mathbf{h}_u, \text{Agg}(\{\text{Msg}(\mathbf{h}_v) : v \in \mathcal{N}(u)\})\right), \quad (1)$$

where $\mathcal{N}(u) = \{v : (u, v) \in \mathcal{E}\}$ denotes the neighborhood of the node u, $\text{Msg}(\cdot)$ extracts information from the neighbor node $v \in N(u)$, $\text{Agg}(\cdot)$ aggregates the neighborhood information, and $\text{Update}(\cdot)$ updates the node representation. Heterogeneous GNNs further consider the heterogeneity of graphs by differentiating node and edge types and assigning different parameters for $\text{Msg}(\cdot)$, $\text{Agg}(\cdot)$ and $\text{Update}(\cdot)$ functions. The message-passing function is:

$$\mathbf{h}_u \leftarrow \text{Update}_{\phi(u)}(\mathbf{h}_u, \text{Agg}_r(\{\text{Msg}_r(\mathbf{h}_v) : v \in \mathcal{N}_r(u), r \in \mathcal{C}_e\}), \quad (2)$$

where $\mathcal{N}_r(u) = \{v : (u, v) \in \mathcal{E} \wedge \phi_e(u, v) = r\}$ is the neighborhood of node u with the relation type r.

DHGNNs further explore the temporal information in dynamic graphs based on Eq. (2). For example, relative time encoding (Hu et al. 2020) encodes time information into edges, i.e., $\mathcal{E}' = \text{Encode}(\{\mathcal{E}^t\}_{t=1}^T)$ followed by heterogeneous message-passings. Another category of DHGNNs adopt sequence-based models to aggregate information from different time slices, i.e., $\mathbf{H} = \text{Seq}(\{\mathbf{H}^t\}_{t=1}^T)$, where $\mathbf{H}^t$ denotes the node representations at time stamp t and $\mathbf{H}$ is the final node representation. Clearly, these existing approaches handle heterogeneous and dynamic information rather separately in a fixed form. In comparison, our proposed method can jointly aggregate spatial-temporal heterogeneous information and automatically adapt to diverse dynamic heterogeneous graph scenarios.

2.3 Neural Architecture Search

Neural architecture search (NAS) aims at automating the design of neural architectures, which can be formulated as a bi-level optimization problem (Elsken, Metzen, and Hutter

[1] https://github.com/wondergo2017/DHGAS

[2] To simplify notations, we omit the layer superscript and use arrows to show the message-passing functions in each layer. We also omit edge features, which can be easily incorporated.

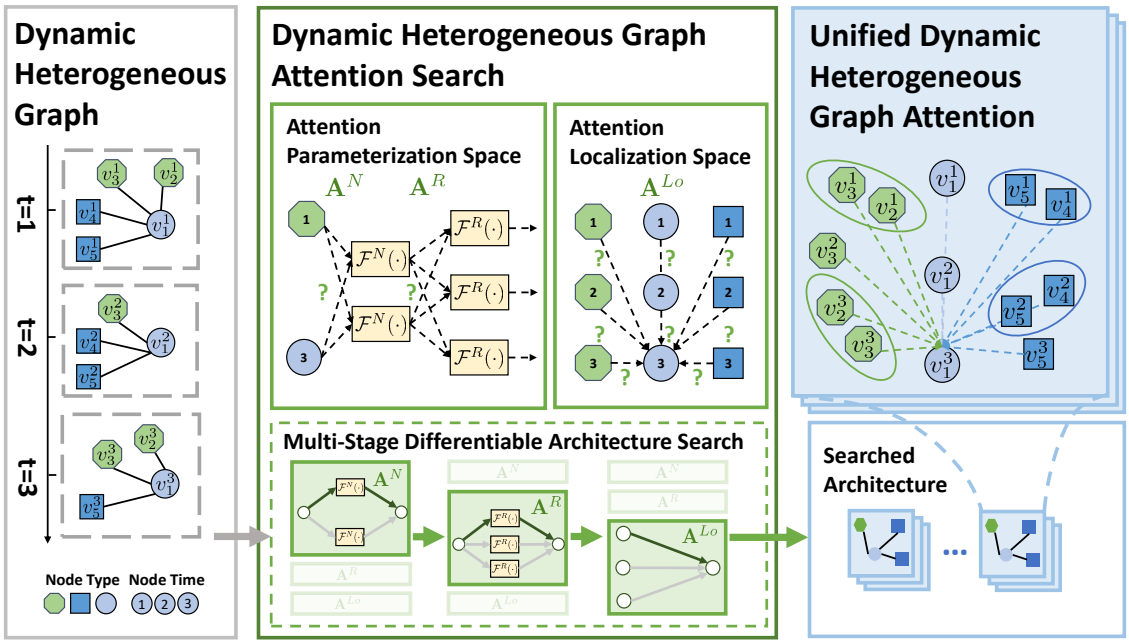


Figure 1: The framework of our proposed dynamic heterogeneous graph attention search (DHGAS) model. For a given dynamic heterogeneous graph with multiple node and edge types and time slices, DHGAS can tailor an optimal architecture based on the unified Dynamic Heterogeneous Graph Attention (DHGA) framework. In particular, DHGAS conducts a multi-stage differentiable architecture search on the attention parameterization space and the attention localization space with several carefully designed constraints. In the localization space, we search for what types of edges and which time stamps the attention should be calculated. In the parameterization space, we search for how the attention functions should be parameterized.

