Dynamic Graph Representation Based on Temporal and Contextual Contrasting

Jin Huang
South China Normal University
Wentai Zhu
South China Normal University
Jing Xiao
South China Normal University
Tian Lu
South China Normal University
Weihao Yu (yuwh3@chinatelecom.cn)
Research Institute of China Telecom Corporate Ltd

Research Article

Keywords: Dynamic graph, Graph representation learning, Contrastive learning, Mutual information

Posted Date: May 4th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1588877/v1

Dynamic Graph Representation Based on Temporal and Contextual Contrasting

Jin Huang[1], Wentai Zhu[1], Jing Xiao[1], Tian Lu[1] and Weihao Yu[2*]

[1]South China Normal University, Guangzhou, 510630, Guangdong, China.
[2]Research Institute of China Telecom Corporate Ltd., Guangzhou, 510630, Guangdong, China.

*Corresponding author(s). E-mail(s): yuwh3@chinatelecom.cn;
Contributing authors: huangjin@m.scnu.edu.cn;
2020023065@m.scnu.edu.cn; xiaojing@scnu.edu.cn;
tianlu@m.scnu.edu.cn;

Abstract

Dynamic graph representation learning is critical for graph-based downstream tasks such as link prediction, node classification, and graph reconstruction. Many graph-neural-network–based methods have emerged recently, but most are incapable of tracing graph evolution patterns over time. To solve this problem, we propose a continuous-time dynamic graph framework: dynamic graph temporal contextual contrasting (DGTCC) model, which integrates both the temporal and topology information and mines the latent evolution trend of graph representation. In this model, the node representation is first generated by a self-attention–based temporal encoder, which measures the importance weights of neighbor nodes in temporal sub-graphs and then stores them in the contextual memory module. After sampling the node representation from the memory module, the model maximizes the mutual information of the same node that occurred in two nearby temporal views by the contrastive learning mechanism, which helps track the evolutional trend of nodes. In inductive learning settings, the results on four real datasets demonstrate the superiority of the proposed DGTCC model.

Keywords: Dynamic graph, Graph representation learning, Contrastive learning, Mutual information

1 Introduction

Graph representation learning[1] has aroused significant attention since it exhibits excellent potential in real-world applications such as graph anomaly detection[2], link prediction, and drug discovery[3]. Various graph representation methods are being developed to transform original graph data into low-dimensional vectors while preserving the intrinsic properties of the graph. Most of the proposed works are mainly designed for static graphs, which means that the nodes and edges of the graph are stationary and do not change over time. However, both nodes and edges in a graph often change dynamically in the real world; there is usually an increase or decrease in the number of edges, nodes, and the change of nodes attribute themselves. As a result of the complex time evolution graph structure, representation learning for dynamic graphs is challenging. For example, a new user joins a social network, users in the social network create new relationships, users in an e-commerce platform continues to interact with new items, and new connections occur in a communication network over time. To apply the existing graph representation model to a dynamic graph, we must treat it as a static graph and completely ignore its evolutionary structure. However, dynamic information has been substantiated to facilitate various graph analysis tasks, such as community detection[4], link prediction[5], and network embedding[6]. Therefore, excellent potential to develop dynamic graph representation methods is demonstrated by considering the evolutionary characteristics of graphs.

According to the modeling methods of dynamic graphs[7], these tasks can be roughly divided into discrete-time methods[1, 8, 9] and continuous-time methods[10, 10, 11]. The former methods rely on constructing a discrete-time dynamic graph, which approximates the dynamic graph as a series of graph snapshots that change over time. Generally, static graph modeling techniques such as GCN[12] or GAT[13] are applied to each snapshot. Then, a recurrent neural network or self-attention mechanism is introduced to capture the complex time dependence between snapshots. However, discrete-time methods may be sub-optimal because they ignore fine-grained time and structural information, critical in real-world applications. For example, in an e-commerce user-item graph, a new interaction is more likely to represent the latest preferences of a user. In addition, introducing edges (interactions) affects node attributes. Node information must be updated whenever a new interaction occurs since the distribution of these edges on the timeline is not uniform. Therefore, continuous-time dynamic graph-based graph approaches are attracting increasingly more attention in graph representation learning, and the most advanced results are achieved. For example, in TGAT[14] a continuous-time kernel encoder with a self-attention mechanism to aggregate information from the temporal neighborhood is proposed. In TGN[15], a generic temporal aggregation framework with node-level memory mechanisms is introduced. In APAN[16], the concepts of asynchrony and mailbox are introduced. Once the interaction is completed, detailed information is sent as an

