Social Network Analysis and Mining (2021) 11:100
https://doi.org/10.1007/s13278-021-00816-1

ORIGINAL ARTICLE

Knowledge distillation on neural networks for evolving graphs

Stefanos Antaris[1,2] · **Dimitrios Rafailidis**[3] · **Sarunas Girdzijauskas**[1]

Received: 28 February 2021 / Revised: 12 July 2021 / Accepted: 13 July 2021 / Published online: 20 October 2021

Abstract
Graph representation learning on dynamic graphs has become an important task on several real-world applications, such as recommender systems, email spam detection, and so on. To efficiently capture the evolution of a graph, representation learning approaches employ deep neural networks, with large amount of parameters to train. Due to the large model size, such approaches have high online inference latency. As a consequence, such models are challenging to deploy to an industrial setting with vast number of users/nodes. In this study, we propose DynGKD, a distillation strategy to transfer the knowledge from a large teacher model to a small student model with low inference latency, while achieving high prediction accuracy. We first study different distillation loss functions to separately train the student model with various types of information from the teacher model. In addition, we propose a hybrid distillation strategy for evolving graph representation learning to combine the teacher's different types of information. Our experiments with five publicly available datasets demonstrate the superiority of our proposed model against several baselines, with average relative drop 40.60% in terms of RMSE in the link prediction task. Moreover, our DynGKD model achieves a compression ratio of 21:100, accelerating the inference latency with a speed up factor ×30, when compared with the teacher model. For reproduction purposes, we make our datasets and implementation publicly available at https://github.com/stefanosantaris/DynGKD.

Keywords Graph representation learning · Evolving graphs · Knowledge distillation

1 Introduction

Graph representation learning has been at the core of several machine learning tasks on graphs, such as node classification (Zhang et al. 2019; Qu et al. 2019; Kipf and Welling 2017) and link prediction (Kumar et al. 2019; Zhang and Chen 2018). The main objective is to learn low-dimensional node embeddings, so that the structure of the graph is reflected on the embedding space. Early approaches work primarily on static graphs (Grover and Leskovec 2016; Veličković et al. 2018; Perozzi et al. 2014). However, most real-world applications are dynamic, where graphs evolve over time. Recently, dynamic approaches have been proposed to capture both the topological and temporal properties of evolving graphs in the node embeddings (Sankar et al. 2020; Nguyen et al. 2018; Pareja et al. 2020). Such approaches have demonstrated remarkable performance on various applications, such as email spam detection (Akoglu et al. 2015), recommender systems (Cao et al. 2019; Ying et al. 2018), name disambiguation in citation networks (Zhang et al. 2018), molecular generation (You et al. 2018; Bresson and Laurent 2019), and so on.

Learning dynamic embeddings that preserve the time-varying structure and node interactions of an evolving graph is a fundamental problem. Existing representation strategies apply several techniques among consecutive graph snapshots to learn accurate node embeddings, such as recurrent neural networks (Pareja et al. 2020), attention mechanisms (Sankar et al. 2020), and temporal regularizers (Li et al. 2017). To preserve the structural and temporal properties of evolving graphs without loss of information, such strategies design neural network architectures with a large amount of model parameters. Given the large model size, existing strategies

✉ Stefanos Antaris
antaris@kth.se

Dimitrios Rafailidis
draf@uth.gr

Sarunas Girdzijauskas
sarunasg@kth.se

[1] KTH Royal Institute of Technology, Stockholm, Sweden

[2] HiveStreaming AB, Stockholm, Sweden

[3] University of Thessaly, Vólos, Greece

present high online inference latency. Despite their remarkable achievements in producing accurate node embeddings, these strategies are not suitable for real-world applications with almost real-time requirements of inference latency. For example, distributed live video streaming solutions in large enterprises, exploit the offices' internal high bandwidth network to distribute the video content among viewers/nodes (Roverso et al. 2015). To select the high-bandwidth connections, distributed solutions exploit graph neural networks to predict the throughput among viewers in real-time (Antaris and Rafailidis 2020b). Based on the predicted throughput, each viewer adapts the connections to efficiently distribute the video content. However, the high online inference latency of the large models negatively impacts the distribution of the video stream in an enterprise network (Antaris et al. 2020).

To reduce the online inference latency, knowledge distillation has been recently introduced to generate compact models without any information loss (Hinton et al. 2015; Bucilua et al. 2006). In particular, knowledge distillation trains a cumbersome model, namely *teacher*, without stringent requirements on inference latency. Therefore, the teacher model is trained as an offline process and employs neural networks with large model sizes. Once the teacher model is trained, the knowledge can be distilled to a compact model, namely *student*, through a well-designed distillation loss function. Therefore, the student model has significantly lower number of model parameters than the teacher model, while preserving high performance accuracy. Given the low online inference latency due to the small model size, the student model can be deployed to online applications (Phuong and Lampert 2019; Tang and Wang 2018; Asif et al. 2020; Qian et al. 2020; Kim and Rush 2016).

Existing knowledge distillation strategies fall in two main categories: (1) feature-based strategies that exploit the generated features of either the last layer or the intermediate layers as a supervision signal to train the student model (Romero et al. 2015; Huang and Wang 2017; Gou et al. 2021), and (2) response-based strategies where the distillation loss function minimizes the differences between the predicted values of the teacher and student models (Chen et al. 2017; Meng et al. 2019). Recently, the impact of knowledge distillation has been studied for static graph representation learning strategies (Yang et al. 2020; Chen et al. 2020; Ma and Mei 2019). These representation strategies extract the structural knowledge of a static graph and perform distillation to transfer the acquired knowledge to a compact student model. However, such approaches ignore the temporal aspect of the evolving networks. A recent attempt to employ knowledge distillation on a dynamic graph representation learning strategy has been presented in our preliminary study of the Distill2Vec model (Antaris and Rafailidis 2020a). Distill2Vec trains a teacher model on the offline graph snapshots and transfers the knowledge to the student model when learning on the online data. Distill2Vec employs a feature-based distillation strategy, by adopting the Kullback-Liebler divergence on the teacher and student node embeddings in the distillation loss function. However, Distill2Vec focuses only on the final node embeddings of the teacher model, ignoring the prediction accuracy of the teacher model and the additional information captured on the intermediate features/layers.

In this article, we propose a **Dyn**amic **G**raph representation learning model with **K**nowledge **D**istillation. DynGKD extends the Distill2Vec model, making the following contributions:

- We propose different distillation loss functions based on the way that we transfer the knowledge from the teacher model to the student model. We also present a hybrid strategy to combine the teacher's features and responses in the distillation loss function, allowing the student model to distill more information than incorporating only one distillation strategy separately.
- We conduct an extensive experimentation on networks with different characteristics, such as two real-world social networks and three evolving networks generated by live video streaming events. We demonstrate that our hybrid distillation strategy for the student model significantly reduces the model size, while constantly outperforms the teacher model and the baseline strategies in the link weight prediction task.

The remainder of the paper is organized as follows: Sect. 2 reviews the related work and in Sect. 3 we formulate the problem of knowledge distillation on dynamic graph representation learning. Section 4 describes the proposed model DynGKD, the experimental evaluation is presented in Sect. 5, and we conclude the study in Sect. 6.

2 Related work

2.1 Graph representation learning

Static Approaches Graph representation learning has attracted a surge of research in the recent years (Wang et al. 2020). Early attempts adopt traditional network embedding techniques, by capturing the distribution of the positive node pairs in the latent space (Hamilton et al. 2017b). DeepWalk (Perozzi et al. 2014) performs random walks on the graph and adopts the skip-gram model (Mikolov et al. 2013) to learn accurate network embeddings that correspond to the log-likelihood of observed nodes in the walks. Node2Vec (Grover and Leskovec 2016) extends the DeepWalk model by biasing the random walks, so as to compute different properties of the graph. Recently, HARP (Chen et al. 2018a)

coarsens related nodes to supernodes to improve the performance of random walks in DeepWalk and Node2Vec.

With the remarkable performance of neural networks, graph representation learning adopts various architectures to compute accurate node embeddings. Existing approaches employ spectral convolutions on graphs to learn node embeddings with similar structural properties (Kipf and Welling 2017). To alleviate the scalability issues of spectral convolutions on large graphs, GraphSage (Hamilton et al. 2017a) and FastGCN (Chen et al. 2018b) employ neighbrhood and importance sampling, respectively. GAT exploits an attention mechanism to transform the node properties to low-dimensional embeddings (Vaswani et al. 2017). Recently, DAGNN decouples the convolution operation to two key factors, that is node properties transformation and propagation (Liu et al. 2020). This allows DAGNN to improve the performance of convolution on graphs by adopting deeper neural network architectures than the previous approaches. However, such approaches are applied on static graphs, ignoring the dynamic properties of the nodes in evolving graphs.

Dynamic Approaches Dynamic graph representation learning approaches adopt various neural network architectures to model the structural and temporal properties of evolving graphs. Early attempts model the evolving graph as an ordered collection of graph snapshots and extend the static approaches to learn accurate node embeddings (Hamilton et al. 2017b). CTDNE employs temporal random walks on consecutive graph snapshots and applies the skip-gram model to learn the transition probability among two nodes (Nguyen et al. 2018). DNE adopts random walks on each graph snapshot and adjusts the embeddings of the nodes that present significant changes among consecutive graph snapshots (Du et al. 2018). DynGEM employs deep auto-encoders to compute the structural properties of each graph snapshot and exploits temporal smoothness methods to ensure stability on the computed node embeddings among consecutive graph snapshots (Goyal et al. 2018). Recently, DynVGAE uses variational auto-encoders (Kipf and Welling 2016) to ensure temporal smoothness by introducing weight parameter sharing among consecutive models (Goyal et al. 2018). DynamicTriad exploits the triadic closure as a supervised signal to capture the social homophily property in evolving social networks (Zhou et al. 2018). EvolveGCN (Pareja et al. 2020) and DynGraph2Vec (Goyal et al. 2020) adopt recurrent neural networks (RNNs) among consecutive graph convolutional networks (Kipf and Welling 2017) to model the evolution of the graph in the hidden state. Furthermore, DySAT (Sankar et al. 2020) employs graph attention mechanisms to capture the evolution of the graph.

Recent approaches compute the node embeddings by leveraging the time ordered node interactions of the evolving graph. DeepCoevolve exploits RNNs to define a point process intensity function, which allows the model to compute the influence of an interaction on the node embedding over time (Dai et al. 2016). Jodie (Kumar et al. 2019) employs both attention mechanism and RNN to predict the evolution of the node and adapt the generated embeddings accordingly. Moreover, TigeCMN (Zhang et al. 2020) designs coupled memory networks to store and update the node embeddings, while TDIG-MPNN (Chang et al. 2020) updates the embeddings through message passing on the high-order correlation nodes. However, existing approaches require large model sizes to efficiently capture the evolution of the graph. Due to the large model sizes, such approaches have significant online inference latency.

