Modeling Co-evolution of Attributed and Structural Information in Graph Sequence

Daheng Wang, Zhihan Zhang, Yihong Ma, Tong Zhao, Tianwen Jiang, Nitesh V. Chawla, Meng Jiang

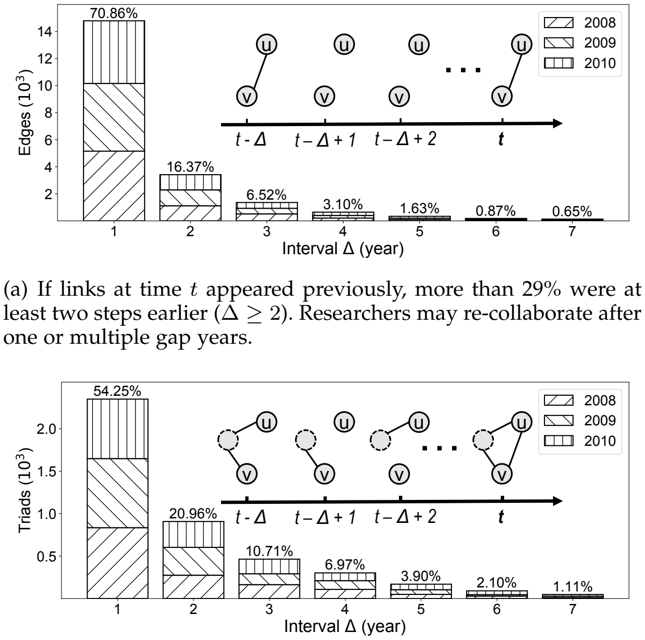
Abstract—Most graph neural network models learn embeddings of nodes in static attributed graphs for predictive analysis. Recent attempts have been made to learn temporal proximity of the nodes. We find that real dynamic attributed graphs exhibit complex phenomenon of *co-evolution* between node attributes and graph structure. Learning node embeddings for forecasting *change of node attributes* and *evolution of graph structure* over time remains an open problem. In this work, we present a novel framework called CoEvoGNN for modeling dynamic attributed graph sequence. It preserves the impact of earlier graphs on the current graph by embedding generation through the sequence of attributed graphs. It has a temporal self-attention architecture to model long-range dependencies in the evolution. Moreover, CoEvoGNN optimizes model parameters jointly on two dynamic tasks, attribute inference and link prediction over time. So the model can capture the co-evolutionary patterns of attribute change and link formation. This framework can adapt to any graph neural algorithms so we implemented and investigated three methods based on it: CoEvoGCN, CoEvoGAT, and CoEvoSAGE. Experiments demonstrate the framework (and its methods) outperforms strong baseline methods on predicting an entire unseen graph snapshot of personal attributes and interpersonal links in dynamic social graphs and financial graphs.

Index Terms—Graph Neural Network, Attributed Graph, Graph Sequence, Evolutionary Prediction.

1 Introduction

Graphs are ubiquitous in the world and real graphs evolve over time via individual behaviors. For example, social network users establish and/or remove links between each other via the behaviors of following, mentioning, replying, and etc. The user's attributes such as textual features from generated content are also changing. These two types of dynamics, social links and user attributes, have impact on each other. Specifically, on academic co-authorship networks, researchers are looking for collaborators (reflected as neighbor nodes) who have similar or complementary knowledge [1] (which may be reflected as published keywords, a type of node attributes). And their personal research topics may change according to new collaborations. The co-evolutionary patterns of node attributes and graph structure are complex yet valuable, and need to be effectively learned by the model for forecasting future attributes and structures in graph-based applications.

Graph Neural Networks (GNNs) have been widely studied for learning representations of nodes from graph data for various tasks such as node classification [3], community detection [4], and link prediction [5], [6]. Most of the existing GNN models assumed that either graph structure or node attributes were static [7]. Particularly, the input is usually an attributed graph. If multiple evolving graphs are present, we can choose to merge all graph snapshots and feed it into one of these models, but the temporal axis would be collapsed and the evolutionary information would be lost. There have been dynamic graph learning methods that explore the idea of combining GNN with recurrent neural network (RNN) for dynamic attributed graphs. WD-GCN [8] stacked an LSTM [9] on top of a GCN [3] module and



(a) If links at time t appeared previously, more than 29% were at least two steps earlier ($\Delta \geq 2$). Researchers may re-collaborate after one or multiple gap years.

(b) If links at time t could be created by closing a triad in previous graphs, more than 45% of the triads were at least two time steps earlier ($\Delta \geq 2$). Researchers who had a common co-author would collaborate in one or multiple years.

Fig. 1. The formation of a new link in co-authorship networks depends on more than one previous graphs. Repeating a link and/or closing a triad that occurred more than two time steps earlier is common for link formation process. Data collected from Microsoft Academic Graph [2].

- *D. Wang, Z. Zhang, Y. Ma, T. Zhao, T. Jiang, N.V. Chawla, and M. Jiang are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, 46556; Additionally, N.V. Chawla is affiliated with Department of Computational Intelligence, Wrocław University of Science and Technology, Wrocław, Poland. E-mail: {dwang8, zzhang23, yma5, tzhao2, tjiang2, nchawla, mjiang2}@nd.edu*
- *Corresponding author: Daheng Wang (dwang8@nd.edu)*

CD-GCN [8] added a skip connection above it. GCRN [10] explored a similar architecture and proposed a modified LSTM by replacing fully connected layers with graph convolution layers [11]. However, these pioneering methods still relied on a fair amount of information in current graphs (though which can be incomplete) and thus were not capable of forecasting an entire snapshot of attributed graph.

Recently, EvolveGCN [12] was proposed to address this issue using GRU [13] to learn the parameter changes in GCN [3] instead of node embedding changes. Specifically, the GCN's weight matrices were treated as hidden states and node embeddings were fed into the GRU at each time. This method iteratively generated node embeddings and, in turn, injected temporal information into the GCN model. However, it has three limitations. First, like other RNN-based methods, it has inherent difficulty in effectively compressing long-range dependencies into the hidden states [14] as well as severe scalability issues as they cannot be easily parallelized [15]. The time complexity is largely intractable: the number of times applying the GRU module grows proportionally with the number of nodes in the data. Second, it assumes the underlying force driving the graph evolution solely comes from the changes of links. It is unaware of the universal co-evolutionary process between node attributes and graph structure. Third, its design is specific to the choice of the GCN algorithm. While different graph neural algorithms (e.g., GCN, GAT [16], GraphSAGE [6]) have different advantages and deliver data-dependent performances, we expect to apply the dynamic framework upon all the algorithms; however, it is unclear how to build EvolveGCN upon any other model that is parameterized by more than one matrix layer-wise such as GraphSAGE.

So, given various attempts that focused around combining RNN and GNN models, modeling the co-evolution of attributed and structural information in graph sequence largely remains an open problem. On one hand, Figure 1 shows that the state of current graph is often influenced by previous graphs multiple steps earlier with a potential time decay factor; Figure 2 shows that the evolution of node attributes and that of links are often highly correlated. So the conventional convolution/aggregation schema along the axis of graph structural depth cannot meet the demand. On the other hand, we have some desired properties of the solution as follows: (1) it should have tractable complexity and good parallelizability for deployment on real dynamic graph sequence; (2) it should be able to forecast future unseen node attributes and graph structure simultaneously; and, (3) it should be able to incorporate an arbitrary static GNN algorithm to aggregate nodes information.

To this end, we propose a novel framework Co-Evolutionary Graph Neural Networks (CoEvoGNN). First, we design an S-stack temporal self attention architecture as the core component of CoEvoGNN. It learns the impact of multiple previous graph snapshots on the current one with self-adapting importance so that it can effectively capture the *evolutionary* patterns in graph sequence. Its temporal self-attention mechanism makes the time complexity grow linearly with the increase of training range. And it remains fully parallelizable compared to existing RNN-based methods. Second, we devise a multi-task loss function that optimizes CoEvoGNN jointly on predicting node attributes and graph structure over time. This allows our framework to learn the *co-evolutionary* interactions between change of attributes and formation of links, and to use these valuable information to better forecast an unseen future graph snapshot. Besides, our framework can utilize any static graph neural algorithm for aggregating neighbor information along the structural axis. We developed and investigated three (but not limited to three) methods based on the proposed framework, named CoEvoGCN, CoEvoGAT, and CoEvoSAGE. We evaluate the performance of CoEvoGNNs methods on forecasting an entire future snapshot of co-authorship attributed graph and virtual currency graph. Experimental results demonstrate it can outperform competitive baselines by $+9.2\%$ of F1 score on link prediction, and by -49.1% of RMSE on attribute inference.

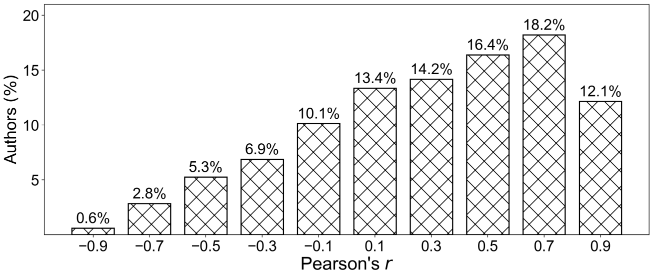


Fig. 2. The evolution of personal attributes (i.e., keyword change) and the evolution of graph structure (i.e., collaborator change) are highly correlated. When authors write more different keywords next year, they will have more different collaborators, and vice versa. More than 60% of the authors show higher-than-0.3 correlated co-evolution.

The main contributions are summarized as below:

- We propose a new framework for modeling the co-evolutionary patterns of node attributes and graph structure leveraging an arbitrary GNN as its underlying structural aggregator.
- We design a novel S-stack temporal self-attention architecture that can effectively distill and fuse influence from multiple previous graph snapshots with self-adapting importance.
- Extensive studies demonstrate our methods outperform strong baselines on forecasting an entire future snapshot of real-world graphs, and they are efficient and parallelizable.

Before we define the research problem and introduce our framework, we present statistical analysis on the long dependencies and evolutionary patterns in real graph data.

2 The Co-Evolution Phenomenon

The co-evolutionary process of node attributes and graph structure in real dynamic graphs is a fundamentally complex phenomenon and imposes great challenges for learning. *First, the node attributes and structure of a graph snapshot depend on the states of multiple previous graphs with an effect of time decay* [17]. Take a co-authorship network as an example: the formation of a collaboration link between two authors can be traced back to their previous co-authored event 2, or 3, or even 5 years ago. In Figure 1(a), we plot the distribution of two author nodes developing a future link at $t \in \{2008, 2009, 2010\}$ if they were linked at $t - \Delta$. In this case, repeating a link can be seen as realization of a link formation process. The proportion of these links are presented by the minimum interval Δ. As we can see, though a fair amount of the links occurred in the last year ($\Delta = 1$), more than 29% of new links can be traced back to previous years of $\Delta > 1$. In Figure 1(b), we plot another important mechanism of link formation – triad closure [18], which interprets closing a triad as forming a new link. It is evident that 46% links formed through this process fell in the range of $\Delta > 1$, though the number quickly drops at longer intervals. This intuitively indicates that earlier graphs before the most recent one contain valuable information for predicting the future graph state, and their relative importances should also be carefully considered.