2019; Wistuba, Rawat, and Pedapati 2019):

$$\begin{aligned} a^* &= \arg\min_{a\in\mathcal{A}} \mathcal{L}_{\text{val}}(a, \mathbf{w}^*(a)), \\ \text{s.t. } \mathbf{w}^*(a) &= \arg\min_{\mathbf{w}\in\mathcal{W}(a)} \mathcal{L}_{\text{train}}(a, \mathbf{w}), \end{aligned} \tag{3}$$

where $\mathcal{A}$ is the architecture search space, $\mathcal{W}(a)$ is the parameter space for a given architecture a, and $\mathbf{w}^*(a)$ are the optimal weights for the architecture a. In this paper, we tailor a search space, including attention localization and parameterization, and a multi-stage differentiable search algorithm for dynamic heterogeneous graphs.

3 The Proposed Method

3.1 Dynamic Heterogeneous Graph Attention

The key idea of our proposed dynamic heterogeneous graph attention (DHGA) framework is to unify the spatial-temporal aggregation and jointly integrate dynamic and heterogeneous information from neighborhoods by an attention-based message-passing mechanism. We first extend the neighborhood definition.

Definition 2 ***Dynamic Heterogeneous Neighborhood****: for the neighborhood of each node u, we use subscripts to denote the relation type and superscripts to denote the time stamp, i.e., $\mathcal{N}_r^t(u) = \{v : (u, v) \in \mathcal{E}^t, \phi_e(u, v) = r\}$. With a slight abuse of notations, we use $\mathcal{N}(u)$ to denote all types of neighbors at all time stamps in dynamic heterogeneous graphs, i.e., $\mathcal{N}(u) = \bigcup_{r,t} \mathcal{N}_r^t(u)$.*

Next, we introduce our tailored message-passing framework to capture the dynamic heterogeneous neighborhood information to update node representations. Following the attention mechanism (Vaswani et al. 2017), for a node u in the time stamp t and its neighbor $v \in \mathcal{N}_r^{t'}(u)$, we calculate the Query-Key-Value vector using a set of mapping functions:

$$\mathbf{q}_u^t = \mathcal{F}^N_{q,\phi_n(u),t}(\mathbf{h}_u^t), \tag{4}$$

$$\mathbf{k}_v^{t'} = \mathcal{F}^N_{k,\phi_n(v),t'}(\mathbf{h}_v^{t'}), \tag{5}$$

$$\mathbf{v}_v^{t'} = \mathcal{F}^N_{v,\phi_n(v),t'}(\mathbf{h}_v^{t'}), \tag{6}$$

where $\mathbf{h}_u^t$ denotes the representation of node u at the time stamp t, $\mathbf{q}$, $\mathbf{k}$, $\mathbf{v}$ represents the query, key and value vector, respectively, and $\mathcal{F}_q^N(\cdot)$, $\mathcal{F}_k^N(\cdot)$, $\mathcal{F}_v^N(\cdot)$ denote the corresponding node mapping functions. In this paper, we adopt small fully-connected neural networks to instantiate all $\mathcal{F}^N(\cdot)$. Notice that the subscripts in the functions indicate that we adopt different functions, i.e., functions with different parameters, based on the node type and time stamps. Then, we calculate the attention score between u and v using a mapping function $\mathcal{F}^R(\cdot)$ on the query and key vector:

$$\alpha_{u,v} = \mathcal{F}^R_{\phi_e(u,v),\Delta t}(\mathbf{q}_u^t, \mathbf{k}_v^{t'}), \tag{7}$$

where $\Delta t = t - t'$, i.e., the relation mapping depends on the difference in time stamps instead of the absolute values. Inspired by HGT (Hu et al. 2020) and RGCN (Schlichtkrull

et al. 2018), we adopt a relation-aware projection to instantiate $\mathcal{F}^R(\cdot)$, i.e.,

$$\mathcal{F}^R_{\phi_e(u,v),\Delta t}(\mathbf{q},\mathbf{k}) = \frac{\mathbf{q}\mathbf{W}_{\phi_e(u,v),\Delta t}\mathbf{k}^\top}{\sqrt{d}}, \tag{8}$$

where $\mathbf{W}_{\phi_e(u,v),\Delta t} \in \mathbb{R}^{d\times d}$ denote the learnable parameters for a specific edge type and time stamps and d is the dimensionality. Finally, we normalize the attention scores using the softmax function and aggregate all neighborhoods, i.e.,

$$\begin{aligned} \mathbf{h}^t_u &\leftarrow \text{Update}(\mathbf{h}^t_u, \textstyle\sum_{v\in\mathcal{N}(u)} \hat{\alpha}_{u,v}\mathbf{v}^{t'}_v)), \\ \hat{\alpha}_{u,v} &= \frac{\exp(\alpha_{u,v})}{\sum_{v'\in\mathcal{N}(u)}\exp(\alpha_{u,v'})}. \end{aligned} \tag{9}$$

Note that we can easily extend our method into multi-head attention (Vaswani et al. 2017) to stabilize the training process and improve the model's expressiveness. We omit the detailed formulations for brevity.

In summary, we can jointly aggregate information across different types of neighborhoods in all time stamps using one layer of DHGA. Compared to previous works which aggregate spatial and temporal information separately, our proposed method can capture more flexible heterogeneous spatial-temporal graph structures.

Besides, we explicitly consider different types of relations, i.e., node types in the node mapping functions $\mathcal{F}^N(\cdot)$ and edges types in the relation mapping functions $\mathcal{F}^R(\cdot)$, and different time stamps by setting different parameters. Thus, our proposed DHGA can learn to adaptively assign different attention scores to handle different dynamic heterogeneous graph applications.