2.2 Knowledge distillation

Knowledge distillation has been widely adopted in several machine learning domains, such as image recognition (Ba and Caruana 2014; Hinton et al. 2015), recommender systems (Tang and Wang 2018; Chen et al. 2017), language translation (Kim and Rush 2016) to generate compact student models with low online inference latency (Bucilua et al. 2006). Knowledge distillation can be divided into three main categories: (1) response-based, (2) feature-based, and (3) relation-based strategies. The response-based strategies distill knowledge to the student model by exploiting the output of the teacher model. Response-based approaches consider the final output of the teacher model as a soft label to regularize the output of the student model (Hinton et al. 2015; Mirzadeh et al. 2020; Kim et al. 2021). Accounting for the output of teacher model to distill knowledge, the student model fails to capture the intermediate supervision applied by the teacher model (Gou et al. 2021). Instead, feature-based approaches distill the high-level information acquired by the teacher to the output features. FitNet incorporates the features of the intermediate layers to supervise the student model, minimizing the L2 distillation loss function (Romero et al. 2015). Moreover, Zhou et al. (2018a) consider the parameter sharing of intermediate layers among teacher and student models. Recently, Chen et al. (2020) formulate a feature embedding task to match the dimensions of the output features generated by the teacher and student models. Although response-based and feature-based strategies distill knowledge from the outputs of specific layers in the teacher model, these approaches ignore the the semantic relationship among the different layers. To handle this issue, relation-based approaches explore the relationships between different feature maps. SemCKD follows a cross-layer knowledge distillation strategy to address the different semantics of intermediate layers in the teacher and student models (Chen et al. 2021). In the IRG model, the student model distills knowledge from the Euclidean distance between examples observed by the teacher model (Liu et al.

2019). In Zagoruyko and Komodakis (2017), Zagoruyko et al. derive an attention map from the teacher's features to distill knowledge to the student model, while KDA (Qian et al. 2020) employs the Nyström low-rank approximation for kernel matrix to distill knowledge through landmark points. Nevertheless, such approaches focus on image processing, and have not been evaluated on evolving graphs.

A few attempts have been made to follow knowledge distillation strategies to reduce the model size of graph representation learning approaches. DMTKG calculates Heat Kernel Signatures to compute the node features, which are used as inputs into Graph Convolutional Networks (GCNs) to learn accurate node embeddings (Lee and Song 2019). DMTKG generates a compact student model by applying a response-based knowledge distillation strategy based on the weighted cross entropy distillation loss function. DistillGCN is a feature-based strategy that exploits the output features of the teachers' GCN layers to transfer the structural information of the graph to the student model (Yang et al. 2020). The distillation loss function in DistillGCN minimizes the prediction error of the student model on the online data and the Kullback Leibler divergence of the features generated by the teacher and student models. However, existing knowledge distillation strategies are designed to transfer knowledge from static graphs, ignoring the evolution of dynamic graphs.

3 Problem formulation

We model the evolution of a dynamic graph as a collection of graph snapshots over time, which is defined as follows (Sankar et al. 2020; Pareja et al. 2020; Nguyen et al. 2018; Antaris et al. 2020):

Definition 1 *Evolving Graphs* An evolving graph is defined as a sequence of K discrete graph snapshots $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_K\}$. At each timestep $k = 1, \ldots, K$, a snapshot $\mathcal{G}_k = (\mathcal{V}_k, \mathcal{E}_k, \mathbf{F}_k)$ is a weighted undirected graph which consists of $N_k = |\mathcal{V}_k|$ nodes, and a connection set $\mathcal{E}_k$. Each node $u \in \mathcal{V}_k$ has a c-dimensional feature vector $\mathbf{F}_k(u) \in \mathbf{F}_k$, with $\mathbf{F}_k \in \mathbb{R}^{N_k \times c}$. For each graph $\mathcal{G}_k$, we consider a weighted adjacency matrix $\mathbf{A_k}$, where $A_k(u, v) > 0$ if $e_k(u, v) \in \mathcal{E}_k$.

In an evolving graph, the node set $\mathcal{V}_k$ and edge set $\mathcal{E}_k$ vary among consecutive graph snapshots. As illustrated in Fig. 1, node $f \in \mathcal{V}_2$ joins the evolving graph in the snapshot $\mathcal{G}_2$, creating a new connection $e_2(f, d) \in \mathcal{E}_2$ with node $d \in \mathcal{V}_2$. Moreover, in the final snapshot $\mathcal{G}_K$, node b disappears, removing the respective edges. A graph representation learning strategy on evolving graphs is defined as follows (Antaris et al. 2020; Antaris and Rafailidis 2020b; Pareja et al. 2020):

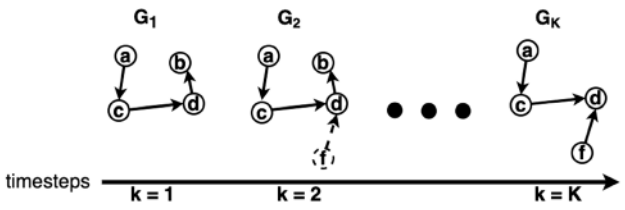

Fig. 1 Overview of an evolving graph over time. We denote with dotted lines the new nodes/edges of the graph snapshot

Definition 2 *Dynamic Graph Representation Learning* Given a sequence of $l \ll K$ previous consecutive graph snapshots $\mathcal{G} = \{\mathcal{G}_{k-l}, \ldots, \mathcal{G}_k\}$, the goal of dynamic graph representation learning is to compute d-dimensional node embeddings $\mathbf{Z}_k \in \mathbb{R}^{N_k \times d}$, with $d \ll N_k$, at the k-th timestep. The learned node embeddings $\mathbf{Z}_k$ should accurately capture the structural and temporal evolution of the graph, up to the k-th timestep.

Our study focuses on knowledge distillation strategies for dynamic graph representation learning approaches, as defined in the following (Antaris and Rafailidis 2020a):

Definition 3 *Knowledge Distillation on Dynamic Graph Representation Learning* The goal of knowledge distillation is to generate a compact student model $\mathcal{S}$, which distills the knowledge acquired by a large teacher model $\mathcal{T}$. The teacher model $\mathcal{T}$ learns the node embeddings $\mathbf{Z}^{\mathcal{T}}$ based on the first m graph snapshots $\mathcal{G}^{\mathcal{T}} = \{\mathcal{G}^{\mathcal{T}}_1, \ldots, \mathcal{G}^{\mathcal{T}}_m\}$, with $1 < m < K$, which correspond to the offline data. Having pretrained the teacher model $\mathcal{T}$, the student $\mathcal{S}$ computes the node embeddings $\mathbf{Z}^{\mathcal{S}}$ on the online data $\mathcal{G}^{\mathcal{S}} = \{\mathcal{G}^{\mathcal{S}}_{m+1}, \ldots, \mathcal{G}^{\mathcal{S}}_K\}$. The student model $\mathcal{S}$ distills the knowledge from the teacher $\mathcal{T}$ through a distillation loss function $\mathcal{L}^{\mathcal{D}}$.

4 Proposed model

4.1 Method overview

As illustrated in Fig. 2, the DynGKD model consists of the teacher and student models. The goal of the proposed DynGKD model is to train a large teacher model as an offline process and distill the knowledge of the teacher model to a small student model during training on the online data.

Teacher Model The teacher model takes as input the m offline graph snapshots $\mathcal{G}^{\mathcal{T}} = \{\mathcal{G}^{\mathcal{T}}_1, \ldots, \mathcal{G}^{\mathcal{T}}_m\}$. The role of the teacher model is to learn the node embeddings $\mathbf{Z}^{\mathcal{T}}$ to capture both the structural and temporal evolution of the m offline graph snapshots. Following Sankar et al. (2020), we adopt two self-attention layers to capture the structural and temporal evolution of the m graph snapshots. Provided that the teacher

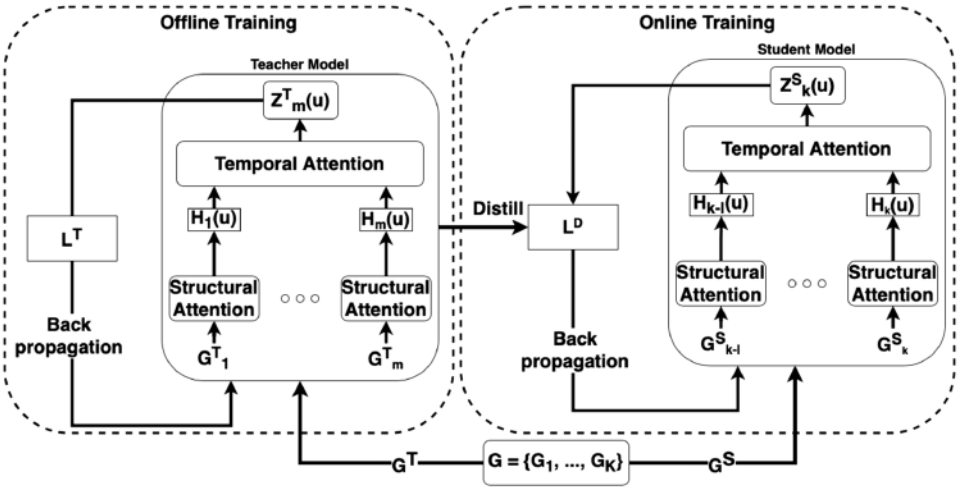

Fig. 2 Overview of the DynGKD model, given a sequence of discrete graph snapshots $\mathcal{G}$. The teacher model is trained as an offline process based on the offline graph snapshots $\mathcal{G}^{\mathcal{T}}$. Then, the student model is trained on the online graph snapshots $\mathcal{G}^{\mathcal{S}}$ and distills knowledge from the teacher model through the distillation loss function $\mathcal{L}^{\mathcal{D}}$

model is trained as an offline process, we employ a large number of model parameters to accurately capture the evolution of the graph for the offline graph snapshots.

Student Model Having pretrained the teacher model, we train a student model on the online graph snapshots. At each timestep k, the student model learns the node embeddings $\mathbf{Z}_k^{\mathcal{S}}$, by employing two self-attention layers on the l previous consecutive graph snapshots $\mathcal{G}^{\mathcal{S}} = \{\mathcal{G}_{k-l}^{\mathcal{S}}, \ldots, \mathcal{G}_k^{\mathcal{S}}\}$. We adopt a distillation loss function $L^{\mathcal{D}}$, during the online training of the student model, to transfer the acquired knowledge by the teacher model.