email to the mailbox of the K-hop neighbor. By reading out the mailbox of the relevant node, the real-time inference is generated simultaneously.

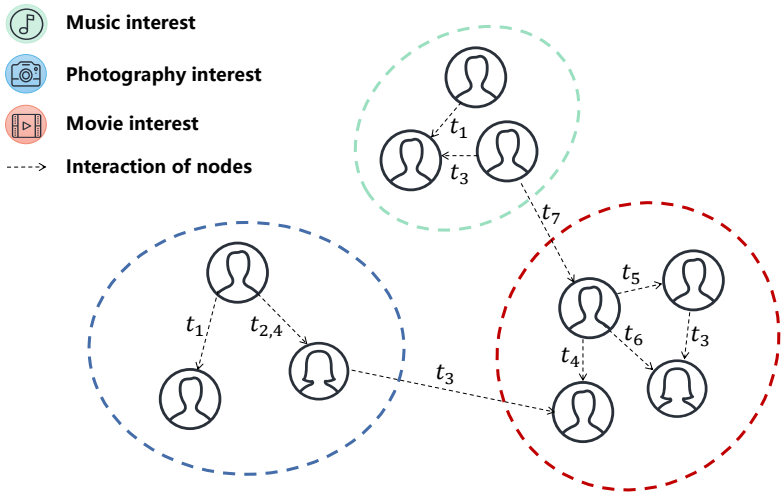

Fig. 1: From the perspective of the dynamic social network graph structure, many user interactions often exist within the same community structure, and user interactions between different communities would be relatively sparse. Individual representation evolves over time, but does not change dramatically.

Owing to the development of self-supervised learning in computer vision and natural language processing, many researchers have extended self-supervised learning to graph representation learning. Currently, there is very limited work that combines self-supervision with link prediction in dynamic graph representation. The graph self-supervised learning method [17] is used to learn node representation by sampling the weighted sub-graph and generating the node attributes and edges of the mask in the sub-graph. Still, the time cost is often expensive in the pre-training process. Contrasting learning aims to learn representations that preserve similarity via contrasting two or more semantically views (obtained via data augmentations of the graph). The existing contrasting-based graph self-supervised learning methods for dynamic graphs learn graph representations by comparing data with positive and negative samples in the feature space[18]. However, most dynamic graph representation models that introduce contrasting learning ignore the semantic association between the temporal neighborhood of each individual. Inspired by time consistency[19], it is reasonable to assume that evolution is generally “smooth," in that the semantics of each remain unchanged over a small amount of time. Based on the above assumptions, we propose a novel continuous-time

dynamic graph representation framework via temporal and contextual contrasting, called DGTCC. The main contributions of our work are summarized as follows.

- We propose a novel dynamic graph representation framework to capture both the temporal and topology information; this is followed by integrating a self-supervised task into the training process of our model to mine the latent evolution trend of graph representation.
- We generalize the contrasting learning strategy and maximize the agreement of the same node embeddings between two temporal views, which encourages mutually independent positive samples to spread apart in the representation space without explicitly sampling negative samples.
- The proposed model is evaluated on four real-world datasets for interaction prediction. Experimental results demonstrate the effectiveness of the proposed method.