Though DHGA is flexible and expressive in modeling dynamic heterogeneous graphs, naively searching architectures based on DHGA can incur high complexity when solving the bi-level optimization in Eq. (3). In the following, we introduce our tailored search space and search algorithm to reduce the complexity while maintaining the expressiveness of the model.

3.2 Attention Localization and Parameterization Search Space

The full version of our proposed DHGA introduced in Section 3.1 calculates attention across all types of neighbors and all time stamps. While being the most expressive architecture, the computation cost is also extensive. To sparsify the attention and enable more lightweight and efficient architectures, we propose the localization space and the parameterization space based on the full DHGA.

Localization Space: Locate where to apply attention. First, we introduce the localization space which determines what types of edges and time stamps should the attention be calculated. In specific, we denote the localization space as $\mathcal{A}^{Lo} = \{0,1\}^{T\times T\times|\mathcal{C}_e|}$. For $\mathbf{A}^{Lo} \in \mathcal{A}^{Lo}$, $\mathbf{A}^{Lo}_{t,t',r}$ denotes whether calculating the representation of node u at the time stamp t should attend to its neighbors $\mathcal{N}^{t'}_r(u)$ in the message-passing. Therefore, $\mathbf{A}^{Lo}$ completely determines where the attention functions are applied.

Notice that our proposed localization space is general and flexible since it can cover many existing architectures as special cases. For example, the full DHGA equivalents to every value of $\mathbf{A}^{Lo}$ equals to one. It can also cover other architectures including GAT (Veličković et al. 2018), Temporal self-attention (Fan et al. 2022), Masked temporal self-attention (Xue et al. 2020; Sankar et al. 2020), identity mapping to support skip connections, etc.

Besides being general and flexible, we also greatly reduce the complexity using the localization space. Specifically, it is easy to see that the full DHGA has a time complexity

$$O(\textstyle\sum_{t=1}^{T}\sum_{t'=1}^{T}\sum_{r\in\mathcal{C}_e}|\mathcal{E}^{t\prime}_r|) = O(T^2|\mathcal{C}_e|\max_{1\le t\le T, r\in\mathcal{C}_e}|\mathcal{E}^t_r|) \tag{10}$$

In comparison, the time complexity of using $\mathbf{A}^{Lo}$ is:

$$O(\textstyle\sum_{t=1}^{T}\sum_{t'=1}^{T}\sum_{r\in\mathcal{C}_e}\mathbf{A}^{Lo}_{t,t',r}|\mathcal{E}^{t'}_r|) = O(|\mathbf{A}^{Lo}|\max_{1\le t\le T, r\in\mathcal{C}_e}|\mathcal{E}^t_r|) \tag{11}$$

where $|\mathbf{A}^{Lo}|$ denotes the number of non-zero values in $\mathbf{A}^{Lo}$. By constraining the size of $|\mathbf{A}^{Lo}|$, we can reduce DHGA time complexity to be independent of the number of time stamps T and the number of relation types $|\mathcal{C}_e|$.

Parameterization Space: How to parameterize attention. To reduce the number of parameters, we propose an parameterization space to search for how the attention functions should be calculated. Specifically, we denote the parameterization space as $\mathcal{A}^{Pa} = \mathcal{A}^N \times \mathcal{A}^R$, where $\mathcal{A}^N = \{1,...,K_N\}^{T\times|\mathcal{C}_n|}$ is the parameterization matrix for the node mapping functions $\mathcal{F}^N(\cdot)$ and $\mathcal{A}^R = \{1,...,K_R\}^{2T\times|\mathcal{C}_e|}$ is the parameterization matrix for the relation mapping functions $\mathcal{F}^R(\cdot)$, and K_N and K_R are two hyper-parameters. In a nutshell, we store K_N mapping functions for $\mathcal{F}_N(\cdot)$ and K_R mapping functions for $\mathcal{F}_R(\cdot)$ as prototypes, and each attention function can choose from the corresponding prototypes. Concretely, let $\mathbf{A}^N \in \mathcal{A}^N$. $\mathbf{A}^N_{t,c} = k$ indicates that the node mapping function for node u with $\phi_n(u) = c$ and time stamp t, i.e., $\mathcal{F}^N_{q,c,t}(\cdot)$, $\mathcal{F}^N_{k,c,t}(\cdot)$, $\mathcal{F}^N_{v,c,t}(\cdot)$, should choose the k^{th} function in all prototypes for $\mathcal{F}^N(\cdot)$. Similarly, for $\mathbf{A}^R \in \mathcal{A}^R$, $\mathbf{A}^R_{\Delta t,c} = k$ denotes that the relation mapping function $\mathcal{F}^R_{c,\Delta t}$ for relation $\phi_e(u,v) = c$ should choose the k^{th} prototype for $\mathcal{F}^R(\cdot)$.

Using the parameterization space, our proposed method can flexibly determine which mapping functions, including both node mapping functions and relation mapping functions, should share parameters. Intuitively, some node types, relation types, or time stamps share similar patterns and therefore can enjoy parameter sharing without affecting the performance. Since these patterns may depend on specific dynamic heterogeneous graph datasets and tasks, we propose to search and learn these patterns adaptively instead of manually setting the parameter sharing rules.

Similar to the localization space, the parameterization space is general and covers diverse existing architectures. For example, when $K_N = T\times|\mathcal{C}_n|$ and $K_R = 2T\times|\mathcal{C}_e|$, we can search for a unique prototype vector for each function and recover the full DHGA. When $K_N = 1$ and $K_R = 1$, we recover the existing homogeneous attention-based GNNs.