Second, node attributes and graph structure are co-evolving and mutually influencing each other over time. For example, in a co-authorship network, forming a new link (i.e., a new collaboration) extends research scope and increases the impact of the authors. And, having new research topics, or having a higher value of h-index, in turn helps the authors to develop new collaborations [19]. Figure 2 shows the distribution of Pearson correlation between attribute and link evolutions. For every author, we calculate the Jaccard similarity of the author's keyword sets and that of his/her collaborators' sets between any two consecutive years. Then, we measure the correlation between these two similarity series over time. If an author changed his/her keywords dramatically in one year and his/her collaborators also changed significantly, the correlation value would be high. We found that more than 60% of the authors show higher-than-0.3 correlation. This mutually influencing characteristic between node attributes and graph structure requires both types of information to be used for training the model. Existing methods were not able to learn effective node embeddings for simultaneously forecasting node attributes and graph structure.

3 Research Problem

In this section, we formally define the research problem and briefly cover the preliminaries on static graph neural networks. Commonly used notations used throughout this paper and their descriptions are summarized in Table 1.

3.1 Problem definition

Traditionally, a static graph is represented as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. The node attribute matrix of G is denoted as $\mathbf{X} \in \mathbb{R}^{n \times r}$, where each row $\mathbf{x}_v$ describes the r-dimensional raw attribute vector of node v. However, real graphs evolve over time. The process of graph evolution manifests in two aspects mutually influencing each other: (1) the change of node attributes $\mathbf{X}^t$ across time steps $t = 0, 1, \ldots, T$; and, (2) the change of graph structure $G^t = (\mathcal{V}, \mathcal{E}^t)$ across time. For brevity, we use $\mathcal{V}$ to denote all unique nodes, i.e., $\mathcal{V} = \bigcup_{t=0}^{T} \mathcal{V}^t$, so the change in G^t is reflected as the change of $\mathcal{E}^t$. We define a sequence of dynamic graphs as:

Definition 1 (Dynamic Graph Sequence). A dynamic graph sequence across time steps from 0 to T contains consecutive snapshots $(G^0, \mathbf{X}^0), (G^1, \mathbf{X}^1), ..., (G^T, \mathbf{X}^T)$ of both the graph structure and node attributes. Each single snapshot $(G^t, \mathbf{X}^t)$ for $t = 0, 1, \ldots, T$ represents a transitional state of the graph during the evolution.

TABLE 1
Symbols and descriptions

Symbol	Description
G	a graph
$\mathcal{V}, \mathcal{E}$	a set of nodes, a set of edges
$\mathbf{X}, \mathbf{x}_v$	node attribute matrix, attribute vector of node v
$(G^t, \mathbf{X}^t)$	a snapshot of G's graph structure and its node attribute matrix $\mathbf{X}$ at time step t
$\mathcal{D}$	a dynamic graph sequence dataset
$\mathbf{h}_v^t$	latent embedding of node v at time step t
$\mathbf{H}^t$	matrix of all nodes embeddings of at time step t
$\mathbf{h}_v^{(l)}$	latent embedding of node v at structural depth l
$f_{static}^{\langle s \rangle}$	a static GNN method as CoEvoGNN's structural aggregator at temporal depth s
$\hat{\mathbf{h}}_v^{\langle s \rangle}$	node v's neighbor structural information embedding at temporal depth s
$a_v^{\langle s \rangle}$	node v's temporal self-attention weight at temporal depth s
S	temporal evolution span
$\mathbf{W}^{\langle s \rangle}$	weight matrix at temporal depth s
$\mathbf{\Gamma}$	temporal fusion matrix
$\mathbf{M}$	attribute transformation matrix
$\mathcal{J}_{\mathbf{X}^t}, \mathcal{J}_{G^t}$	attribute, structure evolutionary loss at time step t
$\mathcal{J}$	overall evolutionary loss
σ, $[\cdot\,;\,\cdot]$	a non-linear function, and concatenation operator

Then, we formally define the research problem as follows:

Problem: *Given* a dynamic graph sequence, i.e., $\mathcal{D} = \{(G^t, \mathbf{X}^t) \,|\, t = 0, 1, \ldots, T\}$, *learn* a mapping function $f(\mathcal{D}) : \mathcal{V} \times \{0, 1, \ldots, T\} \to \mathbb{R}^d$ that embeds each node $v \in \mathcal{V}$ into a d-dimensional (typically $d \ll r, |\mathcal{V}|$) representation vector $\mathbf{h}_v^t$ at each time step t that can preserve the co-evolution of node attributes and graph structure.

This dynamic graph representation learning problem would degenerate into the classic static graph setting when $T = 0$, where the input is a single pair of $(G, \mathbf{X})$. And, the result latent node embeddings matrix $\mathbf{H}$ contains both the information about the graph structure G and node attributes $\mathbf{X}$. For a non-trivial dynamic graph sequence with $T \geq 1$, node embeddings at different time steps, i.e., $\{\mathbf{H}^t \,|\, t = 0, 1, \ldots, T\}$, should not be completely independent from each other. Instead, each $\mathbf{H}^t$ should not only contain information about the current snapshot $(G^t, \mathbf{X}^t)$, but also summarize the co-evolution trend from the graphs in recent past into the near future. Specifically, we aim at learning $\mathbf{H}^t$ can be characterized by following properties:

- Revealing the historical co-evolution trend information of node attributes and graph structure in previous S graph snapshots $(G^{t-S}, \mathbf{X}^{t-S}), \ldots, (G^{t-1}, \mathbf{X}^{t-1})$.
- Being highly indicative about the developing co-evolution trend of node attributes and graph structure of the next S graph snapshots in the future $(G^{t+1}, \mathbf{X}^{t+1}), \ldots, (G^{t+S}, \mathbf{X}^{t+S})$.

Next, we briefly review representative static graph neural network methods, which will latter serve as building blocks of our framework for dynamic graph sequence representation learning.

3.2 Preliminaries

Static graph neural network models learn a function $f_{static}(G, \mathbf{X}) : \mathcal{V} \rightarrow \mathbb{R}^d$ that maps each node v into a d-dimensional latent embedding $\mathbf{h}_v$. The node embedding matrix $\mathbf{H} \in \mathbb{R}^{n \times d}$ incorporates both information about the node attribute matrix $\mathbf{X}$ and the graph structure G. Spectral methods like GCN [3] define f_{static} as:

$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{\Theta}^{(l)}), \tag{1}$$

where $\hat{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. $\mathbf{A}$ is the adjacent matrix of G, $\mathbf{D}$ is degree matrix, and $\mathbf{\Theta}^{(l)}$ is the model parameters of the l-th convolution layer. To avoid notational conflicts, we use superscript (l) in parentheses to denote the structural depth in a graph throughout this paper. Spatial methods like GraphSAGE [6] defines f_{static} as an aggregation operation applied on a locally sampled neighborhood of nodes:

$$\mathbf{h}^{(l+1)}_{\mathcal{N}(v)} = \text{AGG}^{(l+1)}(\{\mathbf{h}^{(l)}_u, \forall u \in \mathcal{N}(v)\}), \tag{2}$$

which summarizes attribute information from node v's sampled neighbors $\mathcal{N}(v)$. The aggregated neighbor information is then concatenated with the embedding of center node itself and gets further transformed, i.e., $\mathbf{h}^{(l+1)}_v = \sigma\left(\mathbf{W}^{(l+1)}[\mathbf{h}^{(l)}_v; \mathbf{h}^{(l+1)}_{\mathcal{N}(v)}]\right)$ where $\mathbf{W}^{(l+1)}$ contains the parameters at depth $l+1$. Obviously, these methods are designed for a single static graph and do not consider the evolutionary patterns of node attributes and graph structure over time. Conceptually, any f_{static} function blends a single snapshot $(G, \mathbf{X})$ into $\mathbf{H}$. So, directly applying static methods when the attributed graph G is actively evolving would fail to capture the valuable temporal trend in evolution.

For a dynamic graph sequence, like the node embeddings generated by static methods, $\mathbf{H}^t \in \mathbb{R}^{n \times d}$ should embed the node attribute information $\mathbf{X}^t$ and graph structure information G^t. In addition, and more importantly, $\mathbf{H}^t$ should embed the influence from multiple previous graph snapshots and be predictive on future graph snapshots. Here, the key questions are: (1) How to distill and fuse the influence of multiple previous snapshots into the current one? (2) How to effectively capture the co-evolutionary process of node attributes $\mathbf{X}^t$ and graph structure G^t that exhibits in $\mathcal{D}$ across time? To answer these questions, we propose a novel framework CoEvoGNN. It has a S-stack temporal self-attention architecture and employs a specifically designed multi-task objective for attribute inference and link prediction over time. Next, we present the algorithm of CoEvoGNN model for evolutionary node embedding generation and the evolutionary loss for training.

4 Proposed Framework

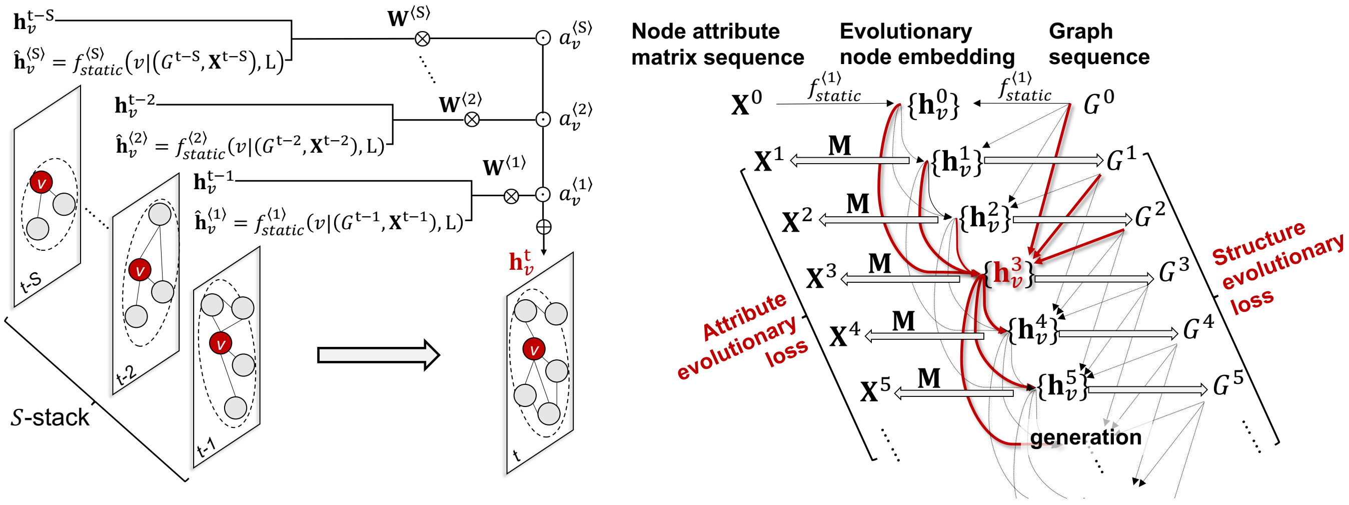
In this section, we present the evolutionary node embedding generation process of CoEvoGNN as illustrated in Figure 3 (a). The pseudocode of our proposed framework is given in Algorithm 1. CoEvoGNN is designed to capture the co-evolution pattern of node attributes and graph structure in dynamic graph sequence along the temporal axis.