2 Related Work

2.1 Conventional Methods of Graph Representation

Graph representation models conventionally focus on sample multiplex structural information in a graph and generate node sequences to learn low-dimensional representations of nodes. DeepWalk[20] was the first algorithm proposed to learn node embedding in an unsupervised manner. Like word embedding tasks, the motivation of DeepWalk is that nodes in the graph and words in the corpus follow a power-law distribution, and Node2vec[21] extends DeepWalk by combining breadth-first and depth-first sampling to obtain properties of homophily and structural equivalence. DeepWalk only captures the first-order similarity between nodes, while LINE[22] captures both first- and second-order similarity. Representations can also be induced from Laplacians of the adjacency matrix by non-negative matrix factorization[23, 24]. However, those graph representation methods are limited by their transductive learning setting and cannot directly predict unseen nodes, but must retrain the entire model.

2.2 Contemporary Graph-neural-network-based Methods

2.2.1 Static Graph.

After the first proposed of traditional graph convolutional neural network (GCN)[12], the newer models of graph-neural-network (GNN) -based graph representation methods have attracted widespread attention and achieved promising results. A GCN is divided into spectral- and spatial-domain methods. The spectral-domain method converts a graph from non-Euclidean to Euclidean space, and the spatial-domain method extracts features from graphs by convolution kernels to handle variable-length neighbor nodes. The traditional graph representation method based on transductive learning cannot

build a prediction model. A new data point added to the test dataset would have to retrain the model from scratch. Therefore, Graphsage[25] is proposed to learn the embedding of each node inductively. Specifically, each node is represented by an aggregation of its neighborhood. Thus, neighbor nodes can also represent a new node, even if it did not appear in the training process in the graph. Based on the above innovation, it was proposed in GAT[13] that attention mechanisms assign different weights to different neighbor nodes.

2.2.2 Discrete-time Methods.

A discrete-time dynamic graph (DTDG) is often represented by a series of graph structure snapshots. The discrete-time window is used to represent the continuous-time interaction between nodes and the evolution trend of the graph. The advantage of using discrete-time graph methods is that the static graph model can be used for each snapshot of the graph. In previous studies, DySAT[26] was the first DTDG representation method that linearly extracted structural and temporal information based on a self-attention mechanism. DNE[27] applies a skip-gram model to DTDG representation learning. The basis is to learn a mapping function for each time graph slice, which refers to the combinatorial objective optimization function of LINE[22]. Unlike the traditional DTDG modeling method, EvolveGCN[28] focuses specifically on GCNs, using recursive neural networks [29] to inject dynamic information into the parameters of the GCN to form an evolving sequence. Although previous DTDG representation methods have achieved leading promising results, model performance is sensitive to the choice of window size, and timing information may be lost in snapshots. We focus herein on continuous-time dynamic graph (CTDG) representation models, as much empirical evidence suggests the superiority of CTDG representation methods on dynamic graphs.

2.2.3 Continuous-time Methods.

Continuous-time methods directly operate on a time evolution graph and focus on designing different time aggregators to extract information rather than on time discretization. A dynamic graph is represented as a series of interactions in chronological order, with precise timestamps recorded. In recent research, Dyrep[30] can express the changes of association and communication in a unified manner by defining events and can generate node representation for newly added nodes in the graph quickly and efficiently. Jodie[31] uses two recurrent neural networks to update the representation of the node at each interaction and can also represent the node's future embedding change trajectory. TGAT[14] uses a time coding kernel in conjunction with a graph attention layer to aggregate time neighbors and be able to infer embeddings for new nodes inductively and observed nodes as the graph evolves. A TGN[15] encapsulates the aggregation of TGAT and leverages node-level memory to capture long-term dependencies. However, the disadvantages of the above approach are a poor understanding of remote dependencies, the difficulty of training, and the inherent weaknesses of a recursive neural network (RNN). In the APAN[16], an

asynchronous mail propagation mechanism is introduced. When the interaction between the two nodes is completed, the interaction information would be delivered as "mail" to the "mailbox" of the k-hop neighbor. The graph query and calculation phase would be transferred to the back of the model inference. In AdaGNN[32], a boosting-based meta learner for GNNs was proposed that automatically learns multiple projections and the corresponding embedding spaces, and captures different aspects of the graph signals.