Using the parameterization space, we can reduce the

number of learnable parameters. It is easy to see that the number of learnable parameters for the full DHGA is of $O(T(|\mathcal{C}_n|+|\mathcal{C}_e|))$. Using the parameterization space, we can reduce it to $O(K_N + K_R)$. When constraining K_N and K_R as constants, the number of learnable parameters is also a constant, i.e., unrelated to the number of edges $|\mathcal{E}|$, the number of time stamps T, or the number of node and edge types $|\mathcal{C}_n|$ and $|\mathcal{C}_e|$.

In short, the localization space and parameterization space balance the model complexity and model expressiveness by determining the edge types and time stamps in calculating the attentions and the parameterization of attentions.

3.3 Multi-Stage Differentiable Search

With our proposed localization space and parameterization space, we introduce our proposed search strategy. Denote the whole search space as $\mathcal{A} = \mathcal{A}^{Lo} \times \mathcal{A}^{Pa}$. It is easy to see that the space can contain up to $2^{T^2|\mathcal{C}_e|}K_N^{T|\mathcal{C}_n|}K_R^{2T|\mathcal{C}_e|}$ possible choices, which is considerably large and it is infeasible to enumerate all possible choices in practice. To reduce the complexity of searching, we first propose heuristic constraints on the search space to remove invalid or ineffective architectures, and then adopt the one-shot neural architecture search algorithm to speed up the search process.

Space Constraint. Inspired by Masked temporal self-attention (Xue et al. 2020; Sankar et al. 2020), we constrain the searched localization to respect the chronological order of graph slices, i.e., the representation of node u at time stamp t can only receive messages from neighborhood $\mathcal{N}_r^{t'}(u)$ with $t' \leq t$. This constraint has clear explanations, since in practice it is infeasible to predict the current situation using future information. Besides, we add another constraint as $|\mathbf{A}_t^{Lo}| \leq K_{Lo}, 1 \leq t \leq T$, where K_{Lo} is a hyperparameter. In this way, we constrain the sparsity of the attention connections in each time slice and reduce the complexity, as shown in paragraph 3.2. Assuming the continuity of mapping functions in the temporal domain, we further break time slices into consecutive patches, where functions within one patch share the same parameters. Lastly, we constrain the last layer of the architecture to only contain connections to the last time slice T so that we can utilize these representations for downstream tasks.

Supernet Construction. Following the recent advancements of NAS (Liu, Simonyan, and Yang 2019; Xie et al. 2018; Guo et al. 2020), we transform the bi-level optimization in Eq. (3) into an one-shot NAS problem using a supernet. Since every possible architecture $a \in \mathcal{A}$ is contained in the supernet, its performance can be quickly evaluated using the corresponding weights in the supernet. Specifically, in the supernet, the categorical choice of a particular operation is relaxed into a softmax overall all possible operations: $\bar{\mathcal{F}}(\mathbf{x}) = \sum_{i=1}^{|A|} \frac{\exp(\beta_i)}{\sum_{j=1}^{|A|}\exp(\beta_j)}\mathcal{F}_i(\mathbf{x})$, where $\mathbf{x}$ is the input, $\bar{\mathcal{F}}(\mathbf{x})$ is the output, $|A|$ denotes the number of possible operations, and β_i denotes the mixing weights for the i^{th} possible function $\mathcal{F}_i(\cdot)$. For the localization space, operations indicate whether the attention function is applied. For the parameterization space, operations represent different node/relation prototype mapping functions. Using the supernet, we can jointly optimize the mixing weights β and all parameters in the mapping functions in a differentiable manner:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta_{\mathbf{w}}\frac{\partial \mathcal{L}_{\text{train}}}{\partial \mathbf{w}}, \beta \leftarrow \beta - \eta_{\beta}\frac{\partial \mathcal{L}_{\text{val}}}{\partial \beta}, \tag{12}$$

where η_β and $\eta_{\mathbf{w}}$ are the learning rate for model weights and architecture weights, respectively.

Multi-stage Supernet Training. To stabilize the training of the supernet, we divide the training process into three stages: node parameterization, relation parameterization, and localization. In the node parameterization stage, we force the attention location as fully-connected, and force the relation mapping functions to share the same parameterization and focus on searching for the node mapping functions. In the relation parameterization stage, we select and fix the choice in the node parameterization space and focus on searching for the relation mapping functions. Similarly, in the localization stage, we focus on searching in the localization space while fixing the choices in the other two spaces. When the training is finished, we obtain an optimal architecture by discretizing the operation choices.

4 Experiments

In this section, we evaluate the proposed method through tasks including link prediction, node classification, and node regression. We compare static homogeneous GNNs **GCN** (Kipf and Welling 2017),**GAT** (Veličković et al. 2018) ; static heterogenous GNNs **RGCN** (Schlichtkrull et al. 2018),**HGT** (Hu et al. 2020); dynamic heterogeneous GNNs **DyHATR**(Xue et al. 2020),**HGT+** (Hu et al. 2020),**HTGNN** (Fan et al. 2022) as hand-designed baselines. We also compare to a state-of-the-art static homogeneous graph NAS method **GraphNAS** (Gao et al. 2020) and a heterogeneous graph NAS method **DiffMG** (Ding et al. 2021).

