Few-shot Link Prediction in Dynamic Networks

Cheng Yang[1,3], Chunchen Wang[1,2], Yuanfu Lu[2], Xumeng Gong[1],
Chuan Shi[1,3] †, Wei Wang[2], Xu Zhang[2]
[1]Beijing University of Posts and Telecommunications, China
[2]WeChat Search Application Department, Tencent Inc. China
[3]Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, China
{yangcheng,wangchunchen,Xumeng1141,shichuan}@bupt.edu.cn,{lucasyflu,unoywang,xuonezhang}@tencent.com

ABSTRACT

Dynamic link prediction, which aims at forecasting future edges of a node in a dynamic network, is an important problem in network science and has a wide range of real-world applications. A key property of dynamic networks is that new nodes and links keep coming over time and these new nodes usually have only a few links at their arrivals. However, how to predict future links for these few-shot nodes in a dynamic network has not been well studied. Existing dynamic network representation learning methods were not specialized for few-shot scenarios and thus would lead to suboptimal performances. In this paper, we propose a novel model based on a meta-learning framework, dubbed as MetaDyGNN, for few-shot link prediction in dynamic networks. Specifically, we propose a meta-learner with hierarchical time interval-wise and node-wise adaptions to extract general knowledge behind this problem. We also design a simple and effective dynamic graph neural network (GNN) module to characterize the local structure of each node in meta-learning tasks. As a result, the learned general knowledge serves as model initializations, and can quickly adapt to new nodes with a fine-tuning process on only a few links. Experimental results show that our proposed MetaDyGNN significantly outperforms state-of-the-art methods on three publicly available datasets.

KEYWORDS

link prediction, dynamic network, few-shot prediction, meta-learning, graph neural networks

ACM Reference Format:
Cheng Yang[1,3], Chunchen Wang[1,2], Yuanfu Lu[2], Xumeng Gong[1], Chuan Shi[1,3] †, Wei Wang[2], Xu Zhang[2]. 2022. Few-shot Link Prediction in Dynamic Networks. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488560.3498417

† The corresponding author.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498417

1 INTRODUCTION

Dynamic or temporal networks can characterize a variety of real-world systems evolving over time, and have been extensively studied over decades [10]. Dynamic link prediction, which aims at predicting future relationships between a set of nodes, has been recognized as a fundamental task to understand the evolutionary patterns of a dynamic network [4]. With a wide range of applications such as product recommendation [38] and friend suggestion [18, 29, 30], link prediction in dynamic networks has gained much attention in recent years.

A key characteristic of dynamic networks is that new nodes and links keep coming over time [10]. These new nodes usually have only a few links at their arrivals, *e.g.,* new users in a social network or new researchers in an academic collaboration network only have a couple of friends or coauthors at the beginning. Due to the wide existence of such new nodes or cold-start nodes in real-world systems [12, 33], it is worth studying how to predict links for the nodes with only a few observed connections in dynamic networks, *i.e., few-shot link prediction in dynamic networks.* However, to the best of our knowledge, as an important and realistic scenario, the problem has not been specifically designed in existing graph representation learning methods.

Recently, a promising direction of dynamic network representation learning techniques, *i.e.,* dynamic graph neural network (GNN) [25, 39], has achieved state-of-the-art (SOTA) performance on node classification and link prediction of dynamic networks. Despite the significant improvements, existing dynamic GNNs treated dynamic link prediction as a supervised problem and were not specialized for few-shot scenarios. Specifically, the prediction losses corresponding to a few-shot node are likely to be dominated by the overall loss. Hence, in a dynamic GNN model, the trainable parameters shared across all nodes could not generalize well for few-shot nodes, and node-specific trainable parameters of few-shot nodes could not be sufficiently trained. Therefore, the above limitations of existing dynamic GNNs will lead to suboptimal performances.

On the other hand, as a widely used technique for few-shot problems, meta-learning [36] can extract general knowledge across different training tasks and quickly adapt it to few-shot testing tasks, and has been integrated with GNNs for few-shot prediction in static networks recently [2, 3, 11, 40, 41]. However, most of them focused on node/graph classification [3, 40, 41]. Though two very recent work [2, 11] can handle few-shot link prediction, they focused on static network modeling and thus failed to consider the time-varying nature of node preferences and temporal dependency of link formation in dynamic networks, which is critical for dynamic link prediction [10]. Fig. 1 (a) presents an illustrative

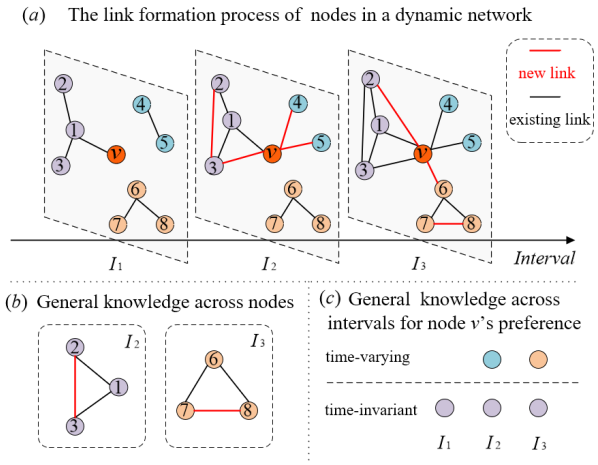


Figure 1: An illustrative example of a dynamic network with 3 snapshots (top), and two levels of general knowledge that can be extracted for link prediction (bottom). The knowledge can be divided into node-wise general knowledge (bottom left), *e.g.,* the law of triadic closure, and time interval-wise general knowledge of a specific node (bottom right), *e.g.,* a node's time-invariant preference.

example of the link formation process in a dynamic network. We can find that the centering node v's neighbor preference changes in different time intervals. If we simplify a dynamic network into a static one, a lot of time-varying information will be discarded, which makes it hard to learn valuable time-invariant knowledge for future predictions. Hence, how to extract general knowledge in the formation of dynamic links via meta-learning, is still a challenge remaining unsolved. Also, it is not trivial to simply combine meta-learning frameworks (*e.g.,* MAML [5]) with existing dynamic GNNs (*e.g.,* TGAT [39]) whose sophisticated architectures cannot quickly adapt to few-shot nodes in meta-learning. Therefore, how to tailor dynamic GNNs for better generalization ability in meta-learning settings is another challenge to be addressed.

In this paper, we propose a novel model named MetaDyGNN, which can take advantage of both meta-learning and graph neural network techniques, for few-shot link prediction in dynamic networks. To address the first challenge, we assume that there are two levels of general knowledge that can be extracted via meta-learning for dynamic link prediction, as shown in Fig 1. The first level in Fig. 1 (b) is the knowledge shared across different nodes, *e.g.,* the law of triadic closure [28] that two nodes with shared neighbors tend to get connected later. The second level in Fig. 1 (c) is the knowledge of a specific node's time-invariant preference. To extract the above general knowledge for fast adaption, we formalize each task in our meta-learning framework as the temporal preference learning of a single node, and propose a hierarchically adaptive meta-learner with both time interval-wise adaption and node-wise (or task-wise) adaption. As a result, during the meta-testing phase, our model can quickly adapt to a *new node* (node-wise adaption) for predicting its potential links in the *future* (time interval-wise adaption), with only a few links of the new node. To address the second challenge, we design a lightweight dynamic GNN module to characterize the local structure of each node in meta-learning tasks. In detail, our module employs attention mechanism to take advantage of time encoding, node features and edge features for node representation learning. Compared with TGAT [39], our module is more efficient and effective in the meta-learning framework. We conduct experiments on three publicly available dynamic network datasets. Experimental results show that our proposed MetaDyGNN significantly outperforms previous methods as well as the simple combinations of meta-learning and GNNs.