Given a dynamic graph sequence $\{(G^t, \mathbf{X}^t) \,|\, t = 0, \ldots, T\}$, CoEvoGNN's weight matrices $\{\mathbf{W}^{\langle s \rangle} \,|\, s = 1, \ldots, S\}$ and its fusion matrix $\mathbf{\Gamma}$, the temporal evolution span S, and a set of static models $\{f^{\langle s \rangle}_{static} \,|\, s = 1, \ldots, S\}$, CoEvoGNN first generates the initial latent embedding of node from the leading graph snapshot $(G^0, \mathbf{X}^0)$ (Line 3 of Algo. 1). In practice, we can use an arbitrary static GNN algorithm (e.g., GCN [3], GAT [16] and GraphSAGE [6]) as $f^{\langle \cdot \rangle}_{static}$ functions. We will examine the choice of $f^{\langle \cdot \rangle}_{static}$ in Section 5. In particular, we concatenate the intermediate node embeddings at different structural depths together, i.e., $\mathbf{h}^{\langle \cdot \rangle}_v = f^{\langle \cdot \rangle}_{static}(v \,|\, (G, \mathbf{X}), L) \in \mathbb{R}^{dL \times 1}$, where d is the latent dimensions and L is the structural depth. This can allow CoEvoGNN to retain complete high-order neighbor structural information from $f^{\langle \cdot \rangle}_{static}$ across time [20], [21], and later determine the relative importance of previous graphs.

Algorithm 1: CoEvoGNN framework

```
Input : Dynamic graph sequence
        $\{(G^t, \mathbf{X}^t) \,|\, t = 0, \ldots, T\}$; parameter
        matrices $\{\mathbf{W}^{\langle s \rangle} \,|\, s = 1, \ldots, S\}$ and fusion
        matrix $\mathbf{\Gamma}$; temporal evolution span $S$; and,
        static graph neural models
        $\{f^{\langle s \rangle}_{static} \,|\, s = 1, \ldots, S\}$.
Output: Node latent embeddings $\mathbf{h}^t_v$, $v \in \mathcal{V}$ and
        $1 \leq t \leq T$.
1  for $v \in \mathcal{V}$ do
2      // Initialization
3      $\mathbf{h}^0_v \leftarrow f^{\langle 1 \rangle}_{static}(v \,|\, (G^0, \mathbf{X}^0), 1)$
4      for $t = 1, \ldots, T$ do
5          // Structural aggregations
6          Let $\hat{H}_v[1, \ldots, \min(t, S)]$ and
            $E_v[1, \ldots, \min(t, S)]$ be new arrays
7          for $s = 1, \ldots, \min(t, S)$ do
8              $\hat{\mathbf{h}}^{\langle s \rangle}_v \leftarrow f^{\langle s \rangle}_{static}(v \,|\, (G^{t-s}, \mathbf{X}^{t-s}), L)$
9              $e^{\langle s \rangle}_v \leftarrow (\mathbf{h}^{t-s}_v)^\top \cdot \mathbf{\Gamma} \cdot \hat{\mathbf{h}}^{\langle s \rangle}_v$
10             $\hat{H}_v[s] = \hat{\mathbf{h}}^{\langle s \rangle}_v$ and $E_v[s] = e^{\langle s \rangle}_v$
11         end
12         // Temporal self-attention
13         Let $A_v[1, \ldots, \min(t, S)]$ be a new array
14         for $s = 1, \ldots, \min(t, S)$ do
15             $a^{\langle s \rangle}_v \leftarrow \frac{\exp(E_v[s])}{\sum_{s'=1}^{\min(t,S)} \exp(E_v[s'])}$
16             $A_v[s] = a^{\langle s \rangle}_v$
17         end
18         // Fusion and normalization
19         $\mathbf{h}^t_v \leftarrow$
            $\sum_{s=1}^{\min(t,S)} A_v[s] \, \sigma\left(\mathbf{W}^{\langle s \rangle} \cdot \left[\mathbf{h}^{t-s}_v; \hat{H}_v[s]\right]\right)$
20         $\mathbf{h}^t_v \leftarrow \mathbf{h}^t_v / \|\mathbf{h}^t_v\|_2$
21     end
22 end
```

After initialization, CoEvoGNN generates latent node embeddings along time steps $t = 1, \ldots, T$ in a cascade mode. For node v at a specific time step t, CoEvoGNN extracts and merges its neighbor structural embeddings in the last S, or precisely $\min(t, S)$, snapshots with self-adapting importance (Line 4-19 in Algo. 1). The newly fused $\mathbf{h}^t_v$ gets l2 normalized and returned as the output node latent embedding (Line 20 in Algo. 1). Next, we introduce the core component of CoEvoGNN for automatically distilling and fusing influence from multiple previous graph snapshots.


$\mathbf{h}_v^{t-S}$
$\hat{\mathbf{h}}_v^{\langle S\rangle} = f_{static}^{\langle S\rangle}(v|(G^{t-S},\mathbf{X}^{t-S}),L)$
$\mathbf{W}^{\langle S\rangle}$
$a_v^{\langle S\rangle}$
$\mathbf{h}_v^{t-2}$
$\hat{\mathbf{h}}_v^{\langle 2\rangle} = f_{static}^{\langle 2\rangle}(v|(G^{t-2},\mathbf{X}^{t-2}),L)$
$\mathbf{W}^{\langle 2\rangle}$
$a_v^{\langle 2\rangle}$
$\mathbf{h}_v^{t-1}$
$\hat{\mathbf{h}}_v^{\langle 1\rangle} = f_{static}^{\langle 1\rangle}(v|(G^{t-1},\mathbf{X}^{t-1}),L)$
$\mathbf{W}^{\langle 1\rangle}$
$a_v^{\langle 1\rangle}$
$\mathbf{h}_v^{t}$
v
t-S
t-2
t-1
t
S-stack
Node attribute matrix sequence
Evolutionary node embedding
Graph sequence
$\mathbf{X}^0$
$f_{static}^{\langle 1\rangle}$
$\{\mathbf{h}_v^0\}$
$f_{static}^{\langle 1\rangle}$
G^0
$\mathbf{X}^1$
M
$\{\mathbf{h}_v^1\}$
G^1
$\mathbf{X}^2$
M
$\{\mathbf{h}_v^2\}$
G^2
$\mathbf{X}^3$
M
$\{\mathbf{h}_v^3\}$
G^3
$\mathbf{X}^4$
M
$\{\mathbf{h}_v^4\}$
G^4
$\mathbf{X}^5$
M
$\{\mathbf{h}_v^5\}$
G^5
Attribute evolutionary loss
Structure evolutionary loss
generation

(a) Evolutionary node embedding generation ($t \geq S$)

(b) Evolutionary attribute and structure losses (S = 3)

Fig. 3. Visual illustration of the framework of CoEvoGNN: (a) The evolutionary node embedding generation process dynamically fuses node latent representations from multiple previous graph into the current one by using the novel S-stack temporal self-attention architecture; and (b) The co-evolutionary loss function consists of attribute evolutionary loss and structure evolutionary loss for capturing the co-evolutions in graph sequence.

4.1 S-stack temporal self-attention

Equipping with static graph neural methods $f_{static}^{\langle\cdot\rangle}$ as its underlying aggregator, CoEvoGNN is able to distill structural information from each single time step independently. This means the resulting node embeddings $\mathbf{H}^t$ are solely determined by its corresponding graph snapshot $(G^t, \mathbf{X}^t)$, and all evolutionary dynamics of the graph are ignored. How can we effectively capture the co-evolution of node attributes and graph structure along the temporal axis? One straightforward way is to enforce the Markov property [22] and directly transform node embeddings from the previous time step $\mathbf{H}^{t-1}$ into the current one $\mathbf{H}^t$ [23]. But this oversimplified setting does not always hold in real cases. As an example: in an evolutionary co-authorship graph, authors collaborate in one year does not necessarily indicate they will collaborate in the next year; but authors could be more likely to collaborate if they have collaboration experience before [24]. Alternatively, we could assume node embeddings at each time $\mathbf{H}^t$ depend on all previous node embeddings $\mathbf{H}^0, \cdots, \mathbf{H}^{t-1}$, following a strict autoregressive paradigm [25]. Most related methods fall in this category and utilizes various RNN models to capture the dynamics of node embeddings [8], [10] or GNN parameters [12]. However, these models have difficulty in compressing long-range dependencies into hidden state [14], as well as severe scalability issues as they cannot be easily parallelized [15].

To this end, we design a novel *S-stack temporal self-attention* architecture (see Figure 3 (a)) for automatically distilling and fusing influence from multiple previous graph snapshots. Particularly, for node v at time step t, we first leverage static models $f_{static}^{\langle\cdot\rangle}$ to obtain its rich neighbor structural information $\hat{\mathbf{h}}_v^{\langle\cdot\rangle}$ (where $\langle\cdot\rangle$ indicates the temporal depth from the previous snapshot to the current one) for each one of the last S, or precisely $\min(t, S)$, snapshots (Line 4-8 of Algo. 1). Each one of these neighbor structural information embeddings $\hat{\mathbf{h}}_v^{\langle s\rangle} \in \mathbb{R}^{dL\times 1}$ is also processed into the pre-attention energy scalar $e_v^{\langle s\rangle}$ by feeding it into a bilinear mapping $\mathbf{\Gamma} \in \mathbb{R}^{d\times dL}$ along with the node latent embedding at the same time step $\mathbf{h}_v^{t-s} \in \mathbb{R}^{d\times 1}$ (Line 9 of Algo. 1). Next, node v's self-adapting weights $a_v^{\langle\cdot\rangle}$ for fusing previous influence are calculated from $e_v^{\langle\cdot\rangle}$ by taking softmax over them (Line 12-17 of Algo. 1). Then, for each one of the previous S-stack, the node latent embedding $\mathbf{h}_v^{t-s}$ and its neighbor structural embedding $\hat{H}_v[s] = \hat{\mathbf{h}}_v^{\langle s\rangle}$ are concatenated and transformed the through the weight matrices $\mathbf{W}^{\langle s\rangle} \in \mathbb{R}^{d\times(d+dL)}$ (Line 19 of Algo. 1). At last, the new node embedding $\mathbf{h}_v^t \in \mathbb{R}^{d\times 1}$ with self-attention on transformed previous latent and structural embeddings according to $A_v = a_v^{\langle 1\rangle}, \ldots, a_v^{\langle\min(t,S)\rangle}$ are returned.