4.2 Teacher model on the offline data

The teacher model DynGKD-$\mathcal{T}$ learns the node embeddings $\mathbf{Z}^{\mathcal{T}}$ based on l consecutive graph snapshots. Provided that we pretrain the teacher model on the offline graph snapshots $\mathcal{G}^{\mathcal{T}} = \{\mathcal{G}_1^{\mathcal{T}}, \ldots, \mathcal{G}_m^{\mathcal{T}}\}$, we consider all the m snapshots during training ($l = m$), with $m \ll K$. Following Sankar et al. (2020), we capture the evolution of the graph by employing two self-attention layers. The structural attention layer captures the structural properties of each graph snapshot, and the temporal attention layer learns the complex temporal properties of the evolving graph.

Structural Attention Layer Given the offline graph snapshots $\mathcal{G}^{\mathcal{T}} = \{\mathcal{G}_1^{\mathcal{T}}, \ldots, \mathcal{G}_m^{\mathcal{T}}\}$, the input of the structural attention layer is the m feature vectors $\{\mathbf{F}_1, \ldots, \mathbf{F}_m\}$. The structural attention layer computes l structural node representations $\mathbf{H}(u) \in \mathbb{R}^{l \times d}$, with $l = m$, of the node $u \in \mathcal{V}$ by implementing a multi-head self-attention mechanism as follows (Veličković et al. 2018; Vaswani et al. 2017):

$$\mathbf{H}(u) = Concat(\mathbf{h}^1(u), \ldots, \mathbf{h}^g(u)) \tag{1}$$

$$\text{with} \quad \mathbf{h}(u) = ELU\left(\sum_{k=1}^{l} \sum_{v \in \mathcal{N}_k(u)} a_k(u,v)\mathbf{W}\mathbf{F}_k(u)\right)$$

where g is the number of attention heads, $\mathcal{N}_k(u)$ is the neighborhood set of the node $u \in \mathcal{V}_k$ at the graph snapshot $\mathcal{G}_k^{\mathcal{T}}$, $\mathbf{W} \in \mathbb{R}^{d \times c}$ is the weight transformation matrix of the c-dimensional node features $\mathbf{F}_k(u)$, and ELU is the exponential linear unit activation function. Variable $a_k(u,v)$ is the attention coefficient among the node $u \in \mathcal{V}_k$ and $v \in \mathcal{V}_k$, defined as follows:

$$a_k(u,v) = \frac{exp\Big(LeakyReLU(A_k(u,v)\mathbf{a}^\top[\mathbf{W}\mathbf{F}_k(u)||\mathbf{W}\mathbf{F}_k(v)])\Big)}{\sum_{w \in \mathcal{N}_k(u)} exp\Big(LeakyReLU(A_k(u,v)\mathbf{a}^\top[\mathbf{W}\mathbf{F}_k(u)||\mathbf{W}\mathbf{F}_k(w)]\Big)} \tag{2}$$

where $||$ is the concatenation operation to aggregate the transformed feature vectors of the nodes $u \in \mathcal{V}_k$ and $v \in \mathcal{V}_k$. Variable $\mathbf{a}^\top$ is a $2d$-dimensional weight vector parameterizing the aggregated feature vectors. The attention weight $a_k(u,v)$ expresses the impact of the node v on the node u at the k-th timestep, when compared with the neighborhood set $\mathcal{N}_k(u)$ (Vaswani et al. 2017).

Temporal Attention Layer The input of the temporal attention layer is the m structural node embeddings $\{\mathbf{H}_1, \ldots, \mathbf{H}_m\}$ learned by the structural attention layer. The temporal attention layer aims to capture the evolution of the graph over time. Therefore, we design the multi-head scale-dot product form of attention to learn m temporal representation $\mathbf{B}(u) \in \mathbb{R}^{m \times d}$ of each node $u \in \mathcal{V}_k$, as follows (Sankar et al. 2020; Vaswani et al. 2017):

$$\begin{aligned} &\mathbf{B}(u) = Concat(\mathbf{b}^1(u), \ldots, \mathbf{b}^p(u)) \\ &\text{with} \quad \mathbf{b}(u) = \boldsymbol{\beta}(u)(\mathbf{H}(u)\mathbf{W}^{value}) \end{aligned} \tag{3}$$

where p is the number of attention heads, $\mathbf{W}^{value} \in \mathbb{R}^{d \times d}$ is the weight parameter matrix to transform the structural embeddings $\mathbf{H}(u)$. Value $\boldsymbol{\beta}(u) \in \mathbb{R}^{l \times l}$ is the attention weight matrix, with $l = m$ for the teacher model, calculated as follows:

$$\boldsymbol{\beta}(u) = \sum_{i,j=k-l}^{k} \frac{exp(\frac{(\mathbf{H}_i(u)\mathbf{W}^{query})(\mathbf{H}_j(u)\mathbf{W}^{key})^\top}{\sqrt{l}})}{\sum_{r=k-l}^{k} exp(\frac{(\mathbf{H}_i(u)\mathbf{W}^{query})(\mathbf{H}_r(u)\mathbf{W}^{key})^\top}{\sqrt{l}})} \tag{4}$$

where $\mathbf{W}^{key} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}^{query} \in \mathbb{R}^{d \times d}$ are the weight parameter matrices of the l structured node embeddings $\mathbf{H}(u)$. The attention weight matrix $\boldsymbol{\beta}(u)$ indicates the differences between the structural node embeddings over the l graph snapshots (Vaswani et al. 2017).

Having learned the l structural node embeddings $\mathbf{H}(u)$ and temporal node embeddings $\mathbf{B}(u)$ of the node u, we calculate the final node embedding $\mathbf{Z}_l^{\mathcal{T}}(u)$ of the teacher model at the l-th timestep, as follows:

$$\mathbf{Z}_l^{\mathcal{T}}(u) = \mathbf{H}_l(u) + \mathbf{B}_l(u) \tag{5}$$

To train the teacher model, we formulate the Root Mean Squared Error (RMSE) loss function with respect to the node embeddings $\mathbf{Z}_l^{\mathcal{T}}$:

$$\min_{\mathbf{Z}_l^{\mathcal{T}}} L^{\mathcal{T}} = \sqrt{\frac{1}{N_l} \sum_{u \in \mathcal{V}_l} \sum_{v \in \mathcal{N}_l(u)} \left(\sigma(\mathbf{Z}_l^{\mathcal{T}}(u)\mathbf{Z}_l^{\mathcal{T}}(v)^\top - A_l(u,v) \right)^2} \tag{6}$$

where σ is the sigmoid activation function and the term $(\mathbf{Z}_l^{\mathcal{T}}(u)\mathbf{Z}_l^{\mathcal{T}}(v)^\top - A_l(u,v))$ is the reconstruction error of the neighborhood $\mathcal{N}_l(u)$ of the node $u \in \mathcal{V}_l$, at the l-th timestep. We optimize the weight parameter matrices in both the structural and temporal attention layers based on the loss function in Eq. 6 and the backpropagation algorithm with the Adam optimizer (Kingma and Ba 2015).

4.3 Knowledge distillation strategies

The student model learns the node embeddings $\mathbf{Z}^{\mathcal{S}}$ based on the online graph snapshots $\mathcal{G}^{\mathcal{S}}$. Provided that the student model distills knowledge from the pretrained teacher model DynGKD-$\mathcal{T}$, the student model requires a significantly low number of model parameters. At each timestep $k = m+1, \ldots, K$, the student model considers l consecutive graph snapshots $\{\mathcal{G}_{k-l}, \ldots, \mathcal{G}_k\}$, with $l \ll K$. We compute the l structural and temporal node embeddings based on Eqs. 1 and 3, respectively. The final node embeddings are computed according to Eq. 5.

To transfer the knowledge from the teacher model, the student model formulates a distillation loss function $L^{\mathcal{D}}$ during the online training. As aforementioned in Sect. 1, the knowledge distillation strategies can be categorized as response-based and feature-based, according to the type of information to transfer the information from the teacher model (Gou et al. 2021). To evaluate the impact of each distillation strategy on the learned embeddings $\mathbf{Z}^{\mathcal{S}}$ of the student model, we formulate one response-based and two feature-based distillation loss functions. Moreover, we propose a hybrid distillation loss function that exploits both the responses and features of the teacher model during the online training process of the student model.

Response-Based Distillation Strategy (*DynGKD-*$\mathcal{R}$) In the response-based distillation strategy DynGKD-$\mathcal{R}$, we focus on the prediction accuracy of the teacher model on the online data. The student model learns the node embeddings $\mathbf{Z}_k^{\mathcal{S}}$ at the k-th timestep, by minimizing the following distillation loss function (Antaris et al. 2020):

$$\min_{\mathbf{Z}_k^{\mathcal{S}}} L^{\mathcal{D}} = \gamma L_k^{\mathcal{S}} + (1-\gamma) L_k^{\mathcal{T}} \tag{7}$$

where $L_k^{\mathcal{S}}$ is the root mean squared error of the student model on the online data. Value $L_k^{\mathcal{T}}$ is the prediction error of the teacher model DynGKD-$\mathcal{T}$ in Eq. 6 on the online data. Hyper-parameter $\gamma \in [0,1]$ controls the tradeoff between the two losses. High γ values balance the training of the node representations $\mathbf{Z}_k^{\mathcal{S}}$ towards the errors of the student model, ignoring the knowledge of the teacher model. The distillation loss function $L^{\mathcal{D}}$ of DynGKD-$\mathcal{R}$ allows the student model to mimic the reconstruction errors of the teacher model based on Eq. 6, while significantly reducing the model size (Antaris et al. 2020).