To conclude, our contributions are as follows:

- We propose a novel method MetaDyGNN based on meta-learning to address the few-shot link prediction problem in dynamic networks. To the best of our knowledge, MetaDyGNN is the first model specifically designed for this important and realistic scenario.
- The proposed model specializes meta-learning and dynamic graph neural network techniques to extract hierarchical knowledge for few-shot dynamic link prediction, and thus can go beyond a simple combination of existing models.
- Experimental results on three publicly available dynamic network datasets show that MetaDyGNN has a relative improvement by up to 9.4% in terms of AUC over SOTA methods.

2 RELATED WORK

Generally, there are two lines of work related to the few-shot link prediction problem in dynamic networks.

2.1 Meta-learning on Graphs

Meta-learning [36] aims at extracting general knowledge (or priors) across different tasks, which can be quickly adapted to a new task with only a few examples. For instance, MAML [5] is a popular meta-learning framework for deep neural networks, which learned the general knowledge across tasks as neural network parameter initialization. At the meta-testing phase, MAML can fine-tune the parameters with a small amount of training instances and thus adapt to a new learning task efficiently.

Recently, meta-learning was integrated with graph neural network models for few-shot predictions on graphs. Meta-GNN [41] was the first in this line of works, which employed MAML [5] for few-shot node classification in a single graph. Chauhan et al. [3] proposed a meta-learning framework based on super-class prototype modeling, and can be combined with conventional GNN models such as GCN [15] and GAT [35] for graph classification. GFL [40] also targeted on categorizing graphs with node-level and graph-level knowledge transfer. The above studies all focused on classification problem.

In terms of few-shot link prediction, Meta-Graph [2] combined meta-learning and GCN [15] for link prediction across multiple graphs, which can not deal with link prediction problem in a single graph. G-Meta [11] was a very recent work based on local subgraph modeling, which can be efficiently applied on either classification or link prediction tasks. Besides, META-MGNN [7] tried to combine the meta-learning and GCN for graph representation learning to do molecular property prediction. There are also some works proposed

for representation learning of tail nodes [19], bipartite networks [17, 31], heterogeneous information networks [20].

However, all these works were designed for static networks and failed to capture the key characteristics for dynamic link prediction [10], *e.g.,* the time-varying nature of node preferences and temporal dependency of link formation.

2.2 Dynamic Network Representation Learning

Learning node representations in a dynamic network has become a rising topic during the last five years. Conventional embedding-based methods employed a shallow neural architecture, and explored the law of triadic closure [42], Hawkes process [43], Macro- and Micro dynamics [21] and temporal random walks [24, 37] to capture the temporal information. DyRep [32] and JODIE [16] further utilized recurrent neural networks (RNNs) to update node representations.

In contrast, many works were fully based on deep neural architectures and have generally better performance with larger model capacity. These works usually formalized a dynamic network as a sequence of graph snapshots, and employed GNNs to model structural context in each snapshot and stacked RNNs on the top of GNNs to further model temporal context [6, 8, 22, 25, 27]. Besides, TGAT [39] integrated the sampling strategy in GraphSAGE [9] and multi-head attention mechanism [34] in GAT [35] to build node representations in a dynamic network. TGN [26] utilized memory network to model the temporal evolution of node representations.

However, these models were designed and trained under the supervised setting, and did not focus on the predictions for few-shot nodes. Also, they usually took the entire dynamic graph as the input and need relatively large computing resources, which would harm the generalization ability under a simple combination of meta-learning techniques.

3 PRELIMINARIES

In this section, we will formalize the few-shot link prediction problem in dynamic networks, *i.e.,* predicting future edges for the new nodes with only a few links. A notation table is available in the supplementary material.

Firstly, a dynamic network can be defined as follows [14]:

Definition 1. ***Dynamic Network.*** A dynamic network can be denoted as $G = (V, E)$ where V and E are the sets of nodes and temporal edges. Each temporal edge $e = (v_i, v_j, t_e) \in E$ refers to an event that node $v_i \in V$ connects with node $v_j \in V$ at timestamp t_e. Note that nodes v_i and v_j could build multiple edges at different timestamps.

As a dynamic network evolves, new nodes continue to join the network and build links. Here we define the new nodes in a dynamic network as:

Definition 2. ***New Nodes in a Dynamic Network.*** Given a dynamic network $G = (V, E)$ and a timestamp t, nodes that first join the network G at a future timestamp $t' > t$, are defined as new nodes after timestamp t.

Then we formalize the link prediction for new nodes in a dynamic network as few-shot link prediction problem:

Definition 3. ***Few-shot Link Prediction in a Dynamic Network.*** Given a dynamic network $G = (V, E)$ and a timestamp t, we use $V_{new} \subset V$ to denote the set of nodes appearing after timestamp t. Given all the nodes and their links before timestamp t, our goal is to predict future links for each new node $v \in V_{new}$ based on its first K connections.

Here K is usually a small number less than 10, and a few-shot problem can also be called as a K-shot one, *e.g.,* 5-shot when $K = 5$. We will compare methods under different Ks in our experiments.

4 METHODOLOGY

In this section, we present a novel method MetaDyGNN to address the few-shot link prediction problem in dynamic networks. Specifically, our framework is designed and built to address two challenges:

• How to extract general knowledge in the formation of dynamic links via meta-learning?

• How to tailor dynamic GNNs for better generalization ability in meta-learning settings?

An overview of the proposed MetaDyGNN is shown in Fig. 2.

4.1 Task Formulation

First, we will briefly introduce the basic setup of meta-learning. A meta-learning framework will divide the training data into a set of tasks and treat these tasks as training instances. In each individual task, the "training" and "testing" examples are respectively named as the support and query sets to avoid confusion. During the meta-training phase, the framework will learn the knowledge shared across different tasks as model parameters. Then at the meta-testing phase, the parameters can quickly adapt to new tasks with only a few examples for fine-tuning.

To extract the two levels of general knowledge about dynamic link formation as shown in Fig. 1, each task in our meta-learning framework is formalized as the temporal preference learning of a single node, *i.e.,* dividing training data into node-level tasks to capture node-wise knowledge. In specific, we regard each task as a binary classification problem between a node's positive temporal links and negative ones. In order to acquire the knowledge of each node's time-invariant preference, we further divide the time span of each node v into n equidistant time intervals $I_v = \{I_v^1, I_v^2, \ldots, I_v^n\}$ and correspondingly split the links into n sets. For task $T_v = (S_v, Q_v)$ of node v, the support set S_v is defined as:

$$\mathcal{S}_v = \{\mathcal{S}_v^1, \mathcal{S}_v^2, \ldots, \mathcal{S}_v^{n-1}\}, \tag{1}$$

where S_v^i is the set of links in the i^{th} interval and query set Q_v is the set of links in the last interval I_v^n. Now the last piece for task formulation is the selection of positive and negative links.