At a high level, CoEvoGNN merges each node's latent embeddings and neighbor structural information embeddings for up to S previous time steps. This is different from solely relying on the most recent time step or compressing information from all previous time steps which can easily leads to unaffordable efficiency. On one hand, the temporal evolution span hyperparameter S controls a tradeoff between the model's expressive power of co-evolution pattern and its space complexity; on the other hand, it allows the adaptability for handling specific data or applications as increasing S brings diminishing marginal benefits in practice. We will discuss the choice of S in the experiments. Furthermore, the temporal self-attention mechanism on S-stack grants each node the flexibility for judging the relative importance of previous graph snapshots and dynamically fusing them into the current node latent embedding.

4.1.1 *Inferring future node embeddings*

The output of CoEvoGNN consists of a sequence of node latent embeddings $\mathbf{H}^t, t = 1, \cdots, T$, summarizing the training dynamic graph sequence. At inference phase, beyond the training range, CoEvoGNN generates an arbitrary number of node latent embeddings at future time steps (e.g., $\mathbf{H}^{T+1}$, $\mathbf{H}^{T+2}$, ...). These future node embeddings directly reflect CoEvoGNN's forecasting capability on the co-evolutoin trend of node attributes and graph structure learned from the observed graph snapshots. Forecasting into far future would be really challenging. In this paper, we only focus on predicting node embeddings of the next time step $(T+1)$ after the training evolutionary graph snapshots and leave forecasting multiple time steps as future work. Next, we introduce the multi-task co-evolutionary objective of CoEvoGNN as well as the training procedure.

4.2 Training on multi-task co-evolutionary loss

In this section, we present the approach for training the CoEvoGNN model. The overall co-evolutionary loss function is defined in Eqn. (3) and the training procedure of CoEvoGNN is presented in Algorithm 2.

To learn the CoEvoGNN model on a dynamic graph sequence for forecasting into future, we carefully devise a multi-task loss function supervising generated node latent representations $\mathbf{h}_v^t$ over training time steps $t = 1, \ldots, T$. In a forward pass, we adopt the mini-batch training mode for stochastic optimization [26]. For each randomly sampled mini-batch of nodes $\mathcal{V}' \subset \mathcal{V}$, the result embeddings gets evaluated by the overall loss. During backpropagation, we use stochastic gradient descent with ADAM updating rule [27] to update the set of weight matrices $\{\mathbf{W}^{\langle s \rangle} \mid s = 1, \ldots, S\}$, the fusion matrix $\mathbf{\Gamma}$, and attribute transformation matrix $\mathbf{M}$ (see Section 4.2.1), which parameterizes the proposed CoEvoGNN model.

$$\min_{\mathbf{h}_v^t, v \in \mathcal{V}', t=1,\ldots,T} \mathcal{J} = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}'} \alpha \mathcal{J}_{\mathbf{X}^t}(\mathbf{h}_v^t) + (1-\alpha)\mathcal{J}_{G^t}(\mathbf{h}_v^t). \tag{3}$$

This multi-task evolutionary objective is mainly composed of two terms: the attribute evolutionary loss $\mathcal{J}_{\mathbf{X}^t}$, and the structure evolutionary loss $\mathcal{J}_{G^t}$. A mixture hyperparameter α is used to balance the magnitude of these two terms. Particularly, the attribute loss $\mathcal{J}_{\mathbf{X}^t}$ evaluates the quality of generated embeddings on node attribute inference. It transforms node embeddings back into the raw attribute space and encourages the resulting vector to approximate the node's true attribute distribution. And, the graph structure loss $\mathcal{J}_{G^t}$ evaluates the quality of generated node embeddings on graph link prediction. It encourages nodes linked by observed edges to have similar latent representations and pushes disconnected nodes to have distinct representations. Figure 3 (b) visualizes an example of this multi-task evolutionary loss ($S = 3$). Next, we provide more details on these two evolutionary objectives.

4.2.1 Attribute evolutionary loss for attribute inference

The attribute evolutionary loss $\mathcal{J}_{\mathbf{X}^t}$ is defined as below:

$$\mathcal{J}_{\mathbf{X}^t}(\mathbf{h}_v^t) = \|\sigma(\mathbf{M} \cdot \mathbf{h}_v^t) - \mathbf{x}_v^t\|_F^2, \tag{4}$$

where $\mathbf{M}$ is the attribute transformation matrix and σ is non-linear function such as ReLU or sigmoid. Given a node latent embedding $\mathbf{h}_v^t \in \mathbb{R}^{d \times 1}$, the attribute transformation matrix $\mathbf{M} \in \mathbb{R}^{r \times d}$ is used for mapping $\mathbf{h}_v^t$ back into the r-dim raw attribute space. Node v's remapped attribute inference vector $\sigma(\mathbf{M} \cdot \mathbf{h}_v^t) \in \mathbb{R}^{r \times 1}$ is then compared against the true node attribute vector $\mathbf{x}_v^t$ by measuring the L2 distance. Note that parameter matrix $\mathbf{M}$, which is irrelevant to T or S, describes the transformation from latent embedding space back to raw attribute space across time, also gets updated with back propagation. This can easily be generalized to other choices such as MLP at the cost of additional model parameters. In practice we find $\mathbf{M}$ generally suffices given high quality node latent embeddings. Additionally, in case of node classification that $\mathbf{x}_v^t$ is one-hot vector, we can set σ as softmax function and compute the cross entropy.

Algorithm 2: Training procedure of CoEvoGNN

```
Initialize model parameters $\{\mathbf{W}^{\langle s \rangle} \mid s = 1, \ldots, S\}$, $\mathbf{\Gamma}$, and $\mathbf{M}$
repeat
    Sample minibatch of nodes $\mathcal{V}'$ from all nodes $\mathcal{V}$
    // Generate evolutionary embeddings
    $\mathbf{H}^1, \ldots, \mathbf{H}^T \leftarrow$ CoEvoGNN($\mathcal{V}'$)    ▷ see Algo. 1
    // Compute evolutionary losses
    $\mathcal{J}_{\mathbf{X}^1}, \ldots, \mathcal{J}_{\mathbf{X}^T} \leftarrow$ Compute the attribute evolutionary loss    ▷ see Eqn. (4)
    $\mathcal{J}_{G^1}, \ldots, \mathcal{J}_{G^T} \leftarrow$ Compute the structure evolutionary loss    ▷ see Eqn. (5)
    $\mathcal{J} \leftarrow$ Compute overall loss    ▷ see Eqn. (3)
    // Update parameters
    $\mathbf{W}^{\langle \cdot \rangle} \xleftarrow{+} -\nabla_{\mathbf{W}^{\langle \cdot \rangle}}(\mathcal{J})$
    $\mathbf{\Gamma} \xleftarrow{+} -\nabla_{\mathbf{\Gamma}}(\mathcal{J})$
    $\mathbf{M} \xleftarrow{+} -\nabla_{\mathbf{M}}(\mathcal{J})$
until finish;
```

4.2.2 Structure evolutionary loss for link prediction

The structure evolutionary loss $\mathcal{J}_{G^t}$ is defined as below:

$$\begin{aligned} \mathcal{J}_{G^t}(\mathbf{h}_v^t) \quad = \quad & -\log\left(\sigma\left((\mathbf{h}_v^t)^\top \cdot \mathbf{h}_u^t\right)\right) \\ & -Q \cdot \mathbb{E}_{u' \sim P_n(v)} \log\left(\sigma\left(-(\mathbf{h}_v^t)^\top \cdot \mathbf{h}_{u'}^t\right)\right), \end{aligned} \tag{5}$$

where node u is one of the 1st-order neighbors of node v. This can be relaxed to that node u co-occurs near node v on a fixed-length random walk. Node u' is a negative sample node, i.e., disconnected node with v, drawn according to the negative sampling distribution $P_n(v)$. Q is the number of negative samples and σ is the non-linear function. Intuitively, Eqn. (5) pulls similar nodes closer and pushes dissimilar nodes away in the latent space. Taken together with Eqn. (4), the multi-task evolutionary loss function (Eqn. (3)) captures the co-evolution patterns of node attributes and graph structure in a dynamic graph sequence over time.

4.3 Complexity Analysis

Assuming the per-batch time complexity of CoEvoGNN's underlying static methods $f_{static}^{\langle \cdot \rangle}$ is $\mathcal{O}\left(\Pi_{l=1}^{L} s_l\right)$ in principle [6] (where L is the structural depth and s_l is the neighbor sampling size at the l-th layer) and they can be parallelized in the S-stack temporal self-attention architecture, the CoEvoGNN's per-batch time complexity is $\mathcal{O}\left(T\,\Pi_{l=1}^{L} s_l\right)$. The computation cost only increases linearly with training range T and is regardless of the temporal evolution span S.

5 Experiments

In this section, we evaluate the effectiveness of CoEvoGNN on two forecasting tasks: (1) node attribute prediction, and (2) graph link prediction. Forecasting an *entire* future snapshot, instead of completing a partially seen graph, requires the understanding of evolutionary patterns. In all experiments, we test on predicting the *next graph snapshot*.

5.1 Datasets

We used 4 datasets from two type of evolutionary graphs.

Evolutionary co-authorship graph. We built a sequence of yearly co-authorship graphs by collecting $226,611$ papers from 2001 to 2010 in computer science from Microsoft Academic Graph [2]. Authors were ranked by their number of papers. The top $2,000$ and $10,000$ were used to make two datasets denoted by $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$. The venues and the paper title's words were used as node attributes after filtering out infrequent ones. As a result, we have 316 venues and $3,549$ words in $\mathcal{D}_{\text{AU}}^{2K}$; and 448 venues, $6,442$ words in $\mathcal{D}_{\text{AU}}^{10K}$. The detailed statistics on each one of the graph snapshots in $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$ are presented in Table 2.

Evolutionary virtual currency graph. We used 2 benchmark datasets Bitcoin-OTC and Bitcoin-Alpha of Bitcoin transaction networks [28] denoted by $\mathcal{D}_{\text{BC}}^{\text{otc}}$ and $\mathcal{D}_{\text{BC}}^{\text{alp}}$. These are two who-trusts-whom network of Bitcoin users trading on two major platforms[1]. To earn reputation and prevent transactions with fraudulent and risky users, members in the graph rate other members in a scale of -10 (total distrust) to +10 (total trust) in steps of 1. We followed the treatments described in [12] to form two sequences of graphs with 138, and 136, time steps, respectively (each for about 2 weeks); and, used node in/out degree as input node features. For fair comparison, we also adopted the strategy of maintaining a window of size of 10 time steps for the existence of an edge in the graph sequence since its initial creation to alleviate the graphs' over-sparsity problem.

TABLE 2
Evolutionary co-authorship datasets over 10 years.