Feature-Based Distillation Strategies (*DynGKD-*$\mathcal{F}$1 */DynGKD-*$\mathcal{F}$2) As aforementioned in Sect. 4.2, the teacher model computes node embeddings $\mathbf{Z}^{\mathcal{T}}$ that preserve both the structural and temporal evolution of the graph. We exploit the node embeddings $\mathbf{Z}^{\mathcal{T}}$ to supervise the training of the student model through the distillation loss function in the feature-based distillation strategy DynGKD-$\mathcal{F}$1 as follows:

$$\min_{\mathbf{Z}_k^{\mathcal{S}}} L^{\mathcal{D}} = \gamma L_k^{\mathcal{S}} + (1-\gamma) L_k^{F1} \tag{8}$$

where $L_k^{F1} = KL(\mathbf{Z}_k^{\mathcal{S}} | \mathbf{Z}_k^{\mathcal{T}})$ is the Kullback-Liebler (KL) divergence among the node embeddings $\mathbf{Z}_k^{\mathcal{S}}$ and $\mathbf{Z}_k^{\mathcal{T}}$. The L_k^{F1} term prevents the student node embeddings $\mathbf{Z}_k^{\mathcal{S}}$ to be significantly different from the teacher node embeddings $\mathbf{Z}_k^{\mathcal{T}}$. Therefore, by optimizing Eq. 8, the student model generates node embeddings $\mathbf{Z}_k^{\mathcal{S}}$ that match the embedding space of the teacher model. Instead, the distillation loss function of the response-based strategy DynGKD-$\mathcal{R}$ in Eq. 7 allows the student model to predict similar values with the teacher model, regardless of the differences between the generated node embeddings $\mathbf{Z}_k^{\mathcal{S}}$ and $\mathbf{Z}_k^{\mathcal{T}}$.

Table 1 Summary statistics of the five datasets

Attributes	Social networks		Video streaming networks		
	Yelp	ML-10M	LiveStream-4K	LiveStream-6K	LiveStream-16K
#Nodes	6569	80,555	3813	6655	17,026
#Connections	95,361	10,000,054	11,066	787,291	482,185
#Time steps	16	14	8	8	8

To generate the final node embeddings $\mathbf{Z}^{\mathcal{T}}$, the teacher model DynGKD-$\mathcal{T}$ computes intermediate structural embeddings $\mathbf{H}^{\mathcal{T}}$ and temporal embeddings $\mathbf{B}^{\mathcal{T}}$ based on Eqs. 1 and 3, respectively. Following (Romero et al. 2015), in the second examined feature-based distillation strategy DynGKD-$\mathcal{F}2$, we adopt the intermediate embeddings as a supervision signal to improve the training of the student model. We formulate the distillation loss function in DynGKD-$\mathcal{F}2$ to incorporate both the structural and temporal embeddings of the teacher model as follows:

$$\min_{\mathbf{Z}_k^{\mathcal{S}}} L^{\mathcal{D}} = \gamma L_k^{\mathcal{S}} + (1-\gamma) L_k^{F2} \tag{9}$$

where $L_k^{F2} = KL(\mathbf{H}_k^{\mathcal{S}} | \mathbf{H}_k^{\mathcal{T}}) + KL(\mathbf{B}_k^{\mathcal{S}} | \mathbf{B}_k^{\mathcal{T}})$ is the KL divergence among the structural and temporal node embeddings of the teacher and student models. Note that the feature-based knowledge distillation of DynGKD-$\mathcal{F}1$ in Eq. 8 allows the student model to mimic only the final embeddings of the teacher model. In contrast, the distillation loss function of DynGKD-$\mathcal{F}2$ in Eq. 9 transfers knowledge by matching both the structural and temporal embedding spaces of the teacher and student models. Therefore, the intermediate embeddings $\mathbf{H}_k^{\mathcal{S}}$ and $\mathbf{B}_k^{\mathcal{S}}$ of the student model are similar to the intermediate node embeddings $\mathbf{H}_k^{\mathcal{T}}$ and $\mathbf{B}_k^{\mathcal{T}}$ of the teacher model.

Hybrid Distillation Strategy (*DynGKD-*$\mathcal{H}$) Although the above strategies provide favorable information for the training of the student model, such strategies focus on transferring individual instance of knowledge from the teacher to the student. In particular, the student model is trained to mimic either the predicted values or the learned embeddings of the teacher model. To train a student model that distills knowledge from both the responses and the embeddings of the teacher model, we formulate a hybrid strategy DynGKD-$\mathcal{H}$ based on the following distillation loss function:

$$\min_{\mathbf{Z}_k^{\mathcal{S}}} L^{\mathcal{D}} = \gamma L_k^{\mathcal{S}} + (1-\gamma) L_k^{H} \tag{10}$$

where $L_k^H = L_k^{\mathcal{T}} + L_k^{F2}$. The term $L_k^{\mathcal{T}}$ corresponds to the prediction error of the teacher model based on Eq. 6, and L_k^{F2} is the KL divergence of the intermediate embeddings generated by the teacher and student models, according to Eq. 9. In contrast to the previously examined distillation strategies, the hybrid loss function in Eq. 10 allows the student model to distill knowledge from different outputs of the teacher model at the same time. This means that the $L_k^{\mathcal{T}}$ loss in Eq. 10 allows the student model to have similar prediction accuracy to the teacher model. In addition, the structural and temporal embeddings $\mathbf{H}_k^{\mathcal{S}}$ and $\mathbf{B}_k^{\mathcal{S}}$ of the student model reflect on the structural and temporal embedding space of the teacher model. As we will demonstrate in Sect. 5.4, this hybrid strategy allows the student model to learn more accurate node embeddings, achieving high prediction accuracy, while significantly reducing the number of required parameters.

5 Experiments

5.1 Datasets

To evaluate the performance of the examined models in different network characteristics, we use two datasets of social networks, and three datasets of live video streaming events. In Table 1, we summarize the datasets' statistics.

Social networks We consider two bipartite networks from Yelp[1] and MovieLens[2](ML-10M). Each graph snapshot in the Yelp dataset corresponds to the ratings of the users to the businesses within a 6 month period. In ML-10M, each graph snapshot consists of the user ratings to the movies within a 1 year period.

Video streaming networks We consider three video streaming datasets, which correspond to the connections of the viewers in real live video streaming events in enterprise networks (Antaris et al. 2020). The duration of each event is 80 minutes. As aforementioned in Sect. 1, each viewer establishes a limited number of connections, so as to distribute the video content with other viewers, using the offices' internal high bandwidth network. To efficiently identify the high-bandwidth connections, each viewer periodically adapts the connections. We monitor the established connections during a live video streaming event and model the viewers interactions as an undirected weighted dynamic graph. The weight of a graph edge corresponds to the throughput measured between two nodes/viewers. Each dataset consists of 8 discrete graph snapshots, where each graph snapshot

[1] https://www.yelp.com/dataset.

[2] https://grouplens.org/datasets/movielens/.

corresponds to the viewers' interactions with a 10 minute period.

5.2 Evaluation protocol

We evaluate the performance of our proposed model on the link weight prediction task on the online graph snapshots $\mathcal{G}^{\mathcal{S}}$. For the social networks, we consider the first 5 timesteps as the offline data and the remaining timesteps are the online data, that is 11 and 9 online graph snapshots for the Yelp and ML-10M datasets, respectively. For each dataset of the video streaming networks, we consider the first 3 graph snapshots as the offline data, and the remaining 5 correspond to the online data.

The task of link weight prediction is to predict the weight of the unobserved edges $\mathcal{U}_{k+1} = \mathcal{E}_{k+1} \setminus \{\mathcal{E}_1, \ldots, \mathcal{E}_k\}$ in the $k+1$ time step, given the node embeddings $\mathbf{Z}_k^{\mathcal{S}}$ generated by the student model at the k-th timestep. Following (Antaris et al. 2020), we concatenate the node embeddings $\mathbf{Z}_k^{\mathcal{S}}(u)$ and $\mathbf{Z}_k^{\mathcal{S}}(v)$, for each connection $(u, v) \in \mathcal{E}_k$, based on the Hadamard operator, and train a Multi-Layer Perceptron (MLP) model, using negative sampling. Having trained the MLP model, we input the concatenated node embeddings for the unobserved edges $\mathcal{U}_{k+1}$, to calculate the predicted weights.

We measure the prediction accuracy of each examined model, based on the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) metrics, defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{U}_{k+1}|} \sum_{e(u,v) \in \mathcal{U}_{k+1}} (A_{k+1}(u,v) - \mathbf{Z}_k^{\mathcal{S}^\top} \mathbf{Z}_k^{\mathcal{S}})^2} \tag{11}$$

$$\text{MAE} = \frac{1}{|\mathcal{U}_{k+1}|} \sum_{e(u,v) \in \mathcal{U}_{k+1}} |A_{k+1}(u,v) - \mathbf{Z}_k^{\mathcal{S}^\top} \mathbf{Z}_k^{\mathcal{S}}| \tag{12}$$

Note that the RMSE metric emphasizes on large prediction errors, rather than the MAE metric does. Following Pareja et al. (2020); Sankar et al. (2020); Antaris et al. (2020), to train the model at each time step k, we randomly select 20% of the unobserved links for validation set. The remaining 80% of the unobserved links are used as test set. We repeat each experiment 5 times and report the average RMSE and MAE over the five trials.

5.3 Examined models

We compare the performance of our proposed model with the following examined models:

- *DynVGAE* Mahdavi et al. (2020) is a dynamic joint variational auto-encoder architecture that shares parameters over consecutive graph auto encoders. Given that there is no publicly available implementation, we implemented DynVGAE from scratch and publish our code.[3]
- *DynamicTriad*[4] Zhou et al. (2018b) a dynamic graph representation learning approach that employs triadic closure to capture the changes over different graph snapshots.
- *TDGNN* Nguyen et al. (2018) a graph representation learning that applies aggregation functions on the temporal graph edges. For reproducability purposes, we implemented TDGNN and made our code publicly available.[5]
- *DyREP*[6] Trivedi et al. (2019) a two-time scale process that captures the temporal node interactions by employing deep recurrent model, so as to calculate the probability of occurrence of future links between two nodes.
- *EvolveGCN*[7] Pareja et al. (2020) a dynamic graph representation learning model with Gated Recurrent Units (GRUs) between the convolutional weights of consecutive GCNs.
- *DMTKG-$\mathcal{T}$* Ma and Mei (2019) the teacher model of the DMTKG knowledge distillation strategy. DMTKG-T employs Heat Kernel Signature (HKS) on static graph snapshots and uses Convolutional Neural Network layers to calculate the latent representations based on DeepGraph. To ensure fair comparison, we train DMTKG- T per snapshot, with each snapshot containing aggregated graph history up to k-th snapshot. We implemented DMTKG-$\mathcal{T}$ from scratch and published our code,[8] as there was no implementation available.
- *DMTKG-$\mathcal{S}$* Ma and Mei (2019) the student model of DMTKG, which minimizes the distillation loss function based on the root mean squared error.
- *KDA-$\mathcal{T}$* Qian et al. (2020) the teacher model of the KDA knowledge distillation strategy. Given that KDA is a model-agnostic distillation strategy, we employed the DynGKD model, presented in Sect. 4.2, for fair comparison.
- *KDA-$\mathcal{S}$* Qian et al. (2020) the student model of KDA, which distills knowledge from the teacher model by adopting the Nyström low-rank approximation for a kernel matrix Williams and Seeger (2001). For reproducability purposes, we publish our implementation,[9] as there is no publicly available implementation of KDA.
- *DynGKD-$\mathcal{T}$* the teacher model of the proposed approach, presented in Sect. 4.2.