4.1 Main Results

Link Prediction. First, we conduct experiments for the link prediction task on two datasets: an academic citation dataset **Aminer** (Ji et al. 2021) and a recommendation dataset **Ecomm** (Xue et al. 2020). The results are shown in Table 1. We have the following findings. (1) **DHGAS** achieves the best result on both datasets with a large margin, i.e., improving the AUC by approximately 2.5% and 4% over the most competent baseline, respectively. The results demonstrate that **DHGAS** can effectively handle the link prediction task on dynamic heterogeneous graph datasets by tailoring the most suitable architecture. (2) DiffMG reports reasonably good results and outperforms most manually designed heterogeneous methods, demonstrating the importance and potentials of automatically designing neural architectures. However, there still exists a large performance gap between DiffMG and our proposed **DHGAS**, especially in the Ecomm dataset. We attribute this difference to that our proposed method can effectively jointly capture the temporal and heterogeneous information, as opposed to DiffMG which only models the heterogeneous information. (3) In

Task Metric	Link Prediction (AUC%) ↑		Node Classification (F1%) ↑		Node Regression (MAE) ↓
Dataset	**Aminer**	**Ecomm**	**Yelp**	**Drugs**	**COVID-19**
GCN	73.84 ± 0.06	77.94 ± 0.22	37.02 ± 0.00	56.43 ± 0.21	846 ± 101
GAT	80.84 ± 0.96	78.49 ± 0.31	35.54 ± 0.00	57.06 ± 0.00	821 ± 91
RGCN	82.75 ± 0.12	82.27 ± 0.51	37.75 ± 0.00	57.97 ± 0.14	833 ± 95
HGT	78.43 ± 1.81	81.09 ± 0.52	34.62 ± 0.00	57.65 ± 0.01	805 ± 88
DyHATR	74.24 ± 2.09	71.69 ± 0.90	34.49 ± 0.16	55.51 ± 0.09	643 ± 36
HGT+	85.60 ± 0.12	76.68 ± 0.85	38.33 ± 0.00	59.09 ± 0.00	-
HTGNN	78.08 ± 0.80	76.78 ± 6.37	36.33 ± 0.07	56.24 ± 0.34	555 ± 34
GraphNAS	81.61 ± 0.98	79.37 ± 0.21	37.73 ± 0.00	57.13 ± 0.52	820 ± 43
DiffMG	85.04 ± 0.30	81.69 ± 0.06	38.65 ± 0.00	58.45 ± 0.15	629 ± 63
DHGAS	$\mathbf{88.13 \pm 0.18}$	$\mathbf{86.56 \pm 0.58}$	$\mathbf{41.99 \pm 0.18}$	$\mathbf{62.35 \pm 0.03}$	$\mathbf{536 \pm 43}$

Table 1: The overall results for different methods for tasks including link prediction, node classification, and node regression. The evaluation metrics are in parentheses, and ↑ (↓) means that higher (lower) value indicate better results. The best results are in bold and the second-best results are underlined."-" indicates the method is not applicable.

general, modeling heterogeneous and temporal information are both critical to boosting the performance of manually designed baselines. For example, HGT+, which adopts the relative time encoding technique, reports the second-best result on Aminer. However, HGT+ fails to handle Ecomm and even performs worse than HGT. The results re-validate that different datasets may require different GNN architectures and manually designed methods may fail to adaptively handle such diverse application scenarios.

Node Classification. Next, we compare different methods for the node classification task adopting two datasets: a business review dataset **Yelp** (Ji et al. 2021) and an e-commerce risk management dataset **Drugs**[3]. From the results also shown in Table 1, we have the following observations: (1) Our proposed method **DHGAS** again reports the best results on both datasets by improving the Macro-F1 score by more than 3%. The results demonstrate that we can effectively handle the node classification task for dynamic heterogeneous graphs by automatically designing architectures using **DHGAS**. (2) The automated baseline DiffMG and manually designed dynamic heterogeneous method HGT+ reports the second-best result on Yelp and Drugs, showing the effectiveness of NAS and importance of capturing dynamic heterogeneous information. Nevertheless, failing to merging the best of two worlds, their performance gap compared to **DHGAS** is still considerable.

Node Regression. For the node regression task, we adopt an epidemic disease dataset **COVID-19** (Fan et al. 2022). We report the results in Table 1 and observe the following findings: (1) Similar to the other two tasks, **DHGAS** again achieves the best performance. The results demonstrate that **DHGAS** can adaptively handle diverse applications of heterogeneous dynamic graphs. (2) For this task, manually designed dynamic baselines (i.e., DyHATR and HTGNN) greatly outperforms static methods, showing that modeling temporal information is critical to predicting the COVID-19 cases, which is consistent with the literature. (3) Though not considering dynamic information, DiffMG again shows competitive performance, illustrating the great potential of NAS methods. **DHGAS** can fully utilize such potentials by our tailored search space and search algorithm for dynamic heterogeneous graphs.