For meta-training, to enable the ability of few-shot link prediction, we need to sample a small subset of dynamic links as each node's positive links, since the degrees of existing nodes are usually much larger than those of new nodes. In detail, we will sample K links as each node's support set $\mathcal{S}_v$ in every epoch, and make sure the numbers of links in different time intervals are roughly equal. Besides, another K links will be sampled as the query set Q_v.

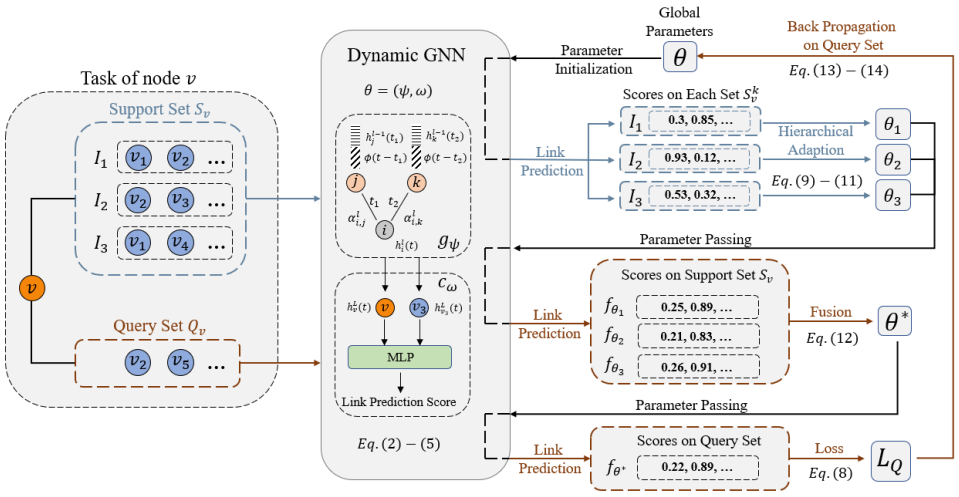


Figure 2: The overall framework of our proposed MetaDyGNN during the meta-training phase. For each task T_v of node v, the global parameter in the dynamic GNN θ will be adapted for each interval S_v^k via hierarchical adaptions. Then we will fuse the adapted parameters across different intervals as θ^*, where the weight of each adapted parameter is determined by its performance on the entire support set S_v. Finally, we will evaluate the loss of the fused parameter θ^* on query set Q_v, and perform back-propagation.

For meta-testing, we use K dynamic links of each new node as the support set for fine-tuning (*i.e.,* K-shot link prediction), and employ all the remaining links as the query set for evaluation.

For negative links, we sample the same number positive ones by following the same sampling strategy [21, 23] based on node degree distribution.

4.2 Link Prediction based on Dynamic GNN

In this subsection, we propose a dynamic GNN module to solve each node-level link prediction task in our meta-learning framework. The proposed module integrates structural information, timestamp information and node/edge features for representing each node in a simple and elegant way. Then a pair of node representations will be fed into a classifier to judge whether a temporal link $e = (v, v', t_e)$ is true or not. Formally, the dynamic GNN module f_θ includes a node encoder g_ψ and a classifier c_ω.

Node encoder g_ψ. The representation of node v at timestamp t_e is aggregated from its temporal neighbors $\mathcal{N}_v(t)$, *i.e.,* neighbors before timestamp t and v itself. Note that we always set $t \leq t_e$, *e.g.,* the start timestamp of the corresponding time interval. Firstly, we construct node v's embedding $\mathbf{h}_v^0$ by a linear transformation of its features $x_v \in \mathbb{R}^d$, based on a shared weight matrix $\mathbf{W}_{node} \in \mathbb{R}^{d \times d'}$, where d is the size of node features and d' is the size of node embeddings. For time encoder $\Phi(\cdot)$, we adopt random Fourier features for encoding [13, 39] which can approach any positive definite kernels according to the Bochner's theorem [1].

Now we will start by describing a single dynamic GNN layer. In specific, we propose to use an attention mechanism to calculate a different weight for each temporal neighbor by the corresponding node embeddings and time encodings. Hence, for node v at time t, we can get its representation by encoder g_ψ:

$$\begin{aligned}\mathbf{h}_v^l(t, t_e) &= g_\psi(v, \mathcal{N}_v(t), t_e) \\ &= \sigma_1\left(\left(\sum_{j \in \mathcal{N}_v(t)} \alpha_{v,j}^l \left(\mathbf{h}_j^{l-1}(t_{v,j}) \| \Phi(t_e - t_{v,j})\right)\right)\mathbf{W}\right),\end{aligned} \tag{2}$$

where l means the l^{th} layer of the GNN, σ_1 is the activation function (here we use $ReLU$), t_e is the timestamp of the interaction between v and its neighbor j, $\|$ is the concatenation operator, $\mathbf{W} \in \mathbb{R}^{2d' \times d'}$ is a shared weight matrix, the attention score $\alpha_{v,j}^l$ of node v's temporal neighbor j

$$\alpha_{v,j}^l = \text{softmax}\left(q_{v,j}^l\right) = \frac{\exp\left(q_{v,j}^l\right)}{\sum_{k \in \mathcal{N}_v(t)} \exp\left(q_{v,k}^l\right)}, \tag{3}$$

$$q_{v,j}^l = \vec{a}\left(\mathbf{h}_v^0 \| \Phi(0) \| \mathbf{h}_j^{l-1}(t_e) \| \Phi(t_e - t_{v,j})\right), \tag{4}$$

where $q_{v,j}^l$ indicates the importance of node j to node v and $\vec{a} \in \mathbb{R}^{4d'}$ is the shared parameter vector in the attention mechanism.

If edge features are also available, we can simply extend our model by concatenating the additional edge feature $x_{v,j}(t_{v,j})\mathbf{W}_{edge}$ between v and j at timestamp $t_{v,j}$ to the computation of Eq. (4), where $\mathbf{W}_{edge} \in \mathbb{R}^{d \times d'}$ is a shared weight matrix.

Classifier c_ω. Given the representations of node v and v' at timestamp t as $\mathbf{h}_v^L(t, t_e)$ and $\mathbf{h}_{v'}^L(t, t_e)$, where L is the number of dynamic GNN layers, we can estimate the probability that temporal link $e = (v, v', t_e)$ is true as:

$$\begin{aligned}p_e &= c_\omega\left(\mathbf{h}_v^L(t, t_e), \mathbf{h}_{v'}^L(t, t_e)\right) \\ &= \sigma_2(\text{MLP}\left(\mathbf{h}_v^L(t, t_e) \| \mathbf{h}_{v'}^L(t, t_e)\right)),\end{aligned} \tag{5}$$

where c_ω is the classifier parameterized by ω, σ_2 is the *sigmoid* function, MLP$(\cdot)$ is a two-layer multilayer perceptron, and $\|$ is the concatenation operator. Then we minimize the loss for each temporal edge e to train the model:

$$\mathcal{L}_e = -y_e \log p_e - (1 - y_e) \log(1 - p_e), \tag{6}$$

where $y_e = 1$ if e exists in edge set E and $y_e = 0$ otherwise.