[3] https://github.com/stefanosantaris/DynVGAE.
[4] https://github.com/luckiezhou/DynamicTriad.
[5] https://github.com/stefanosantaris/TDGNN.
[6] https://github.com/uoguelph-mlrg/LDG.
[7] https://github.com/IBM/EvolveGCN.
[8] https://github.com/stefanosantaris/DMTKG.
[9] https://github.com/stefanosantaris/KDA.

- *DynGKD-*$\mathcal{R}$ the student model that distills the knowledge from the DynGKD-$\mathcal{T}$ model based on the response-based distillation strategy in Eq. 7.
- *DynGKD-*$\mathcal{F}$1 the student model that employs the feature-based distillation loss function in Eq. 8 to transfer knowledge from the teacher model.
- *DynGKD-*$\mathcal{F}$2 the student model that mimics the intermediate node embeddings of the teacher model through the feature-based loss function in Eq. 9.
- *DynGKD-*$\mathcal{H}$ the proposed hybrid distillation strategy that transfers knowledge from the teacher model through the loss function in Eq. 10.

For all the examined models, we tuned the hyper-parameters based on a grid selection strategy on the validation set and report the results with the best configuration. In particular, in DynVGAE and DynamicTriad, we set the embedding size to $d = 256$ with window size $l = 2$ in all datasets. In TDGNN and DyREP, the embedding size is set to $d = 128$, with $l = 3$ previous graph snapshots. EvolveGCN uses $d = 32$ node embeddings dimension and $l = 2$ window size. In DMTKG-$\mathcal{T}$, the emebdding size is fixed to $d = 512$ in all datasets. In DMTKG-$\mathcal{S}$, we reduce the embedding size to $d = 32$. In KDA-$\mathcal{T}$ and our teacher model DynGKD-$\mathcal{T}$, we set the embedding size to $d = 128$, with $l = 5$ consecutive graph snapshots, and the number of head attentions is set to 16 in both Eqs. 1 and 3. In the feature-based student models DynGKD-$\mathcal{F}$1 and DynGKD-$\mathcal{F}$2, the hybrid model DynGKD-$\mathcal{H}$, and the relation-based student model KDA-$\mathcal{S}$, we reduce the number of head attentions to 2 and the window size $l = 3$, while the node embedding dimension is set to $d = 128$ for all datasets. The student model DynGKD-$\mathcal{R}$ achieves the best performance when apply 8 head attentions, with $d = 64$ node embedding size and $l = 2$ previous graph snapshots. As aforementioned in Sect. 4.3, the response-based distillation strategy DynGKD-$\mathcal{R}$ focuses on the prediction error of the teacher model, ignoring the information contained in the generated feature embeddings. Therefore, the student model DynGKD-$\mathcal{R}$ requires higher number of attention heads than the feature-based models, to capture the multiple latent facets of the online graph snapshots. Instead, the feature-based approaches distill information from the teacher output embeddings, which contain the information of the different latent facets. Therefore, the feature-based approaches DynGKD-$\mathcal{F}$1, DynGKD-$\mathcal{F}$2 and DynGKD-$\mathcal{H}$ require 2 attention heads during training on the online data. All experiments were executed on a single server with a CPU Intel Xeon Bronze 3106, 1.70GHz and a GPU Geforce RTX 2080 Ti. The operating system of the server was Ubuntu 18.04.5 LTS, and we implemented the DynGKD model using PyTorch 1.7.1. In Sect. 5.5, we study the influence of the knowledge distillation parameter γ on the performance of each student model.

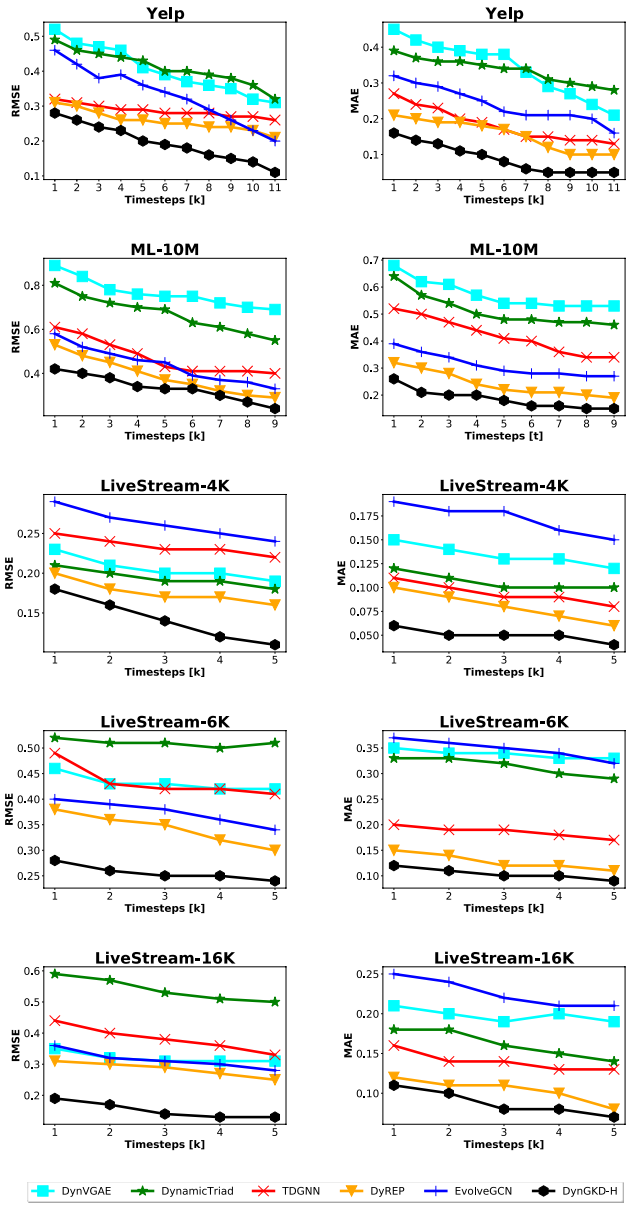

Fig. 3 Performance comparison of the proposed hybrid student model DynGKD-$\mathcal{H}$ against the non-distillation strategies in terms of RMSE and MAE

5.4 Performance evaluation

In Fig. 3, we evaluate the performance of the proposed hybrid student model DynGKD-$\mathcal{H}$ against the non-distillation strategies, in terms of RMSE and MAE. We observe that the proposed hybrid student model constantly outperforms the baseline approaches in all datasets. This suggests that DynGKD-$\mathcal{H}$ can efficiently learn node embeddings $\mathbf{Z}^S$ to capture the evolution of the online graph snapshots. Compared with the second best method DyREP, the proposed DynGKD-$\mathcal{H}$ model achieves high link weight prediction

accuracy, with average relative drops 38.32 and 88.68% in terms of RMSE and MAE in Yelp, 15.77 and 30.07% in the ML-10M dataset. For the video streaming datasets, the DynGKD-$\mathcal{H}$ model achieves average relative drops 26.43 and 59.33% in LiveStream-4K, 33.44 and 22.89% in LiveStream-6K, 89.35 and 19.17% in LiveStream-16K. The DyREP model outperforms the other baseline approaches, as the learned node embeddings capture the temporal changes of each node's neighborhood between consecutive graph snapshots. Instead, the other baseline approaches focus only on the structural changes of the graphs, ignoring the evolution of the node interactions over time. However, the DyREP model is designed to identify historical patterns on the nodes connections over time (Trivedi et al. 2019). Therefore, DyREP has low prediction accuracy on dynamic graphs that are updated significantly over time (Liu et al. 2020). The proposed DynGKD-$\mathcal{H}$ model overcomes this problem by capturing both the structural and temporal evolution of the graph using two self-attention layers. The structural attention layer contains the information of the structural properties of each graph snapshot, while the temporal attention layer captures the evolution of the generated structural embeddings over time. Therefore, the learned node embeddings of the DynGKD-$\mathcal{H}$ model reflect on the temporal variations of the graph.

In Fig. 4, we demonstrate the performance of the examined knowledge distillation strategies in terms of RMSE and MAE. We compare the proposed DynGKD model against DMTKG (Ma and Mei 2019), a baseline approach which employs a response-based distillation strategy on graph neural networks. Moreover, we evaluate our proposed model against KDA (Qian et al. 2020), a model-agnostic relation-based strategy. Given that we exploit the proposed DynGKD model, presented in Sect. 4 as the teacher model KDA-$\mathcal{T}$, we omit the results of KDA-$\mathcal{T}$. On inspection of Fig. 4, we observe that the hybrid student model DynGKD-$\mathcal{H}$ constantly outperforms its variants in all datasets. Note that DynGKD-$\mathcal{H}$ exploits the hybrid distillation strategy in Eq. 10 to transfer different types of information from the teacher model. Therefore, the DynGKD-$\mathcal{H}$ student model mimics both the prediction accuracy and the generated structural and temporal node embeddings of the teacher. Instead, the response-based DynGKD-$\mathcal{R}$ and the feature-based DynGKD-$\mathcal{F}1$ and DynGKD-$\mathcal{F}2$ approaches, exploit only a single source of information from the teacher. This means that the DynGKD-$\mathcal{H}$ strategy distills rich information from the teacher model, which allows the student model to capture the evolution of the graph more efficiently than its variants. We also observe that all the examined variants of the DynGKD student model constantly outperform the DynGKD-$\mathcal{T}$ teacher model in all datasets. This indicates that the generated student models overcome any bias introduced by the