[3]Collected from Alibaba.com

4.2 Ablation Studies and Additional Analyses

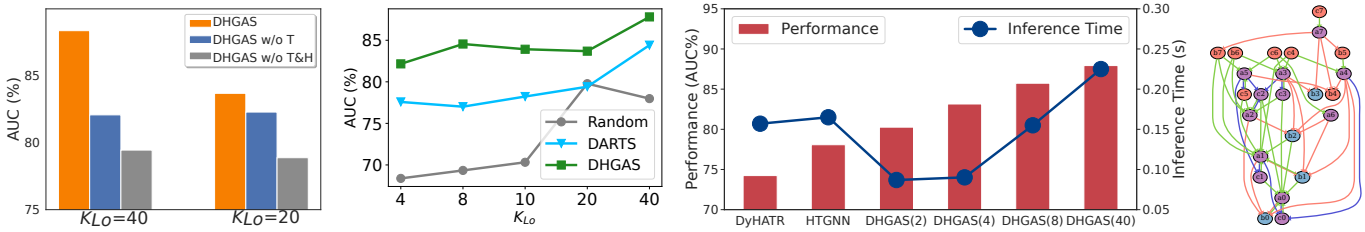
Search Space. To test the effectiveness of our proposed localization space and parameterization space, we compare the full version with two ablated versions: "**DHGAS** w/o temporal" and "**DHGAS** w/o temporal & heterogeneous". The former removes any attention localization in with different time slices and the latter further forces the method to use the same parameterization for all types of nodes and edges. For simplicity, we only report the results on Aminer when the space constraint hyper-parameters K_{Lo} are set as 20 and 40, while other datasets and settings show similar patterns.

Figure 2a shows that removing temporal connections in localization space and heterogeneous parameterization will reduce the performance of the searched model. The results verify the effectiveness of **DHGAS** in exploiting temporal and heterogeneous information in our tailored search space.

Search algorithm. We verify the design of our proposed multi-stage supernet training. We compare **DHGAS** with random search and DARTS (Liu, Simonyan, and Yang 2019) based on our proposed search space. We report the results on the Aminer dataset when the localization constraint hyper-parameters K_{Lo} is chosen from $\{4, 8, 10, 20, 40\}$ while other results indicate similar conclusions.

As shown in Figure 2b, our proposed search algorithm outperforms DARTS and random search for all localization constraints. In particular, as K_{Lo} grows larger, the performance of all methods increases, showing a clear trade-off between efficiency and effectiveness. When K_{Lo} is small, i.e., tight localization constraints, **DHGAS** can automatically search important attention locations and maintain impressive performance. In contrast, random search and DARTS fail in these cases.

The efficiency of the searched architectures. Figure 2c shows that as we gradually increase the computational bud-



(a) (b) (c) (d)

Figure 2: (a) The results of ablation study on the localization and parameterization space on the Aminer dataset (AUC%). (b) Comparison of the search algorithms on the Aminer dataset using DHGAS search space (AUC%). (c) Comparison of searched architectures under different computational budgets K_{Lo} in terms of inference time and performance on Aminer. DHGAS(k) means DHGAS with budget $K_{Lo} = k$. (d) Visualization of the searched architecture on Aminer with $K_{Lo} = 8$.

get K_{Lo}, **DHGAS** can obtain architectures with better performance. The results show that **DHGAS** can search architectures tailored to the datasets as well as balance the computational budgets and model performance.

Visualization of the searched architecture. Figure 2d visualizes the search architecture on Aminer with $K_{Lo} = 8$, where the letter and number denote the node type and time, respectively, and the colors denote the choices of node and relation mapping functions. It verfies that **DHGAS** can flexibly tailor localizations and mapping functions, demonstrating that our method can automate DHGNN designs and save human endeavors tackling graph heterogenity and dynamics.

5 Related Works

Dynamic Heterogeneous Graph Neural Networks. Graph-structured data are ubiquitous in the real-world (Wu et al. 2020; Zhou et al. 2020; Zhang, Cui, and Zhu 2020; Li et al. 2022a,b,c,d, 2021a,b; Zhang et al. 2022c). To generalize the success of GNNs in homogeneous graphs, considerable research attention have been devoted to heterogeneous GNNs (Yang et al. 2020; Wang et al. 2022; Schlichtkrull et al. 2018; Zhang et al. 2019; Wang et al. 2019; Fu et al. 2020; Hu et al. 2020). Some works attempt to consider dynamic information (Skarding, Gabrys, and Musial 2021; Zhu et al. 2022; Yang et al. 2021; Zhang et al. 2022b; Sankar et al. 2020; Wang et al. 2021; Xu et al. 2020; Rossi et al. 2020), and study dynamic heterogeneous graphs (Kazemi et al. 2020; Xue et al. 2022, 2020; Barros et al. 2021; Yuan et al. 2020; Hu et al. 2020; Fan et al. 2022). Despite the success of these existing approaches, they are all manually designed with a fixed architecture. Besides, the spatial and temporal information are processed relatively independently. In comparison, our proposed method can jointly attend dynamic and heterogeneous neighborhoods and automatically adapt to diverse dynamic heterogeneous graph tasks and datasets.

Graph Neural Architecture Search. To automate the design of GNNs, graph NAS has drawn increasing popularity in the last two years (Zhang, Wang, and Zhu 2021), including reinforcement learning based methods (Gao et al. 2020; Zhou et al. 2019; Qin et al. 2021a, 2022b; Zhou et al. 2022; Guan et al. 2021), evolutionary learning based methods (Nunes and Pappa 2020; Li and King 2020; Shi et al. 2022; Guan, Wang, and Zhu 2021; Guan et al. 2022; Zhang et al. 2022a), bayesian optimization based methods (Hou et al. 2021) and differentiable methods (Zhao et al. 2020; Huan, Quanming, and Weiwei 2021; Li et al. 2021c; Cai et al. 2021; Qin et al. 2021b, 2022a) have also been studied. However, all aforementioned works focus on static homogeneous graphs. More relevant to our work, DiffMG (Ding et al. 2021) and HGNAS (Gao et al. 2021) propose to search heterogeneous GNN architectures using meta-paths (Sun et al. 2011) to differentiate node and edge types. However, they cannot capture the temporal information in dynamic graphs. Additionally, AutoSTG (Pan et al. 2021) proposes to search GNN architectures for homogeneous spatial-temporal graphs, neglecting the heterogeneous interactions.