Compared with typical dynamic GNN models, *e.g.,* EvolveGCN [25] which stacked LSTM on top of GCN to encode structural and temporal information, or TGAT [39] which employed multi-head attention mechanism and layer-wise MLP operations for encoding, our module is more lightweight and efficient to calculate gradients for fine-tuning the parameters in the meta-learning framework.

4.3 Hierarchically Adaptive Meta-Learner

Now we will introduce our hierarchically adaptive meta-learner, which conducts time interval-wise and node-wise adaptions to fine-tune the global parameters in the dynamic GNN module for a given task.

4.3.1 Time Interval-wise Adaption. The first-level adaption is to fine-tune the global parameter ψ of node encoder for each time interval. Recall that the support set $\mathcal{S}_v$ of task T_v contains several sets $\mathcal{S}_v^1, \mathcal{S}_v^2, \ldots, \mathcal{S}_v^{n-1}$ divided by time intervals. The time interval-wise adaption updates the global parameter ψ of node encoder to characterize node v's preference in the k^{th} time interval, based on the loss of each set $\mathcal{S}_v^k$ in $\mathcal{S}_v$.

In specific, for each edge $e = (v, v', t_e) \in \mathcal{S}_v^k$, node v's representation is computed as:

$$\mathbf{h}_v^L(t^k, t_e) = g_\psi \left(v, \mathcal{N}_v(t^k), t_e \right), \tag{7}$$

where t^k means the start timestamp of the k^{th} interval. Then we can calculate the loss based on node v's positive and negative dynamic links as:

$$\mathcal{L}\left(\psi, \omega, S_v^k\right) = -\sum_{e \in \{S_v^k \cap E\}} \log(p_e) - \sum_{e \in \{S_v^k \setminus E\}} \log(1 - p_e), \tag{8}$$

where $p_e = h_\omega(\mathbf{h}_v^L(t^k, t_e), \mathbf{h}_{v'}^L(t^k, t_e))$ represents the predicted possibility that dynamic edge $e = (v, v', t_e)$ is true.

Then we adapt the global parameter ψ through several gradient descent steps for specialized parameters for node v in the k^{th} time interval:

$$\psi_v^k = \psi - \beta \frac{\partial \mathcal{L}\left(\psi, \omega, S_v^k\right)}{\partial \phi}, \tag{9}$$

where β is the learning rate for time interval-wise adaption.

4.3.2 Node-wise Adaption. The second-level adaption is to fine-tune the global parameter ω of classifier based on adapted parameter ψ_v^k. In specific, node-wise adaption projects the global parameter ω to ω_v for specialization, and then adapts to each set S_v^k with several gradient descent steps:

$$\omega_v = \omega + \mathbf{h}_v^0 \cdot W_\omega, \tag{10}$$

$$\omega_v^k = \omega_v - \eta \frac{\partial \mathcal{L}\left(\psi_v^k, \omega_v, S_v^k\right)}{\partial \omega_v}, \tag{11}$$

where W_ω is a projection matrix and η is the learning rate for node-wise adaption.

4.3.3 Optimization. So far, we have adapted the global parameter θ to $\theta_v^k = \{\psi_v^k, \omega_v^k\}$ for the k^{th} time interval of node v. In order to optimize the meta-learner on the query set Q_v of every node v, we first fuse the adapted parameters for each node v as:

$$\psi_v^* = \sum_{k=1}^{n-1} a_v^k \psi_v^k, \omega_v^* = \sum_{k=1}^{n-1} a_v^k \omega_v^k, \tag{12}$$

where $a_v^k = \text{softmax}(-\mathcal{L}\left(\psi_v^k, \omega_v^k, S_v\right))$ is the weight of θ_v^k measured by its performance on the entire support set S_v. Compared with the trivial average aggregation, the weight a_v^k can empirically accelerate the convergence of the meta-training phase.

Then we aim to minimize the loss on each query set Q_v, and update global parameter $\theta = (\psi, \omega)$ as well as the projection matrix W_ω through back-propagation:

$$\theta \leftarrow \theta - \gamma \nabla_\theta \sum_{v \in V} \mathcal{L}\left(\psi_v^*, \omega_v^*, Q_v\right), \tag{13}$$

$$W_\omega \leftarrow W_\omega - \gamma \nabla_{W_\omega} \sum_{v \in V} \mathcal{L}\left(\psi_v^*, \omega_v^*, Q_v\right), \tag{14}$$

where γ is the learning rate of the meta-learner.

Recall our motivation illustrated in Fig. 1, the learned global parameters ψ encodes the general knowledge across nodes and time intervals; and ω_v encodes the time-invariant knowledge with respect to node v across different intervals. The detailed meta-training process is shown with Algorithm 1, and we will analyze the model efficiency in the experiments. Code and data are available at https://github.com/BUPT-GAMMA/MetaDyGNN.

Algorithm 1 The Meta-training Phase of MetaDyGNN

Require: Dynamic graph G; the numbers of adaption steps step_i and step_n; randomly initialized $\theta = \{\psi, \omega\}$ and W_ω.
Ensure: The learned parameter $\theta = \{\psi, \omega\}$ and W_ω of the model.

```
Construct support set S_v, and query set Q_v for each node v, where S_v = (S_v^1, S_v^2, ..., S_v^{n-1});
while not done do
    Set the batch of nodes for meta-training;
    for all node v in the batch do
        for all dynamic link set S_v^k ∈ S_v do
            Calculate L(ψ, ω, S_v^k) by Eq. (8);
            Perform time interval-wise adaption by Eq. (9) with step_i updates and obtain ψ_v^k;
            Calculate L(ψ_v^k, ω, S_v^k) by Eq. (8);
            Perform node-wise adaption by Eq. (10-11) with step_n updates and obtain ω_v^k;
        end for
        Calculate fused parameters ψ_v^* and ω_v^* by Eq. (12);
    end for
    Back-propagate the global parameter θ, W_ω by Eq. (13-14);
end while
```

5 EXPERIMENTS

We conduct experiments on three publicly available datasets of dynamic networks, including Wikipedia, Reddit, and DBLP. We compare the proposed method MetaDyGNN with three types of baseline methods on the few-shot link prediction task, and present experimental results to answer four research questions: (RQ1) How does MetaDyGNN performs compared with SOTA methods? (RQ2) To what extent does MetaDyGNN benefit from meta-learning and hierarchical adaption? (RQ3) How does the hyperparameters affect the performance of MetaDyGNN? (RQ4) How is the model efficiency of MetaDyGNN compared with SOTA methods?

Table 1: Statistics of the datasets.