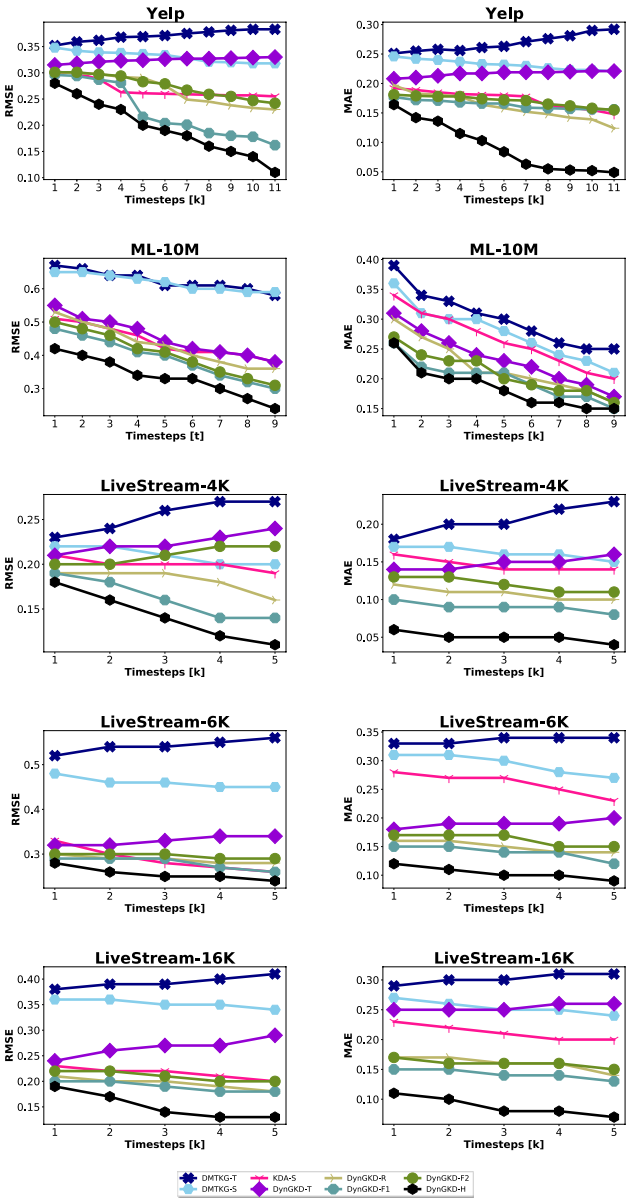

Fig. 4 Comparison of the examined teacher and student models in terms of RMSE and MAE

teacher model DynGKD-$\mathcal{T}$, during training on the offline data. In addition, the proposed hybrid student model DynGKD-$\mathcal{H}$ achieves higher prediction accuracy than the DMTKG-$\mathcal{T}$ and DMTKG-$\mathcal{S}$ models in all datasets. This occurs because DMTKG model is designed to learn node embeddings on static graphs, ignoring the evolution of the graph. Compared with the relation-based strategy KDA-$\mathcal{S}$, our proposed DynGKD-$\mathcal{H}$ model achieves superior performance in all datasets. KDA employs the Nyström low-rank approximation to distill knowledge through the selected landmark points, ignoring the temporal evolution of the nodes.

Table 2 Inference time in seconds and #Parameters in millions of the examined distillation strategies for the Yelp dataset

Baselines	Time steps										
	1	2	3	4	5	6	7	8	9	10	11
DMTKGT-$\mathcal{T}$											
#Parameters	2.182	2.182	2.182	2.182	2.182	2.182	2.182	2.182	2.182	2.182	2.182
Inference time	0.672	0.672	0.672	0.672	0.672	0.672	0.672	0.672	0.672	0.672	0.672
DMTKG-$\mathcal{S}$											
#Parameters	1.063	1.099	1.123	1.155	1.192	1.226	1.468	1.591	1.802	1.914	2.022
Inference time	0.484	0.491	0.512	1.121	0.579	0.682	0.699	0.832	0.905	0.952	1.035
KDA-$\mathcal{T}$											
#Parameters	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636
Inference time	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503
KDA-$\mathcal{S}$											
#Parameters	0.873	0.966	1.060	1.232	1.323	1.420	1.515	1.607	1.670	1.728	1.765
Inference time	0.198	0.201	0.203	0.205	0.208	0.210	0.240	0.280	0.366	0.397	0.401
DynGKD-$\mathcal{T}$											
#Parameters	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636	6.636
Inference time	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503	1.503
DynGKD-$\mathcal{R}$											
#Parameters	1.626	1.813	1.200	2.179	2.345	2.525	2.719	2.910	3.094	3.219	3.410
Inference time	0.214	0.257	0.312	0.366	0.388	0.405	0.447	0.489	0.516	0.572	0.602
DynGKD-$\mathcal{F}$1											
#Parameters	0.873	0.966	1.060	1.232	1.323	1.420	1.515	1.607	1.670	1.728	1.765
Inference time	0.198	0.201	0.203	0.205	0.208	0.210	0.240	0.280	0.366	0.397	0.401
DynGKD-$\mathcal{F}$2											
#Parameters	0.873	0.966	1.060	1.232	1.323	1.420	1.515	1.607	1.670	1.728	1.765
Inference time	0.198	0.201	0.203	0.205	0.208	0.210	0.240	0.280	0.366	0.397	0.401
DynGKD-$\mathcal{H}$											
#Parameters	0.873	0.966	1.060	1.232	1.323	1.420	1.515	1.607	1.670	1.728	1.765
Inference time	0.198	0.201	0.203	0.205	0.208	0.210	0.240	0.280	0.366	0.397	0.401

In Tables 2, 3 and 4, we present the number of required parameters in millions to train each examined distillation strategy. Moreover, we report the online inference time in seconds to compute the node embeddings. As aforementioned in Sect. 4, the teacher models DynGKD-$\mathcal{T}$, DMTKG-$\mathcal{T}$ and KDA-$\mathcal{T}$ learn the node embeddings based on the offline graph snapshots $\mathcal{G}^{\mathcal{T}}$. Therefore, the DynGKD-$\mathcal{T}$, DMTKG-$\mathcal{T}$ and KDA-$\mathcal{T}$ models have the same number of required parameters and inference times, when evaluated on the online graph snapshots $\mathcal{G}^{\mathcal{S}}$. As aforementioned in Sect. 5.3, KDA-$\mathcal{T}$ adopts the DynGKD model, thus requiring the same number of parameters as the DynGKD-$\mathcal{T}$ model. Moreover, the feature-based strategies DynGKD-$\mathcal{F}$1 and DynGKD-$\mathcal{F}$2 and the hybrid strategy DynGKD-$\mathcal{H}$ require the same number of parameters to train on the online graph snapshots. This occurs because the distillation loss functions in Eq. 8–10 exploit the features of the teacher to transfer knowledge to the student model. Therefore, the student model computes node embeddings that reflect on the same embedding space as the teacher embeddings. Instead, in DynGKD-$\mathcal{R}$ the student model distills knowledge from the predicted values of the teacher model, ignoring the information captured by the generated embeddings, thus having different number of required parameters. In addition, the response-based KDA-$\mathcal{S}$ strategy achieves high prediction accuracy when setting the same number of parameters as DynGKD-$\mathcal{F}$1, DynGKD-$\mathcal{F}$2 and DynGKD-$\mathcal{H}$, thus achieving similar inference time. On inspection of Tables 2, 3 and 4, we observe that the feature-based distillation strategies DynGKD-$\mathcal{F}$1 and DynGKD-$\mathcal{F}$2, and the hybrid strategy DynGKD-$\mathcal{H}$ require significantly a lower number of parameters than the response-based approach DynGKD-$\mathcal{R}$. This occurs because the generated model in DynGKD-$\mathcal{R}$ strategy requires high number of attention heads to capture the multiple facets of the evolving graph. In addition, we calculate the compression ratio of the DynGKD-$\mathcal{H}$ strategy, as the number of parameters to train the student model, when compared with the size of the teacher model. The DynGKD-$\mathcal{H}$ student model achieves average compression ratios of 21:100 and 69:100 for the Yelp and ML-10M datasets, when compared with the required parameters of the teacher model DynGKD-$\mathcal{T}$. For the live video streaming datasets, the averaged

Table 3 Inference time in seconds and #Parameters in millions of the examined distillation strategies for the ML-10M dataset

Baselines	Time steps								
	1	2	3	4	5	6	7	8	9
DMTKGT-$\mathcal{T}$									
#Parameters	30.562	30.562	30.562	30.562	30.562	30.562	30.562	30.562	30.562
Inference time	3.182	3.182	3.182	3.182	3.182	3.182	3.182	3.182	3.182
DMTKG-$\mathcal{S}$									
#Parameters	15.106	16.023	17.692	19.296	20.921	21.749	23.982	24.591	25.881
Inference time	2.682	2.703	2.713	2.785	2.792	2.817	2.2833	2.864	2.899
KDA-$\mathcal{T}$									
#Parameters	86.019	86.019	86.019	86.019	86.019	86.019	86.019	86.019	86.019
Inference time	10.864	10.864	10.864	10.864	10.864	10.864	10.864	10.864	10.864
KDA-$\mathcal{S}$									
#Parameters	12.210	13.212	14.332	15.515	17.110	18.187	19.185	20.229	20.707
Inference time	1.135	1.618	1.923	2.185	2.572	3.016	3.551	3.852	4.168
DynGKD-$\mathcal{T}$									
#Parameters	86.019	86.019	86.019	86.019	86.019	86.019	86.019	86.019	86.019
Inference time	10.864	10.864	10.864	10.864	10.864	10.864	10.864	10.864	10.864
DynGKD-$\mathcal{R}$									
#Parameters	24.300	26.302	28.542	30.909	34.098	36.252	38.247	40.336	41.291
Inference time	3.874	4.125	5.011	5.123	5.872	6.023	6.249	7.038	7.492
DynGKD-$\mathcal{F}$1									
#Parameters	12.210	13.212	14.332	15.515	17.110	18.187	19.185	20.229	20.707
Inference time	1.135	1.618	1.923	2.185	2.572	3.016	3.551	3.852	4.168
DynGKD-$\mathcal{F}$2									
#Parameters	12.210	13.212	14.332	15.515	17.110	18.187	19.185	20.229	20.707
Inference time	1.135	1.618	1.923	2.185	2.572	3.016	3.551	3.852	4.168
DynGKD-$\mathcal{H}$									
#Parameters	12.210	13.212	14.332	15.515	17.110	18.187	19.185	20.229	20.707
Inference time	1.135	1.618	1.923	2.185	2.572	3.016	3.551	3.852	4.168

compression ratios are 13:100 in LiveStream-4K, 14:100 in LiveStream-6K, and 32:100 in Livestream-16K. Provided that the proposed hybrid student model DynGKD-$\mathcal{H}$ reduces the model size, we demonstrate the impact of the number of parameters on the online inference time of the node embeddings. We observe that the DynGKD-$\mathcal{H}$ achieves a speed up factor ×17 on average, for the latency inference time in Yelp dataset, when compared with the DynGKD-$\mathcal{T}$ teacher model. DynGKD-$\mathcal{H}$ achieves a speed up factor ×25 for the latency inference time in ML-10M dataset, ×42 in LiveStream-4K, ×23 in LiveStream-6K, and ×36 in Livestream-16K. On inspection of Tables 2, 3 and 4 and Fig. 4, we find that the feature-based strategies and the hybrid strategy require the same amount of parameters during training. However, the DynGKD-$\mathcal{H}$ constantly outperforms the feature-based strategies in terms of RMSE accuracy. This occurs because the DynGKD-$\mathcal{H}$ exploits multiple types of information from the teacher model, which allows the student model to overcome any bias captured in the learned node embeddings of the teacher model. Therefore, the proposed DynGKD-$\mathcal{H}$ student model significantly reduces the model size and the online inference latency, while achieving high prediction accuracy.