	$\mathcal{D}_{\text{AU}}^{2K}$				$\mathcal{D}_{\text{AU}}^{10K}$			
Year	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	#V.	#W.	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	#V.	#W.
2001	874	1633	316	3549	4325	8080	448	6442
2002	1041	2390	251	2764	5235	11330	364	5253
2003	1211	3049	293	2945	6013	14632	414	5411
2004	1344	3847	262	3125	6873	18050	375	5702
2005	1405	4187	293	3172	7328	20752	424	5757
2006	1425	4182	266	3178	7582	21592	389	5826
2007	1467	4563	292	3245	7659	22690	420	5921
2008	1482	4705	267	3282	7707	22705	396	5937
2009	1452	4689	283	3283	7386	21924	412	5934
2010	1406	3907	265	3246	7146	19992	382	5888
Total	1928		316	3549	9854		448	6442

5.2 Experimental settings

Baseline methods: We compare CoEvoGNN's variants using different representative static methods against a spectrum of dynamic graph neural methods:

- GCN [3], GAT [16] and GraphSAGE [6]: We incorporate each one of these static method as CoEvoGNN's underlying operation and denote them as CoEvoGCN, CoEvoGAT, and CoEvoSAGE, respectively. We also directly compare against these static methods taking different merged graphs as input: (i) the most recent graph snapshot $(G^{t_{test}-1}, \mathbf{X}^{t_{test}-1})$, (ii) K most recent graph snapshots $(G^{t_{test}-1}, \mathbf{X}^{t_{test}-1}), \dots, (G^{t_{test}-K}, \mathbf{X}^{t_{test}-K})$, and (iii) all the graph snapshots before t_{test}. Then, each model was applied for inferring node embeddings at t_{test}.
- DynamicTriad [29]: This dynamic network embedding method only models the evolutionary pattern of graph structure and cannot handle node attributes. All graphs $G^{t_{test}-1}, G^{t_{test}-2}, \dots, G^0$ before t_{test} are fed for training, and we focus on the task of future graph link prediction on $G^{t_{test}}$.
- DySAT [30]: The treatment is similar to DynamicTriad except that we use $\mathbf{X}^0$ for initializing the input node embeddings.
- DCRNN [31]: For training, we use all node attributes $\mathbf{X}^{t_{test}-1}, \mathbf{X}^{t_{test}-2}, \dots, \mathbf{X}^0$ and the most recent graph $G^{t_{test}-1}$. The final prediction matrix outputted at t_{test} is directly used for future node attribute prediction, and the node embeddings outputted by the diffusion convolutional layer are used for future graph link prediction.
- STGCN [32]: The treatment is similar to DCRNN except that the node embeddings outputted by the last spatio-temporal convolutional block are used for future graph link prediction.
- EvolveGCN [12]: The input includes all graph snapshots $(G^{t_{test}-1}, \mathbf{X}^{t_{test}-1}), \dots, (G^0, \mathbf{X}^0)$ before t_{test}. We use the link prediction loss for training the model, and use the node embeddings outputted by the evolving graph convolution unit for future graph link prediction.

For GraphSAGE and EvolveGCN, we compute their attribute inference matrix as $\mathbf{H}^{t_{test}} \cdot (\mathbf{H}^{t_{train}\,\mathrm{T}} \cdot \mathbf{X}^{t_{train}})$. The last node embeddings matrix $\mathbf{H}^{t_{test}-1}$ from training graph snapshots and the last training attribute matrix $\mathbf{X}^{t_{test}-1}$ are treated as $\mathbf{H}^{t_{train}}$ and $\mathbf{X}^{t_{train}}$ for EvolveGCN. The core hyper-parameters of all baselines are set following the recommendations from their inventors or otherwise selected using the grid search strategy: β_0 and β_1 of DynamicTriad are found in $\{0.01, 0.1, 1, 10\}$; w_n of DySAT is found in $\{0.01, 0.1, 1\}$; the number of recurrent layers for DCRNN and the number of spatio-temporal convolutional blocks for STGCN are found in $\{1, 2\}$. All deep models are trained by a maximum of 100 epochs. The embedding size is 256 for $\mathcal{D}_{\text{AU}}^{10K}$ and 512 in other cases.

Evaluation metrics: For node attribute prediction, we use Mean Average Error (MAE) and Root Mean Squared Error (RMSE); for link prediction, we use Area Under the precision-recall Curve (AUC), F1 measure, and Precision@$50, 100, 200$. Because the raw attributes were very sparse in $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$, for all methods, we subsample the same number of close-to-zero predictions as the number of bigger-than-1 predictions during test to make it comparable.

[1] https://www.bitcoin-otc.com/ and https://btc-alpha.com/

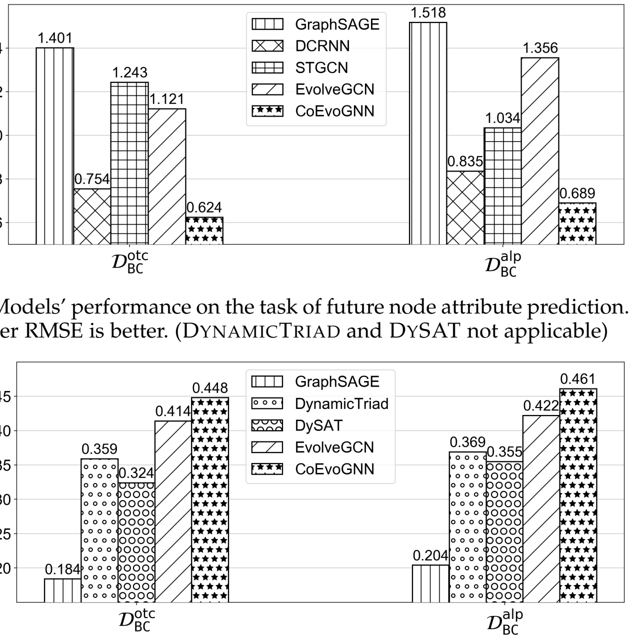
TABLE 3
On co-authorship graph sequence, CoEvoGNN outperforms baselines on forecasting node attributes of $\mathbf{X}^{2010}$ and graph structure G^{2010}.

	$\mathcal{D}_{\text{AU}}^{2K}$					$\mathcal{D}_{\text{AU}}^{10K}$				
	Attributes $\mathbf{X}^{2010}$		Links in G^{2010}			Attributes $\mathbf{X}^{2010}$		Links in G^{2010}		
Method	MAE	RMSE	AUC	F1	P@50, 100, 200	MAE	RMSE	AUC	F1	P@50, 100, 200
GCN [3]	0.649 ±0.018	1.297 ±0.029	0.082 ±0.008	0.196 ±0.011	0.34, 0.42, 0.36 ±0.02, 0.03, 0.05	0.742 ±0.020	1.566 ±0.037	0.034 ±0.004	0.071 ±0.004	0.30, 0.40, 0.34 ±0.03, 0.03, 0.04
GAT [16]	0.658 ±0.021	1.342 ±0.036	0.075 ±0.012	0.192 ±0.013	0.34, 0.36, 0.36 ±0.03, 0.05, 0.04	0.758 ±0.022	1.628 ±0.040	0.028 ±0.003	0.053 ±0.002	0.32, 0.30, 0.32 ±0.02, 0.02, 0.04
GraphSAGE [6]	0.643 ±0.014	1.265 ±0.021	0.084 ±0.007	0.201 ±0.011	0.38, 0.44, 0.41 ±0.02, 0.03, 0.02	0.729 ±0.019	1.438 ±0.025	0.039 ±0.004	0.078 ±0.003	0.36, 0.40, 0.42 ±0.03, 0.04, 0.02
DynamicTriad [29]	N/A -	N/A -	0.112 ±0.014	0.241 ±0.017	0.76, 0.62, 0.60 ±0.03, 0.02, 0.03	N/A -	N/A -	0.058 ±0.007	0.147 ±0.012	0.60, 0.59, 0.57 ±0.05, 0.06, 0.05
DySAT [30]	N/A -	N/A -	0.120 ±0.012	0.222 ±0.020	0.54, 0.46, 0.38 ±0.06, 0.07, 0.06	N/A -	N/A -	0.036 ±0.005	0.127 ±0.011	0.48, 0.43, 0.36 ±0.06, 0.06, 0.05
DCRNN [31]	0.458 ±0.012	0.960 ±0.042	0.019 ±0.002	0.073 ±0.009	0.12, 0.10, 0.10 ±0.01, 0.00, 0.01	0.423 ±0.008	0.853 ±0.023	0.006 ±0.000	0.027 ±0.001	0.09, 0.06, 0.03 ±0.01, 0.00, 0.00
STGCN [32]	0.478 ±0.010	1.127 ±0.029	0.006 ±0.001	0.027 ±0.002	0.04, 0.02, 0.04 ±0.00, 0.00, 0.01	0.567 ±0.015	1.589 ±0.040	0.001 ±0.000	0.007 ±0.000	0.04, 0.04, 0.02 ±0.01, 0.00, 0.00
EvolveGCN [12]	0.684 ±0.015	1.279 ±0.025	0.133 ±0.009	0.256 ±0.014	0.78, **0.80**, 0.67 ±0.07, 0.06, 0.05	0.768 ±0.022	1.603 ±0.039	0.069 ±0.004	0.161 ±0.021	0.69, 0.74, **0.59** ±0.06, 0.07, 0.06
CoEvoGCN	0.452 ±0.007	0.944 ±0.014	0.147 ±0.007	0.269 ±0.009	**0.82**, 0.76, 0.69 ±0.02, 0.03, 0.04	0.414 ±0.009	0.831 ±0.021	0.076 ±0.008	0.167 ±0.008	0.78, **0.76**, 0.54 ±0.04, 0.03, 0.04
CoEvoGAT	0.453 ±0.008	0.946 ±0.012	0.143 ±0.006	0.271 ±0.010	0.78, 0.74, 0.66 ±0.02, 0.03, 0.03	0.415 ±0.009	0.831 ±0.020	0.075 ±0.010	0.167 ±0.009	0.78, **0.76**, 0.54 ±0.03, 0.04, 0.05
CoEvoSAGE	**0.449** ±0.005	**0.938** ±0.008	**0.151** ±0.004	**0.274** ±0.005	**0.82**, **0.80**, **0.72** ±0.02, 0.01, 0.02	**0.410** ±0.004	**0.828** ±0.007	**0.079** ±0.005	**0.170** ±0.004	**0.80**, **0.76**, 0.58 ±0.02, 0.02, 0.01

5.3 Overall performance

Table 3 presents results on co-authorship graphs $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$. We report the performance of static methods GCN, GAT and GraphSAGE trained using all historical graph snapshots. But simply merging and feeding all previous graph snapshots into a static model loses the co-evolutionary patterns and thus underperforms almost all dynamic methods. It verifies that static methods cannot accurately forecast node attributes and graph structure. Three variants of CoEvoGNNs perform similar to each other; CoEvoSAGE makes slightly lower RMSE values and higher F1 values on both datasets. **Without causing ambiguity, we refer CoEvoSAGE as CoEvoGNN for comparison in this section.** Figure 4 presents the results on evolutionary virtual currency graphs $\mathcal{D}_{\text{BC}}^{\text{otc}}$ and $\mathcal{D}_{\text{BC}}^{\text{alp}}$.