Dataset	# Node	# Dynamic edge	#Timestamp
Wikipedia	9,227	157,475	continuous
Reddit	10,984	672,448	continuous
DBLP	28,085	286,894	27 snapshots

5.1 Experimental Setup

5.1.1 Datasets. We employ three benchmark datasets which have been used for evaluating dynamic network embeddings and dynamic GNNs. The statistics are shown in Table 1.

• **Wikipedia**[1] [39] is a dataset containing edited encyclopedia pages and their editors, where nodes represent wiki pages and their editors. Each dynamic link indicates a timestamped editing interaction.

• **Reddit**[2] [39] is a dataset of the active users on Reddit and their posts under subreddits. The nodes are users and their posts. The links represent timestamped actions of posting requests.

• **DBLP**[3] [21] is an academic collaboration network, where the nodes are authors and each link represents a co-authored paper at the corresponding year.

5.1.2 Baseline Approaches. Our proposed method will be compared with the following three types of baselines: **(1) Static GNNs**, including GraphSAGE [9] and GAT [35]; **(2) Dynamic GNNs**, including DyRep [32], EvolveGCN [25] and TGAT [39]; **(3) Meta-based GNNS**, including Meta-GNN [41], GraphSAGE+MAML [5, 9] and TGAT+MAML [5, 39]. We show the detailed information of baselines in supplementary material.

5.1.3 Evaluation Task. Our evaluation task is link prediction for few-shot nodes that are not observed in the training set. In our experiments, we follow two steps to split the data: (1) we first split all the links into training / validation / testing sets chronologically. In specific, the entire time duration $[0, D]$ is divided into three intervals: $[0, D_{train})$, $[D_{train}, D_{val})$, $[D_{val}, D] = 6 : 2 : 2$. (2) We take the nodes already exist before D_{train} and their associated links in $[0, D_{train})$ as the training set; the nodes whose first appearances fall into the second interval $[D_{train}, D_{val})$ and their associated links in $[D_{train}, D_{val})$ as the validation set; the nodes that only appear after D_{val} and their associated links in $[D_{val}, D]$ as the test set. For the nodes in the validation and test set, we assume that their first $K = 2, 4, 6, 8$ links are known and can be used to adapt parameters for meta-learning based methods or compute node representations for non-meta ones. Then the goal is to identify the future links given the same number of randomly sampled negative links. Note that all methods will output an affinity score for each future/negative link, and the prediction can be treated as either a binary classification problem or a ranking problem. Therefore, we employ three metrics to fully evaluate our model and baseline methods: accuracy (ACC), AUC and Macro-F1. Also, we find that for all the methods except DyRep, it is beneficial to re-train the models with training and validation sets after hyperparameter tuning on the validation set. This is because the temporal context in the validation set is closer and more related to the test set. For a fair comparison, we use the same evaluation procedures for all methods, and report the better result (retrain or not) of each model.

5.1.4 Hyperparameter Settings. We employ Adam optimizer for training, and early stopping strategy with a tolerance of three epochs to select the best epoch. The maximum number of epochs are set to 30. For all three datasets, we set batch size as 64 and embedding dimension as 64. For the temporal node encoder, we also use a two-layer architecture for a fair comparison with baselines. For the sake of efficiency, we sample 16 neighbors in $N_v(t)$ for neighbor aggregation. For the meta-learning part, the number of intervals $n = 3$ and the neighbor dropout as 0.5. We perform a single step of gradient descent for both time interval- and node-wise adaptions. We set the learning rates for meta-learner $\eta = 0.001$, time interval-wise adaption $\alpha = 2e - 4$, and and node-wise adaption $\beta = 0.025$. We conduct experiments on a Linux server with a single GPU (GeForce RTX) and CPU (Intel Xeon E5-2620). We implement the proposed MetaDyGNN with PyTorch 1.4.0. The code and datasets are released on https://github.com/BUPT-GAMMA/MetaDyGNN

5.2 Main Experiments (RQ1)

In this subsection, we present the main results and compare our MetaDyGNN with three types of baselines. Table 2 and 3 respectively present the performances of all methods when $K = 8$ and $K = 2, 4, 6$. The best and second-best results in each column are highlighted in bold font and underlined, respectively. We only present the most challenging baselines in Table 3 for brevity, and have the following observations:

• Our proposed MetaDyGNN significantly outperforms all baseline methods on the three dynamic network datasets by a large margin. In terms of AUC, MetaDyGNN has 6.17/2.56/9.40% relative improvements over the best performed baseline method TGAT under the 8-shot setting, which demonstrates the effectiveness of MetaDyGNN on predicting future links for unseen nodes with only a few links. MetaDyGNN also performs best under the 2-, 4- and 6-shot settings.

• Compared with the static GNN models and their combinations of meta-learning, *i.e.,* GraphSAGE, GAT, Meta-GNN and GraphSAGE+MAML, our model can further capture the temporal dependency via temporal attention layers and time interval-wise adaptions. Thus our method is more powerful in dynamic link prediction, where temporal factors play an important role [10].

[1] http://snap.stanford.edu/jodie/wikipedia.csv
[2] http://snap.stanford.edu/jodie/reddit.csv
[3] https://dblp.uni-trier.de

Table 2: Experimental results of few-shot link prediction on three datasets. Larger scores indicate better performances.

Model/Result	Wikipedia			Reddit			DBLP		
	ACC(%)	AUC(%)	Macro-F1(%)	ACC(%)	AUC(%)	Macro-F1(%)	ACC(%)	AUC(%)	Macro-F1(%)
GraphSAGE	75.84	77.17	74.76	89.32	93.21	88.28	72.17	76.35	71.12
GAT	76.03	78.76	74.94	89.86	93.36	88.64	73.18	77.16	72.56
DyRep	58.01	62.54	56.93	62.14	66.08	61.27	59.14	66.73	57.79
EvolveGCN	56.36	63.15	54.27	58.21	62.64	57.13	57.98	64.14	56.92
TGAT	87.15	90.44	86.89	93.53	95.45	93.27	76.32	81.13	76.26
Meta-GNN	78.92	80.96	77.13	86.27	90.87	85.44	75.48	79.64	74.81
GraphSAGE+MAML	79.54	81.21	77.49	87.41	91.34	85.91	76.18	80.17	75.12
TGAT+MAML	83.29	85.04	83.04	87.46	91.07	87.09	73.86	78.03	72.67
MetaDyGNN	**95.21**	**96.02**	**94.32**	**96.03**	**97.89**	**95.98**	**83.15**	**88.76**	**82.15**
Improvement	*9.25*	*6.17*	*8.55*	*2.67*	*2.56*	*2.91*	*8.95*	*9.40*	*7.72*

Table 3: Experimental results of few-shot link prediction on three datasets. Larger scores indicate better performances.