5.5 Impact of knowledge distillation

In Fig. 5, we present the impact of the hyper-parameter γ (Eqs. 7–10) on the performance of each knowledge distillation strategy. We vary the hyper-parameter γ from 0.1 to 0.9 by a step of 0.1, to study the influence of the teacher DynGKD-$\mathcal{T}$ model on the different distillation loss functions $L^{\mathcal{D}}$, during the online training process of the student model. For each value of γ, we report the average RMSE value of each knowledge distillation strategy over all the online graph snapshots $\mathcal{G}^{\mathcal{S}}$. We observe that all strategies achieve the best prediction accuracy when the hyper-parameter γ is set to 0.4 for the Yelp and ML-10M datasets, and 0.3 in LiveStream-4K, LiveStream-6K and LiveStream-16K. For low values of γ, the distillation strategies emphasize more on the knowledge of the teacher model than the student model. Therefore, the student model prevents any additional training on the online data, which negatively impacts the learned

Table 4 Inference time in seconds and #Parameters in millions of the examined distillation strategies for the LiveStream datasets

	Time steps				
	1	2	3	4	5
LiveStream-4K					
DMTKGT-$\mathcal{T}$					
#Parameters	3.673	3.673	3.673	3.673	3.673
Inference time	0.879	0.879	0.879	0.879	0.879
DMTKG-$\mathcal{S}$					
#Parameters	1.192	1.216	1.290	1.356	1.418
Inference time	0.634	0.679	0.684	0.692	0.705
KDA-$\mathcal{T}$					
#Parameters	6.960	6.960	6.960	6.960	6.960
Inference time	1.174	1.174	1.174	1.174	1.174
KDA-$\mathcal{S}$					
#Parameters	0.879	0.905	0.937	0.941	0.945
Inference time	0.456	0.468	0.496	0.512	0.548
DynGKD-$\mathcal{T}$					
#Parameters	6.960	6.960	6.960	6.960	6.960
Inference time	1.174	1.174	1.174	1.174	1.174
DynGKD-$\mathcal{R}$					
#Parameters	1.639	1.690	1.754	1.762	1.770
Inference Time	0.471	0.534	0.577	0.593	0.612
DynGKD-$\mathcal{F}$1					
#Parameters	0.879	0.905	0.937	0.941	0.945
Inference time	0.456	0.468	0.496	0.512	0.548
DynGKD-$\mathcal{F}$2					
#Parameters	0.879	0.905	0.937	0.941	0.945
Inference time	0.456	0.468	0.496	0.512	0.548
DynGKD-$\mathcal{H}$					
#Parameters	0.879	0.905	0.937	0.941	0.945
Inference time	0.456	0.468	0.496	0.512	0.548
LiveStream-6K					
DMTKGT-$\mathcal{T}$					
#Parameters	5.129	5.129	5.129	5.129	5.129
Inference time	6.016	6.016	6.016	6.016	6.016
DMTKG-$\mathcal{S}$					
#Parameters	2.564	2.758	2.897	2.918	2.994
Inference Time	2.015	2.102	2.113	2.127	2.174
KDA-$\mathcal{T}$					
#Parameters	10.042	10.042	10.042	10.042	10.042
Inference time	9.906	9.906	9.906	9.906	9.906
KDA-$\mathcal{S}$					
#Parameters	1.334	1.382	1.384	1.395	1.407
Inference time	1.085	1.909	2.634	2.906	3.046
DynGKD-$\mathcal{T}$					
#Parameters	10.042	10.042	10.042	10.042	10.042
Inference time	9.906	9.906	9.906	9.906	9.906
DynGKD-$\mathcal{R}$					
#Parameters	2.549	2.645	2.648	2.670	2.695
Inference time	1.980	2.560	3.893	4.472	4.844
DynGKD-$\mathcal{F}$1					
#Parameters	1.334	1.382	1.384	1.395	1.407
Inference time	1.085	1.909	2.634	2.906	3.046
DynGKD-$\mathcal{F}$2					
#Parameters	1.334	1.382	1.384	1.395	1.407
Inference time	1.085	1.909	2.634	2.906	3.046
DynGKD-$\mathcal{H}$					
#Parameters	1.334	1.382	1.384	1.395	1.407
Inference time	1.085	1.909	2.634	2.906	3.046
LiveStream-16K					
DMTKGT-$\mathcal{T}$					
#Parameters	10.437	10.437	10.437	10.437	10.437
Inference time	8.015	8.015	8.015	8.015	8.015
DMTKG-$\mathcal{S}$					
#Parameters	5.219	5.934	7.824	8.012	8.135
Inference time	1.235	1.462	3.873	4.991	6.826
KDA-$\mathcal{T}$					
#Parameters	9.020	9.020	9.020	9.020	9.020
Inference Time	7.927	7.927	7.927	7.927	7.927
KDA-$\mathcal{S}$					
#Parameters	1.750	2.384	3.170	3.501	3.517
Inference time	0.529	1.228	2.359	4.179	6.184
DynGKD-$\mathcal{T}$					
#Parameters	9.020	9.020	9.020	9.020	9.020
Inference time	7.927	7.927	7.927	7.927	7.927
DynGKD-$\mathcal{R}$					
#Parameters	3.381	4.648	6.219	6.882	6.915
Inference time	0.887	2.056	4.084	6.558	7.012
DynGKD-$\mathcal{F}$1					
#Parameters	1.750	2.384	3.170	3.501	3.517
Inference time	0.529	1.228	2.359	4.179	6.184
DynGKD-$\mathcal{F}$2					
#Parameters	1.750	2.384	3.170	3.501	3.517
Inference time	0.529	1.228	2.359	4.179	6.184
DynGKD-$\mathcal{H}$					
#Parameters	1.750	2.384	3.170	3.501	3.517
Inference time	0.529	1.228	2.359	4.179	6.184

embeddings to capture the evolution of the graph. Instead, increasing the value of γ degrades the performance of the distillation strategies. This occurs because the student model is trained with low supervision from the teacher model.

Additionally, in Fig. 5, we study the influence of the hyper-parameter γ on the online inference latency of the knowledge distillation strategies. For each value of γ, we report the average inference latency in seconds over all the online graph snapshots. Given that DynGKD-$\mathcal{F}$1, DynGKD-$\mathcal{F}$2 and DynGKD-$\mathcal{H}$ exploit the teacher embeddings

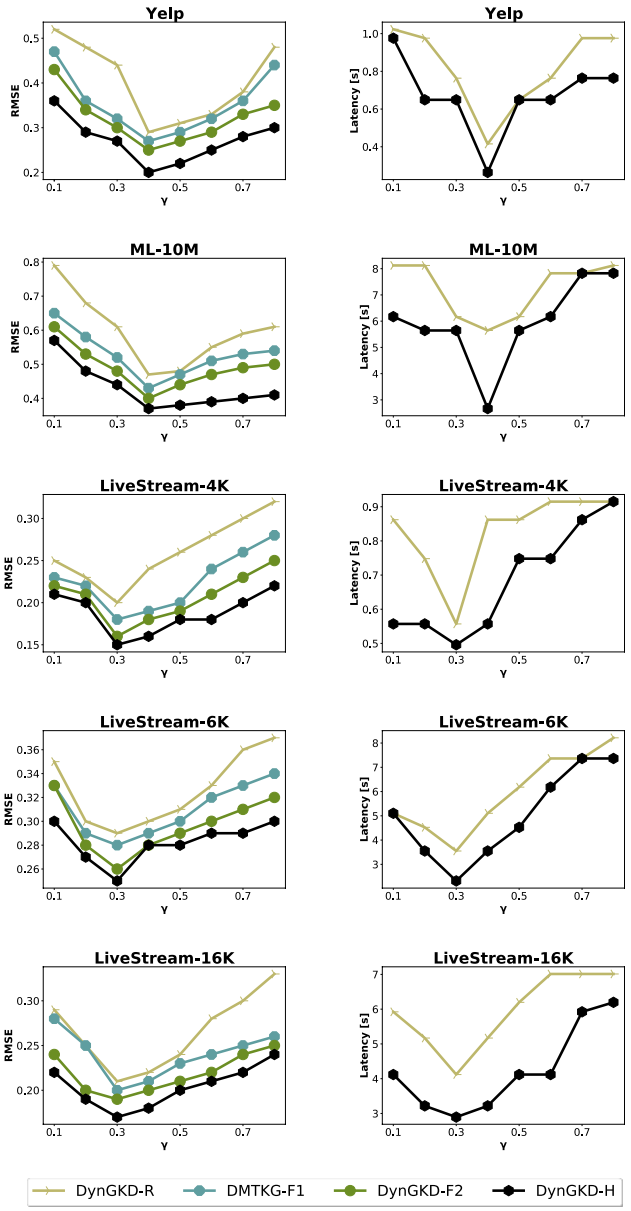

Fig. 5 Impact of parameter γ on the prediction accuracy and online inference latency of the examined DynGKD student models. We omit the DynGKD-$\mathcal{F}1$ and DynGKD-$\mathcal{F}2$ strategies, as they have the same model size with DynGKD-$\mathcal{H}$ for the online inference latency experiments

to transfer knowledge to the student model, the generated student models learn node embeddings that are similar to the embeddings of the teacher model. Therefore, the student models in DynGKD-$\mathcal{F}1$, DynGKD-$\mathcal{F}2$ and DynGKD-$\mathcal{H}$ require the same number of parameters during training. We omit the DynGKD-$\mathcal{F}1$ and DynGKD-$\mathcal{F}2$ strategies, as the student models have the same inference latency as DynGKD-$\mathcal{H}$. On inspection of Fig. 5, we observe that the best γ values, in terms of online inference latency, are 0.4 in Yelp and ML-10M, and 0.3 in LiveStream-4K, LiveStream-6K and LiveStream-16K. Increasing the value of γ negatively impacts the online inference latency of the student model. This occurs because the student model distills less knowledge from the teacher model. As a consequence, the student model requires a large amount of parameters to capture the evolution of the graph. This observation reflects on the importance of balancing the impact of the teacher model on the training process of the student model.