Both dynamic network embedding methods DynamicTriad and DySAT give comparable performance to CoEvoGNN on the task of future graph link prediction. And, DynamicTriad can perform slightly better than DySAT. However, these two models only consider the dynamics of evolving graph structure instead of capturing the co-evolution of node attributes and graph structure. The "N/A" in Table 3 means they are not applicable, and they are also excluded from Figure 4(a). In contrast, by fusing influence from multiple previous states, CoEvoGNN can give F1 scores of 0.274 and 0.170 on two of the co-authorship datasets $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$, which is $+13.7\%$ and $+15.6\%$ relatively over DynamicTriad. The similar trend on $\mathcal{D}_{\text{BC}}^{\text{otc}}$ and $\mathcal{D}_{\text{BC}}^{\text{alp}}$ can be observed in Figure 4(b). This tells considering node attribute evolution is beneficial for modeling the change of graph structure as they are mutually influencing each other. They should be jointly modeled as a holistic co-evolutionary pattern.



(a) Models' performance on the task of future node attribute prediction. Lower RMSE is better. (DynamicTriad and DySAT not applicable)

(b) Models' performance on the task of future graph link prediction. Higher F1 is better. (DCRNN and STGCN excluded)

Fig. 4. CoEvoGNN outperforms baseline methods on forecasting an entire future snapshot of virtual currency graph.

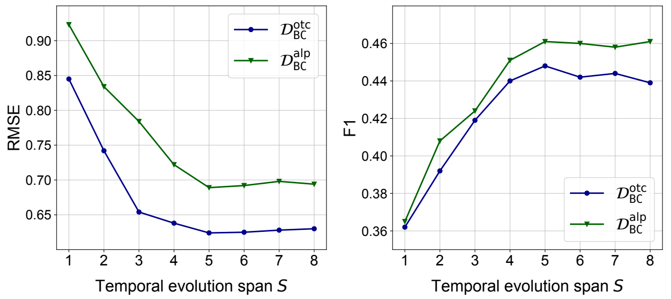
For spatiotemporal forecasting methods DCRNN and STGCN, they are designed for modeling the change of node attributes assuming the graph structure remains static. DCRNN outperforms all other baseline methods on the task of future node attribute prediction. It can score RMSEs of 0.960 and 0.853 on $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$ (0.754 and 0.835 on

TABLE 4
Forecasting node attributes and links in $\mathcal{D}^{2K}_{\text{AU}}$ and $\mathcal{D}^{10K}_{\text{AU}}$ with different values of temporal evolution span S.

	$\mathcal{D}^{2K}_{\text{AU}}$		$\mathcal{D}^{10K}_{\text{AU}}$	
	Attributes $\mathbf{X}^{2010}$	Links in G^{2010}	Attributes $\mathbf{X}^{2010}$	Links in G^{2010}
Depth	RMSE	F1	RMSE	F1
$S = 1$	0.988 ±0.033	0.236 ±0.015	0.897 ±0.031	0.149 ±0.012
$S = 2$	0.952 ±0.015	0.259 ±0.011	0.863 ±0.019	0.162 ±0.007
$S = 3$	**0.938** ±0.008	**0.274** ±0.005	0.832 ±0.009	**0.170** ±0.004
$S = 4$	0.940 ±0.007	0.270 ±0.005	**0.828** ±0.007	**0.170** ±0.004
$S = 5$	0.941 ±0.008	0.269 ±0.004	0.829 ±0.006	0.168 ±0.004
$S = 6$	0.941 ±0.010	0.271 ±0.005	0.830 ±0.007	0.167 ±0.006
$S = 7$	0.945 ±0.012	0.266 ±0.007	0.833 ±0.009	0.164 ±0.006
$S = 8$	0.951 ±0.014	0.262 ±0.009	0.840 ±0.011	0.160 ±0.010

$\mathcal{D}^{\text{otc}}_{\text{BC}}$ and $\mathcal{D}^{\text{alp}}_{\text{BC}}$). But DCRNN cannot produce acceptable performance on the task of future graph link prediction (scoring F1s of 0.073 and 0.027 on $\mathcal{D}^{2K}_{\text{AU}}$ and $\mathcal{D}^{10K}_{\text{AU}}$ and being excluded from Figure 4(b) because of low performance). The proposed CoEvoGNN is able to score RMSEs of 0.938 and 0.828 on $\mathcal{D}^{2K}_{\text{AU}}$ and $\mathcal{D}^{10K}_{\text{AU}}$ (-2.3% and -2.9% relatively over DCRNN); and, at the same time, perform much better on the task of future graph link prediction. This again demonstrates the advantage of CoEvoGNN by modeling the co-evolutionary pattern of node attributes and graph structure as they are mutually influencing each other.

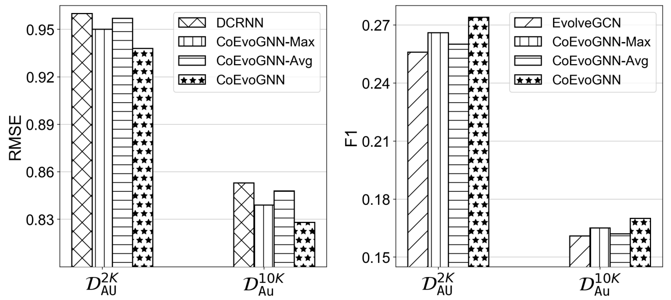
The most competitive baseline EvolveGCN achieves the best performance for predicting future graph links among all others. It scores F1s of 0.256 and 0.161 on $\mathcal{D}^{2K}_{\text{AU}}$ and $\mathcal{D}^{10K}_{\text{AU}}$ (0.414 and 0.422 on $\mathcal{D}^{\text{otc}}_{\text{BC}}$ and $\mathcal{D}^{\text{alp}}_{\text{BC}}$). Although its input also includes all historical graph snapshots, one fundamental difference between EvolveGCN and our CoEvoGNN is that EvolveGCN assumes the underlying force driving the graph evolution only comes from the changes in graph structure. It can be trained under its node classification mode but that requires the class information for each node at each time step which is commonly unavailable. In either way, EvolveGCN is unaware of the co-evolution process between node attributes and graph structure. So, EvolveGCN can only generate future node attribute predictions of similar quality as the static model GraphSAGE. As a result, CoEvoGNN outperforms it by -26.6% and -48.3% in terms of RMSE on $\mathcal{D}^{2K}_{\text{AU}}$ and $\mathcal{D}^{10K}_{\text{AU}}$ (-44.3% and -49.1% on $\mathcal{D}^{\text{otc}}_{\text{BC}}$ and $\mathcal{D}^{\text{alp}}_{\text{BC}}$); and CoEvoGNN scores $+7.0\%$ and $+5.6\%$ relatively higher F1 on $\mathcal{D}^{2K}_{\text{AU}}$ and $\mathcal{D}^{10K}_{\text{AU}}$ ($+8.2\%$ and $+9.2\%$ on $\mathcal{D}^{\text{otc}}_{\text{BC}}$ and $\mathcal{D}^{\text{alp}}_{\text{BC}}$). We also note that all methods have slightly larger standard deviation values of error on $\mathcal{D}^{10K}_{\text{AU}}$ compared with $\mathcal{D}^{2K}_{\text{AU}}$ because of the increased sparsity. And, CoEvoGNN can give more stable performance compared with baselines across datasets.



(a) Node attribute prediction. Lower RMSE value is better.

(b) Graph link prediction. Higher F1 indicates better performance.

Fig. 5. Forecasting node attributes and links in bitcoin graphs with different values of temporal evolution span S.



(a) Node attribute prediction. Lower RMSE value is better.

(b) Graph link prediction. Higher F1 indicates better performance.

Fig. 6. Standard CoEvoGNN with self-attention performs better than Max or Avg aggregators.

5.4 Effect of S-stack temporal self-attention

To understand CoEvoGNN's advantage, we further examine the effectiveness of the S-stack temporal self-attention architecture which is the core component for distilling and fusing influence from multiple previous graph snapshots. In particular, we examine different choices of the temporal evolution span S, and conduct ablation studies to verify the contribution of the temporal self-attention mechanism.

5.4.1 *Choice of temporal evolution span S*

CoEvoGNN is designed to fuse the influence from multiple previous graph snapshots, and we set the temporal evolution span S as a model hyperparameter controlling how many previous snapshots to be fused at each time step. This design grants the freedom of balancing between efficiency and model capability to users. On one hand, setting $S = 1$ enforces CoEvoGNN to follow the Markov property [22] and generate new node embedding relying only on the previous one. This would limit CoEvoGNN's expressive power on the co-evolution pattern since each time step is being modeled by directly transforming the previous one. On the other hand, setting S to a large value brings in additional computational burden since real evolutionary graph exhibits a time decay effect (see Section 2). We test different settings of S from the value of 1 up to 8 and present the results in Table 4 and Figure 5.

For evolutionary co-authorship graphs $\mathcal{D}^{2K}_{\text{AU}}$ and $\mathcal{D}^{10K}_{\text{AU}}$, increasing the value of S from 1 up to 3 can make CoEvoGNN achieving better performance on both of the two tasks: the RMSE decreases from 0.988 to 0.938 on $\mathcal{D}^{2K}_{\text{AU}}$ (from

TABLE 5
Performance of CoEvoGNN on evolutionary co-authorship graphs $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$ with different of training length T. Training over time for more years performs better on both tasks.

	$\mathcal{D}_{\text{AU}}^{2K}$		$\mathcal{D}_{\text{AU}}^{10K}$	
	Attributes $\mathbf{X}^{2010}$	Links in G^{2010}	Attributes $\mathbf{X}^{2010}$	Links in G^{2010}
Training	RMSE	F1	RMSE	F1
$01-09$	0.938 ±0.008	**0.274** ±0.005	**0.828** ±0.007	**0.170** ±0.004
$02-09$	**0.937** ±0.010	0.266 ±0.007	0.845 ±0.010	0.158 ±0.06
$03-09$	0.943 ±0.013	0.258 ±0.010	0.846 ±0.015	0.145 ±0.012
$04-09$	0.954 ±0.019	0.249 ±0.014	0.860 ±0.022	0.138 ±0.020
$05-09$	0.958 ±0.023	0.230 ±0.017	0.883 ±0.024	0.121 ±0.024
$06-09$	0.961 ±0.030	0.211 ±0.026	0.896 ±0.035	0.105 ±0.038
$07-09$	0.972 ±0.032	0.182 ±0.031	0.912 ±0.037	0.087 ±0.042
$08-09$	1.054 ±0.041	0.169 ±0.038	0.945 ±0.039	0.078 ±0.042

0.897 to 0.828 on $\mathcal{D}_{\text{AU}}^{10K}$) for future node attribute prediction; and the F1 improves from 0.236 to 0.274 on $\mathcal{D}_{\text{AU}}^{2K}$ (from 0.149 to 0.170 on $\mathcal{D}_{\text{AU}}^{10K}$) for future graph link prediction. After that, further increasing S brings little improvement and the performance slightly drops at $S = 8$ probably due to too much noise brought in from farther graphs. The similar trend can also be observed on evolutionary virtual currency graphs $\mathcal{D}_{\text{BC}}^{\text{otc}}$ and $\mathcal{D}_{\text{BC}}^{\text{alp}}$ and the diminishing marginal benefits of increasing S are noticeable. But different from co-authorship graphs, the elbow point appears at $S = 5$. This can be explained by the fact that these two virtual currency graphs contains longer time steps (138 and 136 vs. 10 of $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$) and each graph is also sparser. So, CoEvoGNN needs more historical information in generating node embeddings. In practice, we suggest using a grid search strategy to find the optimal S in the range of $[2, 8]$.