Model/Result/AUC(%)	2-shot			4-shot			6-shot		
	Wikipedia	Reddit	DBLP	Wikipedia	Reddit	DBLP	Wikipedia	Reddit	DBLP
TGAT	90.20	95.20	81.01	90.21	95.23	80.98	90.23	95.34	81.03
Meta-GNN	75.81	82.96	74.03	77.89	87.69	76.63	79.03	89.92	78.84
GraphSAGE+MAML	76.21	83.34	74.17	78.42	88.31	77.17	79.21	90.34	79.17
TGAT+MAML	80.14	88.67	73.04	81.24	89.67	76.24	83.07	92.67	78.12
MetaDyGNN	**90.31**	**95.76**	**82.14**	**92.75**	**96.87**	**83.83**	**94.82**	**97.21**	**86.24**
Improvement	*0.12*	*0.59*	*1.39*	*2.81*	*1.72*	*3.51*	*5.08*	*1.96*	*6.42*

• Compared with dynamic GNN models, *i.e.,* DyRep, Evolve GCN and TGAT, we design a meta-learner with hierarchical adaptions to extract general knowledge across nodes and time intervals, and thus can adapt our parameters to the new nodes rapidly. In contrast, their parameters are trained and fixed, and thus cannot generalize well in few-shot scenarios. Also, note that DyRep and Evolve GCN have even worse performance than static GNNs. A possible reason is that they both employed RNNs to update node representations over time and could not adapt to the sparsity of few-shot nodes.

• Compared with the combination of dynamic GNNs and meta-learning, *i.e.,* TGAT+MAML, our modification of base model can better fit the framework of meta-learning, which requires less computation resources and can faster adapt to few-shot nodes. As an evidence, TGAT+MAML requires large memory usage and we have to reduce the batch size to fit the limit of GPU devices. Consequently, the integration of TGAT+MAML performs even worse than TGAT. Therefore, the few-shot link prediction problem cannot be addressed by a straightforward integration of existing techniques.

5.3 Analysis of Model Components (RQ2)

In this subsection, we conduct ablation studies to evaluate the performance improvement brought by meta-learning framework and the two-level hierarchical adaptions.

5.3.1 Performance gains from meta-learning framework. We design two ablated model to demonstrate the necessity of meta-learning framework. 1) The first ablated model **MetaDyGNN-BM** only contains the base model without any meta-learning strategy, and is trained in a supervised manner. 2) The second ablated

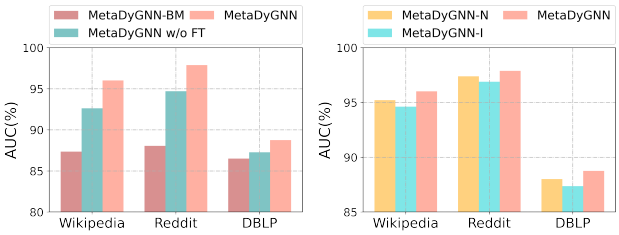


Figure 3: Ablation study of meta-learning strategies (left) and hierarchical adaptions (right).

model **MetaDyGNN w/o FT** removes the fine-tuning of meta-learning in the training procedure, but will update the parameters of the model with few-shot links during meta-testing. Fig. 3 (left) presents the results of our model and two ablated model with metrics ACC/AUC/Macro-F1. Through the experimental results we can conclude that the fine-tuning in both meta-training and meta-testing can improve the prediction performance of our model. Fine-tuning at training phase can help extract general knowledge and fine-tuning at testing phase can help adapt the parameters to a specific node. Therefore, it is necessary to take advantage of meta-learning techniques for the problem.

5.3.2 Performance gains from hierarchical adaptions. We specialize meta-learning for dynamic link prediction by applying the hierarchical adaptions to extract both node-wise and time interval-wise knowledge. We design another two ablated models to verify the effectiveness of our model design. 1) **MetaDyGNN-N** only has

the node-wise adaption and 2) **MetaDyGNN-I** only has the time interval-wise adaption. As shown in Fig. 3 (right), our full model is always better than the two model variants: around 1 – 1.5% improvements on AUC. The results indicate that both node-wise and time interval-wise adaptions contribute to the performance gains by capturing the general knowledge at different levels.

5.4 Analysis of Model Hyperparameters (RQ3)

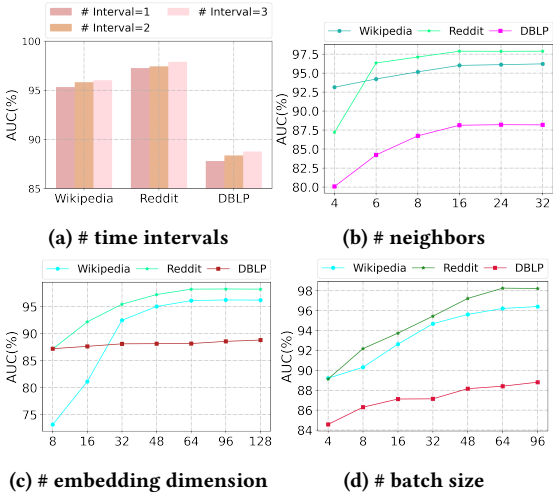


Figure 4: Performance under different hyperparameters.

In this subsection, we will investigate how primary hyper-parameters[4] influence the performance of MetaDyGNN. In specific, we present the analysis of (1) the number of time intervals to split for each support set; (2) the number of neighbors to sample for aggregation in the node encoder; (3) the dimension of node representations; (4) the number of batch size. and (5) the running time analysis. The results are shown in Fig. 4 and we have the following observations:

• Fig. 4 shows that performance can be enhanced when we divide a support set into more time intervals. This is because the time interval-wise adaption can be more fine-grained and better fit the target node. However, as the number of time intervals gets larger, the performance gains will shrink while the time cost of meta-training increases. Thus, we use trisection as a good balance of efficiency and effectiveness.

• Our model performance will be improved and get steady when the number of neighbors to sample for each support set increases to 16, the dimension of node representations and the batch size increase to 64. Thus, MetaDyGNN is relatively stable when these hyperparameters vary within a reasonable range.

5.5 Analysis of Time Efficiency (RQ4)

We present the performance with respect to running time of 6 dynamic graph representation learning methods in Fig. 5, where MetaDyGNN-M replaces the weighted fusion in Eq. (12) by mean operation. We can see that MetaDyGNN spends less time than TGAT and EvolveGCN. Compared with dynamic GNN models which need all the links of a dynamic network as input, we can use less links to gain more accuracy in few-shot link prediction, with the help of sampling strategies. We also test the time cost of TGAT+MAML, which suffers from a sophisticated node encoder and cannot be fit into this figure due to its high complexity. Besides, compared with the variant MetaDyGNN-M, our model converges more quickly and also has slight improvement. The variant MetaDyGNN-N is faster with the second best performance, and can be used if a high demand for efficiency is needed. The above observations demonstrate the efficiency of our proposed method.

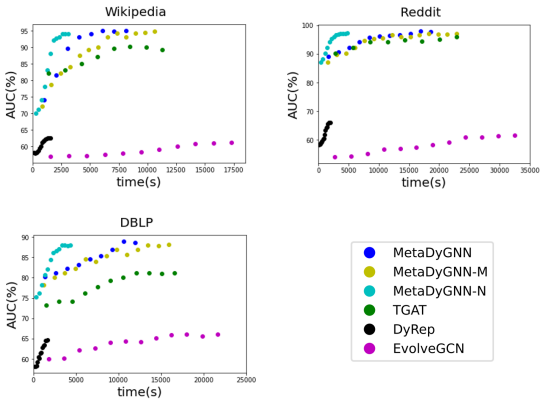


Figure 5: Running time analysis. Each dot corresponds to an epoch of a model.