In summary, the existing graph NAS methods cannot fully capture the complex spatial-temporal information in real dynamic heterogeneous graphs. Our proposed **DHGAS** is the first tailored dynamic heterogeneous graph neural architecture search method, to the best of our knowledge.

6 Conclusion

In this paper, we propose a novel Dynamic Heterogeneous Graph Attention Search (**DHGAS**) method to automate the design of DHGNN. We propose a unified dynamic heterogeneous graph attention framework to jointly consider heterogeneous and dynamic neighbors of nodes. Based on the framework, we design a localization space to determine where the attention should be applied and a parameterization space to determine how the attention should be parameterized. We further design a multi-stage differentiable search algorithm to efficiently explore the search space. Extensive experiments on real-world dynamic heterogeneous graph datasets demonstrate the superiority of our method.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China No. 2020AAA0106300, National Natural Science Foundation of China (No. 62250008, 62222209, 62102222, 62206149), China National Postdoctoral Program for Innovative Talents No. BX20220185 and China Postdoctoral Science Foundation No. 2022M711813.

References

Barros, C. D.; Mendonça, M. R.; Vieira, A. B.; and Ziviani, A. 2021. A survey on embedding dynamic graphs. *ACM Computing Surveys*, 55(1): 1–37.

Cai, S.; Li, L.; Deng, J.; Zhang, B.; Zha, Z.-J.; Su, L.; and Huang, Q. 2021. Rethinking graph neural architecture search from message-passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6657–6666.

Ding, Y.; Yao, Q.; Zhao, H.; and Zhang, T. 2021. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 279–288.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.

Fan, Y.; Ju, M.; Hou, S.; Ye, Y.; Wan, W.; Wang, K.; Mei, Y.; and Xiong, Q. 2021. Heterogeneous Temporal Graph Transformer: An Intelligent System for Evolving Android Malware Detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Fan, Y.; Ju, M.; Zhang, C.; and Ye, Y. 2022. Heterogeneous Temporal Graph Neural Network. In *Proceedings of the 2022 SIAM International Conference on Data Mining*, 657–665.

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*, 2331–2341.

Gao, Y.; Yang, H.; Zhang, P.; Zhou, C.; and Hu, Y. 2020. Graph Neural Architecture Search. In *IJCAI*, volume 20, 1403–1409.

Gao, Y.; Zhang, P.; Li, Z.; Zhou, C.; Liu, Y.; and Hu, Y. 2021. Heterogeneous Graph Neural Architecture Search. In *2021 IEEE International Conference on Data Mining*, 1066–1071.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for Quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*.

Guan, C.; Wang, X.; Chen, H.; Zhang, Z.; and Zhu, W. 2022. Large-Scale Graph Neural Architecture Search. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 7968–7981. PMLR.

Guan, C.; Wang, X.; and Zhu, W. 2021. Autoattend: Automated attention representation search. In *International Conference on Machine Learning*.

Guan, C.; Zhang, Z.; Li, H.; Chang, H.; Zhang, Z.; Qin, Y.; Jiang, J.; Wang, X.; and Zhu, W. 2021. AutoGL: A Library for Automated Graph Learning. *CoRR*, abs/2104.04987.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, 544–560.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*.

Hou, Z.; Cen, Y.; Dong, Y.; Zhang, J.; and Tang, J. 2021. Automated Unsupervised Graph Representation Learning. *IEEE Transactions on Knowledge and Data Engineering*.

Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, 2704–2710.

Huan, Z.; Quanming, Y.; and Weiwei, T. 2021. Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering*, 552–563.

Huang, H.; Shi, R.; Zhou, W.; Wang, X.; Jin, H.; and Fu, X. 2021. Temporal Heterogeneous Information Network Embedding. In *The 30th International Joint Conference on Artificial Intelligence*.

Ji, Y.; Jia, T.; Fang, Y.; and Shi, C. 2021. Dynamic heterogeneous graph embedding via heterogeneous hawkes process. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 388–403.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2020. Representation Learning for Dynamic Graphs: A Survey. *Journal of Machine Learning Research*, 21(70): 1–73.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Li, H.; Wang, X.; Zhang, Z.; Ma, J.; Cui, P.; and Zhu, W. 2021a. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering*.

Li, H.; Wang, X.; Zhang, Z.; Yuan, Z.; Li, H.; and Zhu, W. 2021b. Disentangled Contrastive Learning on Graphs. *Neural Information Processing Systems*.

Li, H.; Wang, X.; Zhang, Z.; and Zhu, W. 2022a. OOD-GNN: Out-of-Distribution Generalized Graph Neural Network. *IEEE Transactions on Knowledge and Data Engineering*.

Li, H.; Wang, X.; Zhang, Z.; and Zhu, W. 2022b. Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987*.

Li, H.; Zhang, Z.; Wang, X.; and Zhu, W. 2022c. Disentangled Graph Contrastive Learning With Independence Promotion. *IEEE Transactions on Knowledge and Data Engineering*.

Li, H.; Zhang, Z.; Wang, X.; and Zhu, W. 2022d. Learning Invariant Graph Representations Under Distribution Shifts. In *Neural Information Processing Systems*.

Li, Q.; Shang, Y.; Qiao, X.; and Dai, W. 2020. Heterogeneous dynamic graph attention network. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*, 404–411.