5.4.2 Contribution of temporal self-attention

On graph level, CoEvoGNN fuses influence from up to S previous time steps; and, on the node level, it is natural to assume each node have different dependency strengths to its previous states. Another critical aspect of CoEvoGNN is the design of temporal-self-attention (see Line 9 and 12-17 of Algo. 1) allowing each node to independently determine the relative importance of previous S snapshots when generating its new embedding. To validate the effectiveness of this design, we build two variants of the model: (1) CoEvoGNN-Max that takes the max pooling operation on node's previous states; and, (2) CoEvoGNN-Avg that takes mean vector of node's previous states (see Line 19 of Algo. 1). We compare the improvements of CoEvoGNN and its variants against DCRNN for the task of node attribute prediction, and against EvolveGCN for future graph link prediction. The results are represented in Figure 6.

It is evident that CoEvoGNN significantly outperform the other two variants and producing much lower RMSE values on the task of future node attribute prediction and higher F1 values on the task of graph link prediction. We can also observe that CoEvoGNN-Max performs slightly better than CoEvoGNN-Avg probably because it is able to extract the most salient latent embeddings from previous states instead of averaging the information. By automatically learning the relative importances of pervious graphs and dynamically adapting to those weights, CoEvoGNN is able to produce lower RMSE (-1.3% on $\mathcal{D}_{\text{AU}}^{2K}$ and -0.9% on $\mathcal{D}_{\text{AU}}^{10K}$ relatively over CoEvoGNN-Max) for node attribute prediction and higher F1 ($+1.9\%$ on $\mathcal{D}_{\text{AU}}^{2K}$ and $+1.3\%$ on $\mathcal{D}_{\text{AU}}^{10K}$ relatively over CoEvoGNN-Max) for graph link prediction. The similar findings can also be observed on evolutionary virtual currency graphs $\mathcal{D}_{\text{BC}}^{\text{otc}}$ and $\mathcal{D}_{\text{BC}}^{\text{alp}}$ but are not presented here due to space limit. These observations confirm that the design of temporal self-attention over node's previous states is successful in dynamically fusing influence from multiple previous states. Taken together with a reasonable choice of the temporal evolution span S, CoEvoGNN is able to model the co-evolution of node attributes and graph structure with self-adapting importance over time.

5.5 Effect of training length T

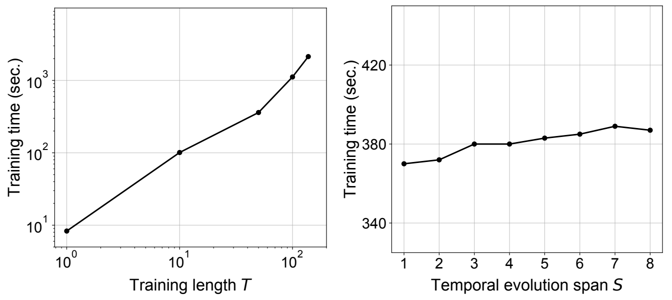
Since CoEvoGNN generates node embeddings along the training time steps, we examine the effect of the training length T on the performance of CoEvoGNN. Specifically, we vary the number of graphs in the training sequence from all historical graphs to the only one before the test graph. Table 5 presents the results on two of the evolutionary co-authorship graphs $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$.

On dataset $\mathcal{D}_{\text{AU}}^{2K}$, CoEvoGNN achieves the best performance for predicting graph links in G^{2010} when using all historical graph snapshots from 2001 to 2009; and has comparable good performance for predicting node attribute in $\mathbf{X}^{2010}$ if being fed graph snapshots starting from 2002. The trend is consistent on the $\mathcal{D}_{\text{AU}}^{10K}$ dataset: CoEvoGNN achieves its best performance for both tasks when trained using all historical graph snapshots. This indicates that longer training ranges lead to better performance of CoEvoGNN, which follows our intuition that longer training range provides more information and can make the model yield better performance.

5.6 Qualitative analysis

The goal of CoEvoGNN is to model dynamic attributed graph sequence for forecasting change of node attributes and evolution of graph structure at future time. We provide concrete examples on the predictions made by CoEvoGNN to better illustrate its effectiveness and practical utilities.

Specifically, we first examine the forecasted keywords (i.e., node attributes) at the last time step on the evolutionary co-authorship graph $\mathcal{D}_{\text{AU}}^{10K}$. One representative example is Dr. Jennifer Rexford from Princeton University. She is an influential researcher working in areas of computer networks and internet routing. Her most popular keywords from 2001 to 2007 include "network", "safe", "modular", "hierarchy", "cooperative" and etc. And, the forecasted keywords given



(a) Training time of CoEvoGNN is linear to the training graph sequence length T.

(b) Training time of CoEvoGNN is insensitive to the temporal evolution span S.

Fig. 7. Effiency of CoEvoGNN on $\mathcal{D}_{\text{BC}}^{\text{alp}}$ with different values of training length T and temporal evolution span S.

by CoEvoGNN at 2008 cover most of her previously associated keywords but also include some new ones such as "decentralized", "virtualized", and "seamless". They appeared in her collaborators' top keywords and turned out to be associated with an impactful paper entitled "OpenFlow: Enabling Innovation in Campus Networks" published at SIGCOMM Computer Communication Review in year 2008.

Similarly, we also examine the forecasted collaborations (i.e., links) given by CoEvoGNN on $\mathcal{D}_{\text{AU}}^{10K}$. Another good example is Dr. Kevin Skadron from the University of Virginia. He works in areas of computer architecture and parallel computing. We found CoEvoGNN can successfully forecast all six links to his co-authors of an impactful paper entitled "Rodinia: A Benchmark Suite for Heterogeneous Computing" published at IEEE International Symposium on Workload Characterization in 2009. The paper's title word "heterogeneous" did not appear in his previous associated keywords but did appear in the keywords of his collaborators. This verifies the captured impact from node attributes to graph structure. We conclude that CoEvoGNN can preserve the co-evolutionary patterns of attribute change and link formation in real dynamic graph sequence. These patterns can then be effectively utilized for forecasting future unseen node attributes and graph structure information.

5.7 Efficiency and sensitivity

We test the time efficiency of CoEvoGNN through different values of training length T and temporal evolution span S. All experiments are conducted on single server with dual 12-core Intel Xeon 2.10GHz CPUs with single NVIDIA GeForce GTX 2080 Ti GPU. Figure 7 shows the model's per epoch training time is generally linear to the length of the training graph sequence and is insensitive to different values of temporal evolution span. This is also confirmed by our analysis on the complexity of CoEvoGNN given in Section 4.3.

We also investigate the hyperparameter α of CoEvoGNN for balancing the attribute and structure evolutionary losses. Empirical analysis shows that the model is able to achieve optimal performance when setting α to mix those two evolutionary losses differing by no more than one order of magnitude.

TABLE 6
Capability of dynamic methods on (i) capturing the evolution of graph structure G, (ii) capturing the evolution of node attributes $\mathbf{X}$, (iii) forecasting future graph structure G, and (iv) forecasting future node attributes $\mathbf{X}$. CoEvoGNN meets requirements in all four categories.

	Method								
Capability	CTDNE	DynamicTriad	DySAT	WD/CD-GCN	GCRN	DCRNN	STGCN	EvolveGCN	**CoEvoGNN**
Evolution of G	✓	✓	✓	✓	✓			✓	✓
Evolution of $\mathbf{X}$				✓	✓	✓	✓	✓	✓
Forecasting G	✓	✓	✓					✓	✓
Forecasting $\mathbf{X}$						✓	✓		✓

6 Related Work

Graph neural networks. The convolution operation for graph from spectral domain using the graph Laplacian matrix was first proposed in [4]. An extension of a simple max/mean pooling at the beginning of the network to reduce the cost of graph Fourier transform operation was proposed in [33]. In [3], the authors proposed the spectral-based GCN model for semi-supervised node classification on graph. PATCHY-SAN [34] generalized the convolution operation to multiple graphs using graph labeling procedure to assign a unique order of nodes. R-GCN [35] extended the idea of GCN on knowledge graphs by training different weights for different relation types. GraphSAGE [6] was the first inductive graph representation learning method which aggregates attribute information from the sampled neighborhood of a node. PinSAGE [36] later adapted the the former one to a web-scale recommender system. In [37], the authors provided theoretical explanations to the expressive power of graph neural networks. Some recent reviews on methods and applications of representation learning on graphs were provided in [38], [39], [40]. All these models assume a single snapshot of static graph structure and node attributes as input. They cannot be directly applied on a sequence of evolutionary graphs.

Dynamic graph representation learning. Dynamic network embedding methods aim at modeling the temporal pattern of graph structure change without considering node attributes. CTDNE [41] proposed to model temporal structure dependencies in continuous-time dynamic networks by conducting temporal random walks. DynamicTriad [29] preserved the dynamic structural information by modeling the triadic closure process in network. DySAT [30] employed a self-attention mechanism over both neighbor nodes and historical representations. Timers [42] minimized the margin between reconstruction loss of incremental updates and the minimum loss in incremental SVD. dyngraph2vec [43] modeled the temporal transitions using a deep architecture composed of dense and recurrent layers. DynGEM [44] used a deep autoencoder to preserve the first-order and second-order proximities. DNE [45] adapted the Skip-gram embedding model for dynamic network. And, dynnode2vec [46] extended the static network embedding method node2vec [47] into the dynamic setting. These methods were not designed to handle node attributes. They can neither capture the evolution pattern of node attributes nor forecast future attribute information.