6 Conclusions

In this article, we presented the DynGKD model, a knowledge distillation strategy on neural networks for evolving graphs. The proposed DynGKD model generates a compact student model which efficiently captures the evolution of the online graph snapshots in the node embeddings, while achieving low online inference latency. We introduced three distillation loss functions to transfer different types of information from the teacher model to the student model. Moreover, we proposed a hybrid distillation loss function that combines both the predicted values and the embeddings of the teacher model to the student model. This allows the student model to distill different information from the teacher model and capture the evolution of the online graph snapshots, while requiring small model sizes. Evaluated against several baseline approaches on five real-world datasets, we demonstrated the efficiency of our model to reduce the online inference latency on the online data, while achieving high prediction accuracy. Our experiments showed that the proposed hybrid student model DynGKD-$\mathcal{H}$ achieves 40.60 and 44.02% relative drops in terms of RMSE and MAE in all datasets, respectively. Moreover, the proposed hybrid student model achieves averaged compression ratio of 21:100, when compared with the teacher model, which corresponds to a speed up factor ×30 on average for the online inference time. An interesting future direction is to explore adaptive aggregation strategies on the generation of the structural and temporal embeddings in Eq. 5, aiming to capture various properties of the graph evolution (Hamilton et al. 2017a; Wang et al. 2020). Accounting for the dynamic nature of the evolving graphs, we also plan to investigate relation-based approaches with online distillation strategies, where both the teacher and the student models capture the semantics of the data examples simultaneously (Guo et al. 2020; Mirzadeh et al. 2020; Sun et al. 2021). This means that the teacher model will not only compute the structural and temporal properties of an evolving graph, but will also assist the student model with updated information.

Funding Open access funding provided by Royal Institute of Technology.

References

Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Discov 29(3):626–688

Antaris S, Rafailidis D (2020) Distill2vec: dynamic graph representation learning with knowledge distillation. In: ASONAM

Antaris S, Rafailidis D (2020) Vstreamdrls: dynamic graph representation learning with self-attention for enterprise distributed video streaming solutions. In: ASONAM

Antaris S, Rafailidis D, Girdzijauskas S (2020) Egad: evolving graph representation learning with self-attention and knowledge distillation for live video streaming events

Asif U, Tang J, Harrer S (2020) Ensemble knowledge distillation for learning improved and efficient networks

Ba J, Caruana R (2014) Do deep nets really need to be deep? In: Advances in Neural Information Processing Systems, vol 27. Curran Associates, Inc.

Bresson X, Laurent T (2019) A two-step graph convolutional decoder for molecule generation. In: NeurIPS

Bucilua C, Caruana R, Niculescu-Mizil A (2006) Model compression. In: KDD, pp 535–541

Cao Y, Wang X, He X, Hu Z, Chua TS (2019) Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences. In: WWW, pp 151–161

Chang X, Liu X, Wen J, Li S, Fang Y, Song L, Qi Y (2020) Continuous-time dynamic graph learning via neural interaction processes, pp 145–154

Chen D, Mei JP, Zhang Y, Wang C, Wang Z, Feng Y, Chen C (2021) Cross-layer distillation with semantic calibration. In: AAAI, vol 35, pp 7028–7036

Chen G, Choi W, Yu X, Han T, Chandraker M (2017) Learning efficient object detection models with knowledge distillation. In: NeurIPS, pp 742–751

Chen H, Perozzi B, Hu Y, Skiena S (2018) Harp: hierarchical representation learning for networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 32

Chen H, Wang Y, Xu C, Xu C, Tao D (2020) Learning student networks via feature embedding. IEEE Trans Neural Netw Learn Syst 32(1):25–35

Chen J, Ma T, Xiao C (2018) Fastgcn: fast learning with graph convolutional networks via importance sampling. Preprint arXiv:1801.10247

Chen Y, Bian Y, Xiao X, Rong Y, Xu T, Huang J (2020) On self-distilling graph neural network. Preprint arXiv:2011.02255

Dai H, Wang Y, Trivedi R, Song L (2016) Deep coevolutionary network: embedding user and item features for recommendation. Preprint arXiv:1609.03675

Du L, Wang Y, Song G, Lu Z, Wang J (2018) Dynamic network embedding: an extended approach for skip-gram based network embedding. In: IJCAI, pp 2086–2092

Gou J, Yu B, Maybank SJ, Tao D (2021) Knowledge distillation: a survey. Int J Comput Vis 129(6):1789–1819

Goyal P, Chhetri SR, Canedo A (2020) dyngraph2vec: capturing network dynamics using dynamic graph representation learning. Knowl-Based Syst 187:104816

Goyal P, Kamra N, He X, Liu Y (2018) Dyngem: deep embedding method for dynamic graphs. Preprint arXiv:1805.11273

Grover A, Leskovec J (2016) Node2vec: scalable feature learning for networks. In: KDD, pp 855–864

Guo Q, Wang X, Wu Y, Yu Z, Liang D, Hu X, Luo P (2020) Online knowledge distillation via collaborative learning. In: CVPR, pp 11020–11029

Hamilton WL, Ying R, Leskovec J (2017a) Inductive representation learning on large graphs. In: NeurIPS, pp 1025–1035

Hamilton WL, Ying R, Leskovec J (2017b) Representation learning on graphs: methods and applications. Preprint arXiv:1709.05584

Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. In: NIPS

Huang Z, Wang N (2017) Like what you like: knowledge distill via neuron selectivity transfer. Preprint arXiv:1707.01219

Kim J, Hyun M, Chung I, Kwak N (2021) Feature fusion for online mutual knowledge distillation. In: ICPR, pp 4619–4625

Kim Y, Rush AM (2016) Sequence-level knowledge distillation. In: EMNLP, pp 1317–1327

Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: ICLR

Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv:abs/1611.07308

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: ICLR

Kumar S, Zhang X, Leskovec J (2019) Predicting dynamic embedding trajectory in temporal interaction networks. In: SIGKDD, pp 1269–1278

Lee S, Song BC (2019) Graph-based knowledge distillation by multi-head attention network. Preprint arXiv:1907.02226

Li J, Dani H, Hu X, Tang J, Chang Y, Liu H (2017)Attributed network embedding for learning in a dynamic environment. In: CIKM, pp 387–396

Liu M, Gao H, Ji S (2020) Towards deeper graph neural networks. In: KDD, pp 338–348

Liu Y, Cao J, Li B, Yuan C, Hu W, Li Y, Duan Y (2019) Knowledge distillation via instance relationship graph. In: CVPR, pp 7096–7104

Liu Z, Huang C, Yu Y, Song P, Fan B, Dong J (2020) Dynamic representation learning for large-scale attributed networks. In: CIKM, pp 1005–1014

Ma J, Mei Q (2019) Graph representation learning via multi-task knowledge distillation. Preprint arXiv:1911.05700

Mahdavi S, Khoshraftar S, An A (2020) Dynamic joint variational graph autoencoders. In: Machine learning and knowledge discovery in databases, pp 385–401

Meng Z, Li J, Zhao Y, Gong Y (2019) Conditional teacher-student learning. In: ICASSP, pp 6445–6449

Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: NIPS, pp 3111–3119

Mirzadeh SI, Farajtabar M, Li A, Levine N, Matsukawa A, Ghasemzadeh H (2020) Improved knowledge distillation via teacher assistant. In: AAAI, pp 5191–5198

Nguyen GH, Lee JB, Rossi RA, Ahmed NK, Koh E, Kim S (2018) Continuous-time dynamic network embeddings. In: WWW, pp 969–976

Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, Kanezashi H, Kaler T, Schardl TB, Leiserson CE (2020) EvolveGCN: evolving graph convolutional networks for dynamic graphs. In: AAAI

Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: KDD, pp 701–710

Phuong M, Lampert C (2019) Towards understanding knowledge distillation. In: ICML, pp 5142–5151

Qian Q, Li H, Hu J (2020) Efficient kernel transfer in knowledge distillation. arXiv:abs/2009.14416

Qu M, Bengio Y, Tang J (2019) Gmnn: graph markov neural networks. In: ICML, pp 5241–5250

Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y (2015) Fitnets: hints for thin deep nets. In: ICLR

Roverso R, Reale R, El-Ansary S, Haridi S (2015) Smoothcache 2.0: Cdn-quality adaptive http live streaming on peer-to-peer overlays. In: MMSys, pp 61–72

Sankar A, Wu Y, Gou L, Zhang W, Yang H (2020) Dysat: deep neural representation learning on dynamic graphs via self-attention networks. In: WSDM, pp 519–527

Sun L, Gou J, Yu B, Du L, Tao D (2021) Collaborative teacher-student learning via multiple knowledge transfer. Preprint arXiv:2101.08471

Tang J, Wang K (2018) Ranking distillation: learning compact ranking models with high performance for recommender system. In: KDD, pp 2289–2298

Trivedi R, Farajtabar M, Biswal P, Zha H (2019) Dyrep: learning representations over dynamic graphs. In: ICLR

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser LU, Polosukhin I (2017) Attention is all you need. In: NeurIPS, vol 30

Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: ICLR

Wang X, Bo D, Shi C, Fan S, Ye Y, Yu PS (2020) A survey on heterogeneous graph embedding: methods, techniques, applications and sources. Preprint arXiv:2011.14867

Williams C, Seeger M (2001) Using the nyström method to speed up kernel machines. In: NeurIPS, vol 13

Yang Y, Qiu J, Song M, Tao D, Wang X (2020) Distilling knowledge from graph convolutional networks. In: CVPR

Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale recommender systems. In: KDD, pp 974–983

You J, Liu B, Ying R, Pande V, Leskovec J (2018) Graph convolutional policy network for goal-directed molecular graph generation. In: NeurIPS

Zagoruyko S, Komodakis N (2017) Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In: ICLR

Zhang M, Chen Y (2018) Link prediction based on graph neural networks. In: NeurIPS

Zhang Y, Zhang F, Yao P, Tang J (2018) Name disambiguation in aminer: clustering, maintenance, and human in the loop. In: KDD, pp 1002–1011

Zhang Y, Pal S, Coates M, Ustebay D (2019) Bayesian graph convolutional neural networks for semi-supervised classification. In: AAAI, pp 5829–5836

Zhang Z, Bu J, Ester M, Zhang J, Yao C, Li Z, Wang C (2020) Learning temporal interaction graph embedding via coupled memory networks. In: WWW, pp 3049–3055

Zhou G, Fan Y, Cui R, Bian W, Zhu X, Gai K (2018a) Rocket launching: a universal and efficient framework for training well-performing light net. In: AAAI

Zhou L, Yang Y, Ren X, Wu F, Zhuang Y (2018b) Dynamic network embedding by modelling triadic closure process. In: AAAI