6 CONCLUSION

In this paper, we propose a novel model named MetaDyGNN to address the few-shot link prediction problem in dynamic networks. We specialize the meta-learning framework for this task by introducing hierarchical time interval-wise and node-wise adaptions. In this way, we can better extract the general knowledge across nodes or time intervals, and quickly adapt to previously unseen nodes with only a few links. Also, we design a simple and effective dynamic GNN module, which can better fit the meta-learning framework than existing dynamic GNN models. We conduct experiments on three publicly available datasets and the results show that our proposed MetaDyGNN significantly outperforms SOTA methods as well as the simple combinations of dynamic GNN models and meta-learning. Ablation studies further demonstrate the effectiveness of each proposed model component.

For future work, an interesting direction is to specialize our model for more specific application scenarios such as recommendation systems. Though a straightforward migration could be feasible, it would be better if user behaviors are further considered.

7 ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. U20B2045, 62002029, 61772082, 61702296), the Fundamental Research Funds for the Central Universities 2020RC23.

[4]We only present the performance under AUC metric due to space limitation, and list the results of ACC/Macro-F1 in the supplementary material.

REFERENCES

[1] Salomon Bochner. 1934. A theorem on Fourier-Stieltjes integrals. *Bull. Amer. Math. Soc.* 40, 4 (1934), 271–276.

[2] Avishek Joey Bose, Ankit Jain, Piero Molino, and William L Hamilton. 2019. Meta-graph: Few shot link prediction via meta learning. *arXiv preprint arXiv:1912.09867* (2019).

[3] Jatin Chauhan, Deepak Nathani, and Manohar Kaul. 2020. Few-Shot Learning on Graphs via Super-Classes based on Graph Spectral Measures. *arXiv preprint arXiv:2002.12815* (2020).

[4] Aswathy Divakaran and Anuraj Mohan. 2019. Temporal link prediction: a survey. *New Generation Computing* (2019), 1–46.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of ICML*. 1126–1135.

[6] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187 (2020), 104816.

[7] Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V Chawla. 2021. Few-Shot Graph Learning for Molecular Property Prediction. *Proceedings of WWW* (2021).

[8] Ehsan Hajiramezanali, Arman Hasanzadeh, Nick Duffield, Krishna R Narayanan, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. *arXiv preprint arXiv:1908.09710* (2019).

[9] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Proceedings of NeurIPS* (2017).

[10] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics reports* 519, 3 (2012), 97–125.

[11] Kexin Huang and Marinka Zitnik. 2020. Graph meta learning via local subgraphs. *arXiv preprint arXiv:2006.07889* (2020).

[12] Vassilis N Ioannidis, Da Zheng, and George Karypis. 2020. Few-shot link prediction via graph neural networks for covid-19 drug-repurposing. *arXiv preprint arXiv:2007.10261* (2020).

[13] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).

[14] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2019. Relational representation learning for dynamic (knowledge) graphs: A survey. *arXiv preprint arXiv:1905.11485* (2019).

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of SIGKDD*. 1269–1278.

[17] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. Melu: Meta-learned user preference estimator for cold-start recommendation. In *Proceedings of SIGKDD*. 1073–1082.

[18] Peng Liu, Lemei Zhang, and Jon Atle Gulla. 2019. Real-time social recommendation based on graph embedding and temporal context. *International Journal of Human-Computer Studies* 121 (2019), 58–72.

[19] Zemin Liu, Wentao Zhang, Yuan Fang, Xinming Zhang, and Steven CH Hoi. 2020. Towards locality-aware meta-learning of tail node embeddings on networks. In *Proceedings of CIKM*. 975–984.

[20] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on heterogeneous information networks for cold-start recommendation. In *Proceedings of SIGKDD*. 1563–1573.

[21] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S Yu, and Yanfang Ye. 2019. Temporal network embedding with micro-and macro-dynamics. In *Proceedings of CIKM*. 469–478.

[22] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NeurIPS*. 3111–3119.

[24] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *Companion Proceedings of WWW*. 969–976.

[25] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of AAAI*, Vol. 34. 5363–5370.

[26] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[27] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of WSDM*. 519–527.

[28] Georg Simmel. 1950. *The sociology of georg simmel*. Vol. 92892. Simon and Schuster.

[29] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of WSDM*. 555–563.

[30] Peijie Sun, Le Wu, and Meng Wang. 2018. Attentive recurrent social recommendation. In *Proceedings of SIGIR*. 185–194.

[31] Qiuling Suo, Jingyuan Chou, Weida Zhong, and Aidong Zhang. 2020. Tadanet: Task-adaptive network for graph-enriched meta-learning. In *Proceedings of SIGKDD*. 1789–1799.

[32] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *Proceedings of ICLR*.

[33] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. In *Proceedings of NeurIPS*. 6907–6917.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *Proceedings of ICLR* (2018).

[36] Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial intelligence review* 18, 2 (2002), 77–95.

[37] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. *arXiv preprint arXiv:2101.05974* (2021).

[38] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of SIGKDD*. 723–732.

[39] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *Proceedings of ICLR* (2020).

[40] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh Chawla, and Zhenhui Li. 2020. Graph few-shot learning via knowledge transfer. In *Proceedings of AAAI*, Vol. 34. 6656–6663.

[41] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. 2019. Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of CIKM*. 2357–2360.

[42] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Proceedings of AAAI*, Vol. 32.

[43] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *Proceedings of SIGKDD*. 2857–2866.

A NOTATION TABLE

Table 4: Descriptions of key notations.

Notations	Descriptions
G	the dynamic network
V, E	the sets of nodes and dynamic links
v, e, t_e	node v, edge e, timestamp t_e of edge e
T_v, S_v, Q_v	the task, support set, and query set of node v
S_v^k	the set of links in the k^{th} interval of S_v
f_θ	dynamic GNN module for link prediction
g_ψ, c_ω	node encoder and classifier in the dynamic GNN
$\theta = (\psi, \omega)$	the parameters of dynamic GNN, ψ is for node encoder and ω is for classifier
$\theta_v^k = (\psi_v^k, \omega_v^k)$	the adapted parameters for the k^{th} interval of node v
a_v^k	the weight of ψ_v^k, ω_v^k during parameter fusion
$\theta_v^* = (\psi_v^*, \omega_v^*)$	the fused parameters for node v
β, η, γ	the learning rates for time interval-wise adaption, node-wise adaption, and meta-learner
$\vec{x}_v$	node feature of node v
$\mathbf{W}_{node}, \mathbf{W}_{edge}$	the parameters to construct node and edge embeddings
$\Phi(\cdot)$	time encoder
$\mathbf{h}_v^L(t, t_e)$	the embedding of node v at timestamp t_e through L-layer dynamic GNN model and t is the timestamp for neighbor aggregation
$\vec{a}$	the shared attention mechanism
$\alpha_{v,j}^l$	the attention score of node v's neighbor j at the l^{th} layer of GNN

B BASELINE DESCRIPTIONS

Here we present the detailed descriptions of baselines as follows:

• **GraphSAGE** [9] can be adopted to learn a few-shot node's embedding by sampling and aggregating its neighborhood. We use the two-layer GraphSAGE as the baseline and follow the other settings in the original paper.