Li, Y.; and King, I. 2020. Autograph: Automated graph neural network. In *International Conference on Neural Information Processing*, 189–201.

Li, Y.; Wen, Z.; Wang, Y.; and Xu, C. 2021c. One-shot graph neural architecture search with dynamic search space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 8510–8517.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Luo, W.; Zhang, H.; Yang, X.; Bo, L.; Yang, X.; Li, Z.; Qie, X.; and Ye, J. 2020. Dynamic heterogeneous graph neural network for real-time event prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3213–3223.

Nunes, M.; and Pappa, G. L. 2020. Neural architecture search in graph neural networks. In *Brazilian Conference on Intelligent Systems*, 302–317.

Pan, Z.; Ke, S.; Yang, X.; Liang, Y.; Yu, Y.; Zhang, J.; and Zheng, Y. 2021. AutoSTG: Neural Architecture Search for Predictions of Spatio-Temporal Graph. In *Proceedings of the Web Conference 2021*, 1846–1855.

Qin, Y.; Wang, X.; Cui, P.; and Zhu, W. 2021a. GQNAS: Graph Q Network for Neural Architecture Search. In Bailey, J.; Miettinen, P.; Koh, Y. S.; Tao, D.; and Wu, X., eds., *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*, 1288–1293. IEEE.

Qin, Y.; Wang, X.; Zhang, Z.; Xie, P.; and Zhu, W. 2022a. Graph Neural Architecture Search Under Distribution Shifts. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 18083–18095. PMLR.

Qin, Y.; Wang, X.; Zhang, Z.; and Zhu, W. 2021b. Graph differentiable architecture search with structure learning. *Advances in Neural Information Processing Systems*, 34.

Qin, Y.; Zhang, Z.; Wang, X.; Zhang, Z.; and Zhu, W. 2022b. NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search. *CoRR*, abs/2206.09166.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 519–527.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Berg, R. v. d.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, 593–607.

Shi, M.; Wilson, D. A.; Zhu, X.; Huang, Y.; Zhuang, Y.; Liu, J.; and Tang, Y. 2022. Genetic-GNN: Evolutionary architecture search for Graph Neural Networks. *Knowledge-Based Systems*, 108752.

Skarding, J.; Gabrys, B.; and Musial, K. 2021. Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE Access*, 9: 79143–79168.

Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; and Wu, T. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11): 992–1003.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Wang, X.; Bo, D.; Shi, C.; Fan, S.; Ye, Y.; and Philip, S. Y. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*.

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*, 2022–2032.

Wang, Y.; Chang, Y.-Y.; Liu, Y.; Leskovec, J.; and Li, P. 2021. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974*.

Wistuba, M.; Rawat, A.; and Pedapati, T. 2019. A survey on neural architecture search. *arXiv:1905.01392*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*.

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

Xue, G.; Zhong, M.; Li, J.; Chen, J.; Zhai, C.; and Kong, R. 2022. Dynamic network embedding survey. *Neurocomputing*, 472: 212–223.

Xue, H.; Yang, L.; Jiang, W.; Wei, Y.; Hu, Y.; and Lin, Y. 2020. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 282–298.

Yang, C.; Xiao, Y.; Zhang, Y.; Sun, Y.; and Han, J. 2020. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering*.

Yang, M.; Zhou, M.; Kalander, M.; Huang, Z.; and King, I. 2021. Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1975–1985.

Yuan, C.; Li, J.; Zhou, W.; Lu, Y.; Zhang, X.; and Hu, S. 2020. DyHGCN: A dynamic heterogeneous graph convolutional network to learn users' dynamic preferences for information diffusion prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 347–363.

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*.

Zhang, W.; Lin, Z.; Shen, Y.; Li, Y.; Yang, Z.; and Cui, B. 2022a. DFG-NAS: Deep and Flexible Graph Neural Architecture Search. *CoRR*, abs/2206.08582.

Zhang, Z.; Cui, P.; and Zhu, W. 2020. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Zhang, Z.; Wang, X.; Zhang, Z.; Li, H.; Qin, Z.; and Zhu, W. 2022b. Dynamic Graph Neural Networks Under Spatio-Temporal Distribution Shift. In *Thirty-Sixth Conference on Neural Information Processing Systems*.

Zhang, Z.; Wang, X.; and Zhu, W. 2021. Automated machine learning on graphs: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*.

Zhang, Z.; Zhang, Z.; Wang, X.; and Zhu, W. 2022c. Learning to Solve Travelling Salesman Problem with Hardness-adaptive Curriculum. *arXiv preprint arXiv:2204.03236*.

Zhao, Y.; Wang, D.; Gao, X.; Mullins, R.; Lio, P.; and Jamnik, M. 2020. Probabilistic dual network architecture search on graphs. *arXiv:2003.09676*.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81.

Zhou, K.; Song, Q.; Huang, X.; and Hu, X. 2019. Auto-gnn: Neural architecture search of graph neural networks. *arXiv:1909.03184*.

Zhou, Y.; Wang, X.; Chen, H.; Duan, X.; Guan, C.; and Zhu, W. 2022. Curriculum-NAS: Curriculum Weight-Sharing Neural Architecture Search. In *Proceedings of the 30th ACM International Conference on Multimedia*, 6792–6801.

Zhu, Y.; Lyu, F.; Hu, C.; Chen, X.; and Liu, X. 2022. Learnable Encoder-Decoder Architecture for Dynamic Graph: A Survey. *arXiv preprint arXiv:2203.10480*.