The set of spatiotemporal traffic forecasting methods aim at capturing the evolutionary pattern of node attributes given a fixed graph structure. DCRNN [31] modeled the traffic flow as a diffusion process on a directed graph and adopted an encoder-decoder architecture for capturing the temporal attribute dependencies. STGCN [32] modeled the traffic network as a general graph and employed a fully convolutional structure [11] on the temporal axis. These methods assume the graph structure remains static all the time, thus being incapable of capturing the evolution of graph structure or forecasting into future graph structure [48]. LRGCN [49] was proposed for predicting path failure in time-evolving graphs. It generates node embeddings by using an architecture combining the RNN and the GCN, and then employs a self-attention mechanism to merge node embeddings into path embeddings for binary failure classification. Similar to our COEVOGNN, the input also includes all historical graph snapshots. However, it also needs the path composition and label information which are highly traffic domain specific. This method cannot be applied in modeling dynamic graph in other domains such as an evolutionary co-authorship network.

When the graph structure and node (or edge) attributes are jointly evolving, dynamic graph can be generally categorized into continuous graphs and discrete graphs. Continuous graphs retain detailed temporal information of each edge formation [22]. DYREP [50] utilized the temporal point processes. TEDIC [51] and TGNS [52] proposed various local temporal aggregation operators. They are not applicable when the detailed event timestamps are not available. We focus on discrete graphs in this work which present as an ordered sequence of graph snapshots. Each snapshot contains aggregated dynamic information of a fixed time interval [17]. Some existing work on discrete graphs follow the idea of combining RNN model for capturing temporal changes with GNN as feature extractor [8] [10] [12]. VGRNN [53] combined variational graph inference [54] with GCRN [10] for learning probabilistic representations and capturing the potential variability observed in discrete graphs. Similar to the proposed COEVOGNN, DYSAT [55] and TGAT [56] also used the self-attention mechanism for learning temporal node representations. However, these two methods can only handle changing graph structures and do not model the co-evolution of node attributes and graph structure in real dynamic graphs. Thus, they cannot preserve the complex co-evolutionary patterns in dynamic graph sequence. A survey on dynamic graph learning can be found in [57].

Another recent work JODIE [58] aimed at learning the trajectory of embeddings in dynamic bipartite graph of users and items. It was designed to model the temporal interactions between two components of the dynamic bipartite graph where each interaction is marked by a timestamp and characterized by a feature vector. So, this method cannot be directly applied on general graph for capturing the co-evolutionary patterns. The capability of these dynamic methods and our COEVOGNN is summarized in Table 6.

Graph evolution analysis. There exists a line of work study the fundamental dynamics driving the evolution of graphs. The phenomenon of densification and shrinking diameters during graph evolution was studied in [17]. A frequency-based pattern describing the evolution of large networks over time was proposed in [59]. In [60], the authors searched for typical patterns of structural changes in dynamic networks for predicting the evolution of social networks. In [60], the authors searched for typical patterns of structural changes for predicting the evolution of social networks. [61] studied the structure and evolution of online social networks. A evolutionary model for graph classification was proposed in [62]. The cohesive co-evolution patterns in dynamic attributed graphs was studied in [63], and [64] studied mining heavy subgraphs in time-evolving networks. The trend mining in dynamic attributed graphs was studied in [65]. A comprehensive survey on evolutionary network analysis was provided in [22].

7 CONCLUSIONS

In this work, we proposed a new framework for learning node embeddings from evolutionary attributed graph and inferring future node representations. It aggregated the information in previous snapshots to the current one using temporal self-attention and employed a multi-task loss function based on attribute inference and link prediction over time. Experimental results demonstrated our method outperformed strong baselines on forecasting an entire future snapshot of co-authorship and virtual currency network.

ACKNOWLEDGMENTS

This research was supported in part by National Science Foundation (NSF) Grants no. IIS-1849816 and no. CCF-1901059. This research was also supported in part by the research project no. 2016/23/B/ST6/01735, the National Science Centre, Poland.

REFERENCES

[1] D. Wang, M. Jiang, Q. Zeng, Z. Eberhart, and N. V. Chawla, "Multi-type itemset embedding for learning behavior success," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2397–2406.

[2] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia, "Microsoft academic graph: When experts are not enough," *Quantitative Science Studies*, vol. 1, no. 1, pp. 396–413, 2020.

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

[4] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[5] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 731–739.

[6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[7] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 2017, pp. 633–641.

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

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

[10] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *International Conference on Neural Information Processing*. Springer, 2018, pp. 362–373.

[11] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016.

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

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[14] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

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

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

[17] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007.

[18] J. S. Coleman and J. S. Coleman, *Foundations of social theory*. Harvard university press, 1994.

[19] D. Wang, T. Jiang, N. V. Chawla, and M. Jiang, "Tube: Embedding behavior outcomes for predicting success," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1682–1690.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[21] D. Wang, M. Jiang, M. Syed, O. Conway, V. Juneja, S. Subramanian, and N. V. Chawla, "Calendar graph neural networks for modeling time structures in spatiotemporal user behaviors," *arXiv preprint arXiv:2006.06820*, 2020.

[22] C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 10, 2014.

[23] M. Qu, Y. Bengio, and J. Tang, "Gmnn: Graph markov neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[24] M. Jiang, C. Faloutsos, and J. Han, "Catchtartan: Representing and summarizing dynamic multicontextual behaviors," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 945–954.

[25] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 29–37.

[26] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 661–670.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 333–341.

[29] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[30] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dynamic graph representation learning via self-attention networks," *arXiv preprint arXiv:1812.09430*, 2018.

[31] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

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

[33] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[34] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[35] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018.

[36] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 974–983.

[37] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[38] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[39] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *arXiv preprint arXiv:1812.04202*, 2018.

[40] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[41] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018, pp. 969–976.

[42] Z. Zhang, P. Cui, J. Pei, X. Wang, and W. Zhu, "Timers: Error-bounded svd restart on dynamic networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[43] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowledge-Based Systems*, vol. 187, p. 104816, 2020.

[44] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*, 2018.

[45] L. Du, Y. Wang, G. Song, Z. Lu, and J. Wang, "Dynamic network embedding: An extended approach for skip-gram based network embedding." in *IJCAI*, vol. 2018, 2018, pp. 2086–2092.

[46] S. Mahdavi, S. Khoshraftar, and A. An, "dynnode2vec: Scalable dynamic network embedding," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 3762–3765.

[47] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016.

[48] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, 2020.

[49] J. Li, Z. Han, H. Cheng, J. Su, P. Wang, J. Zhang, and L. Pan, "Predicting path failure in time-evolving graphs," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1279–1289.

[50] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *ICLR*, 2019.

[51] Y. Wang, P. Li, C. Bai, and J. Leskovec, "Tedic: Neural modeling of behavioral patterns in dynamic social interaction networks," in *TheWebConf*, 2021.

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

[53] E. Hajiramezanali, A. Hasanzadeh, K. D. Narayanan, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," *NeurIPS*, vol. 32, 2019.

[54] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *stat*, vol. 1050, p. 21, 2016.

[55] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "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*, 2020.

[56] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *ICLR*, 2020.

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

[58] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1269–1278.

[59] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis, "Mining graph evolution rules," in *joint European conference on machine learning and knowledge discovery in databases*. Springer, 2009.

[60] B. Bringmann, M. Berlingerio, F. Bonchi, and A. Gionis, "Learning and predicting the evolution of social networks," *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 26–35, 2010.

[61] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks," in *Link mining: models, algorithms, and applications*. Springer, 2010, pp. 337–357.

[62] N. Jin, C. Young, and W. Wang, "Gaia: graph classification using evolutionary computation," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010.

[63] E. Desmier, M. Plantevit, C. Robardet, and J.-F. Boulicaut, "Cohesive co-evolution patterns in dynamic attributed graphs," in *International Conference on Discovery Science*. Springer, 2012.

[64] P. Bogdanov, M. Mongiovì, and A. K. Singh, "Mining heavy subgraphs in time-evolving networks," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, 2011, pp. 81–90.

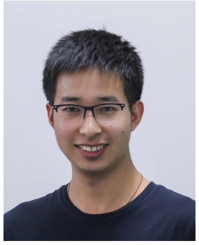
[65] E. Desmier, M. Plantevit, C. Robardet, and J.-F. Boulicaut, "Trend mining in dynamic attributed graphs," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 654–669.

Daheng Wang is currently a Ph.D. candidate at the University of Notre Dame. He received his M.S. degree from the University of Pittsburgh in 2015. His research field is machine learning, specifically representation learning, for user behavior modeling. He aims at building effective predictive models and recommender systems.

Zhihan Zhang received his B.S. degree from Peking University in 2020. He is currently a Ph.D. student at the University of Notre Dame. His research interests include natural language processing and data mining.

Yihong Ma received his B.S. degree from Shanghai University of Finance and Economics in 2020. He is currently a Ph.D. student at the University of Notre Dame. His research interests include data mining and machine learning.

Tong Zhao is currently a Ph.D. student at the University of Notre Dame. He received his B.S. degree from Case Western Reserve University in 2017. His research interests include computational behavior modeling, anomalous behavior detection, graph mining, and graph neural networks.



Tianwen Jiang is currently a Ph.D. candidate at Harbin Institute of Technology, Harbin, China. He received his B.S. degree from Harbin Institute of Technology in 2016. His research focuses on knowledge mining and structuring from massive text corpora.

Nitesh V. Chawla is the Frank M. Freimann Professor of Computer Science and Engineering at the University of Notre Dame. He is the Founding Director of the Lucy Family Institute for Data and Society. He has also served as the director of the Center for Network and Data Science. He started his tenure-track position at Notre Dame in 2007, and was promoted and tenured in 2011, and chaired full professor in 2015. He has published over 300 papers with over 38,000 citations and h-index of 68. His research is focused on data science, machine learning, and network science. His work has also led to transformative interdisciplinary applications in healthcare, environmental and climate sciences, education, and national security. He is the recipient of the 2015 IEEE CIS Outstanding Early Career Award for 2015, the IBM Watson Faculty Award in 2012, and the IBM Big Data and Analytics Faculty Award in 2013, the National Academy of Engineering New Faculty Fellowship, the Rodney Ganey Award in 2014 and Michiana 40 Under 40 in 2013. He is also a Fellow at the Kellogg Institute for International Studies; the Kroc Institute for International Peace Studies; the Liu Institute for Asia and Asian Studies; the Pulte Institute for Global Development; and the Reilly Center for Science, Technology, and Values. He serves on the steering committee for the Health and Wellbeing Initiative and Technology Ethics Center.

Meng Jiang is an assistant professor in the Department of Computer Science and Engineering at the University of Notre Dame. He joined the faculty in 2017, after completing his Ph.D. in Computer Science and Technology at Tsinghua University in 2015 and postdoctoral training in Computer Science at the University of Illinois at Urbana-Champaign. His research focuses on machine learning for data-driven decision making, user behavior modeling, and information extraction. He has published more than 70 papers in top conferences and journals of the relevant field such as the IEEE Transactions on Knowledge and Data Engineering, IEEE Intelligent Systems, Proceedings of ACM SIGKDD, IEEE ICDM, and AAAI. He has over 2600 citations according to Google Scholar. He has delivered eleven conference tutorials in the major conferences. He was the best paper finalist in ACM SIGKDD 2014.