• **GAT** [35] incorporates the multi-head attention mechanism to allocate each neighbor an aggregation weight for node embedding learning. We use the two-layer GAT as our baseline.

• **DyRep** [32] employs RNNs to update node embeddings constrained by a temporal point process. We can get the embeddings for few-shot nodes by aggregating the embeddings of existing nodes.

• **EvolveGCN** [25] updates GCN-based [15] node representations by operating RNNs over discrete snapshots.

• **TGAT** [39] encodes continuous timestamp into vectors and employs multi-head attention mechanism to aggregate its neighborhood with temporal context.

• **Meta-GNN** [41] combines meta-learning framework and GNN to tackle the few-shot node classification problem in a single graph. We adopt its task formulation and loss function according for the requirement of few-shot link prediction.

• **GraphSAGE+MAML** simply combines GraphSAGE [9] as the encoder to learn node representations and MAML [5] to meta-learn the parameters.

• **TGAT+MAML** simply combines TGAT [39] as the encoder to learn temporal node representations and MAML [5] to meta-learn the parameters.

C ADDITIONAL HYPERPARAMETER EXPERIMENTS

C.1 Adaption Steps

Let step_i be the number of time interval-wise adaption steps and step_n be the node-wise adaption steps. Fig. 6 shows the performance of MetaDyGNN when step_i, step_n are set within the range 0 to 3 and the evaluation metric is AUC. We can observe that the performance is stable when both steps range from 1 to 3. Since the increase of adaption steps will lead to an increase in the demand for computing resources, we finally choose one step for both time interval- and node-wise adaptions.

C.2 Results under ACC and Macro-F1 Metrics

Due to space limitation, we put the results of different hyperparameters under ACC and Macro-F1 metrics in Fig. 7 and 8. The conclusions are consistent with the AUC metric in the main document.

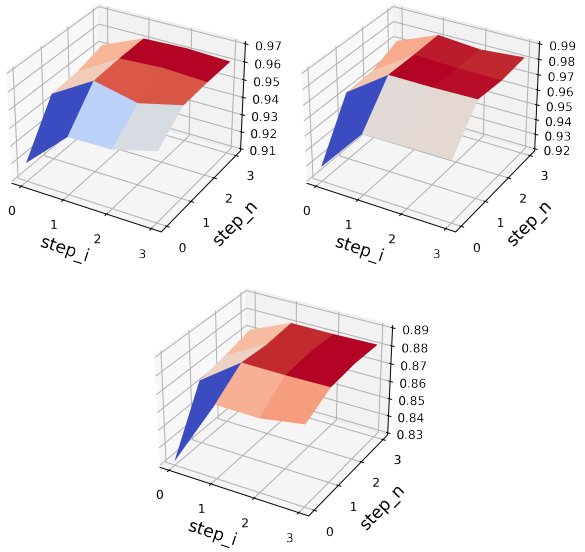
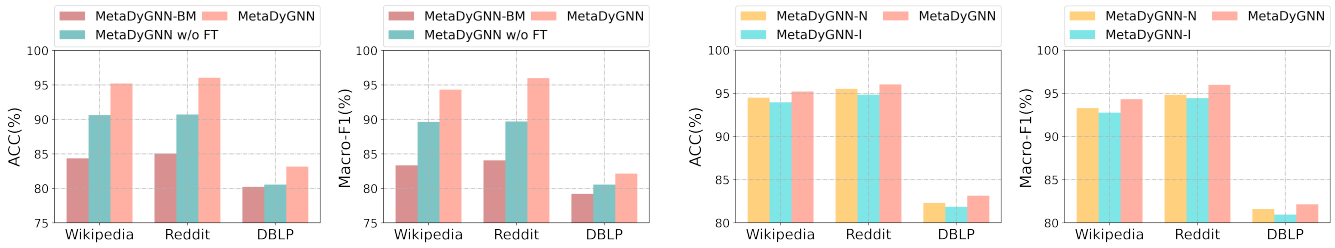
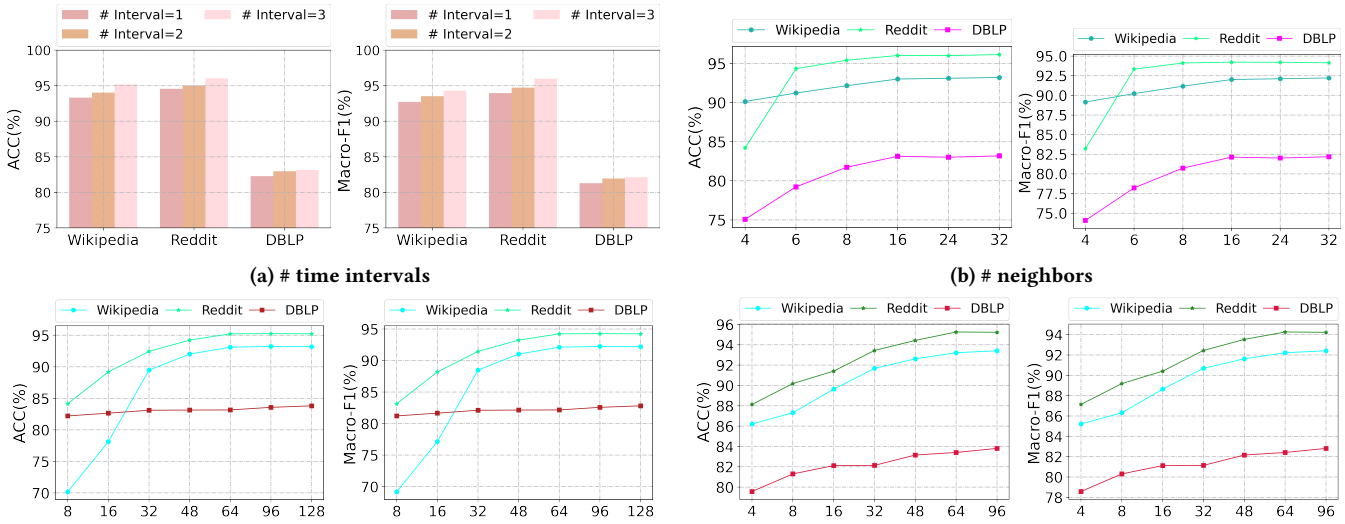


Figure 6: Performance w.r.t the update steps, where step_i is for time interval-wise adaption and step_n is for node-wise adaption. The evaluation metric is AUC.



(a) # meta-learning strategies (b) # hierarchical adaptions

Figure 7: Ablation study of meta-learning strategies (a) and hierarchical adaptions (b).



(a) # time intervals (b) # neighbors

(c) # embedding dimension (d) # batch size

Figure 8: Performance under different hyper-parameters.