2.3 Contrastive Learning on Graph Representation

Among the self-supervised learning methods, the self-supervised methods of contrasting learning have elicited broad adoption in many graph representation studies and have achieved significant results. The meaning of contrast learning on the graph is that for any two nodes, the more similar (belonging to the same category) they are, the closer the graph representation would be, and vice versa. In DeepWalk, the similarity of the nodes is considered to select the node for the next walk. The main inspiration of Deep InfoMax is the use of local and global mutual information. DGI[33] uses the readout function to combine global information to obtain the representation of nodes and constructs negative samples (rearranges the features of the corresponding nodes, combined with topology information) to make the generated node representations closer to positive samples at the same time.

Many existing studies have also proved that contrasting learning methods that do not use negative examples in self-supervised learning tasks can also achieve better results. In[34], it is mentioned that the potential representation of nodes can be better learned by not using negative samples when performing contrasting learning. In addition, in BYOL[35], characterization is further learned by enhancing characterization, negative samples are not used, and training degradation is avoided by increasing prediction and stop gradient. According to SimSiam[36], negative samples can be ignored in self-supervised learning tasks and only Siamese networks and stop-gradient operations can be relied on to achieve the most advanced performance. These methods have successfully improved the representation learning of visual data.

Different samples are selected as positive or negative examples in contrasting learning, leading to different experimental effects. However, recent research on dynamic graph modeling with the introduction of contrasting learning[17, 18] only considers maximizing the similarity between positive sample nodes and minimizing the similarity between negative samples, which ignores the characteristics of the interactive change trend over time of nodes. On the contrary, DGTCC aggregates the changes in network topology and node attributes when learning the hidden-layer representation of dynamic network nodes and synthesizes the temporal information of node interaction using temporal context contrasting learning.

3 METHOD

3.1 Problem definition

Static Graph. We use $\mathcal{G} = (V, \mathcal{E})$, which is used in many traditional graph representation models, to denote a static graphs, where $V \in \{v_1, v_2, ...v_n\}$ is the set of n nodes in the static graph, and $\mathcal{E} \subseteq V \times V$ is the set of edges generated by interactive nodes.

Dynamic Graph. The characterization of the dynamic graph includes two methods: DTDG and CTDG. Among them, a DTDG is often represented by a series of graph structure snapshots $\mathcal{G} = (\mathcal{G}_1, \mathcal{G}_2, ...\mathcal{G}_t)$ with the same length t. Discrete time windows are used to represent continuous-time interactions between nodes. Therefore, the performance of the DTDG model is susceptible to the choice of window size, and temporal information may be lost in the snapshot.

In contrast, the input of the CTDG model is a sub-graph of all event interactions generated at time t, and the node representation is dynamically updated through the temporal sub-graph aggregation model. Interaction can be expressed as $G_\lambda = (v_i, v_j, t)$. Compared to discrete-time dynamic graphs, a CTDG can better describe the interaction information between nodes.

3.2 Proposed Method

First, a brief overview of the proposed method is given; then, each part of the method is elaborated. DGTCC primarily includes an encoder, temporal information processing, and a timing context comparison module and decoder. Attention-based encoders are used to extract low-dimensional vector representations of interactive nodes over time. The timing information processing part fuses the new node representation learned by the encoder with the previously stored node historical representation. In the timing context comparison module, we designed a specific comparison loss for intra-fragment and inter-fragment graphs to increase the temporal diversity of dynamic graph representation. The overall framework is shown in Figure 2.

3.2.1 Dynamic Graph Encoders

Since a GCN assigns the same weight to each neighbor node, a GCN usually does not have a good effect in processing dynamic graphs because the importance of the node in the neighbor nodes $\mathcal{N} \in \mathcal{R}^n$ is not exactly the same; n is the number of nodes in the set, and $\mathcal{N}_i$ denotes the set of neighbor nodes of node v_i.

We use an attention-based graph encoder to learn the embedded $\mathcal{Z}_i$ of the interaction node in the current local sub-graph. The encoder introduces a classical attention mechanism to measure the importance of different neighbors of node v_i by aggregating the d-hop time neighborhood node information. We define the source and destination nodes as v_s, v_d. For an interaction produced by v_s and v_d at time t, our goal is to dynamically capture the current

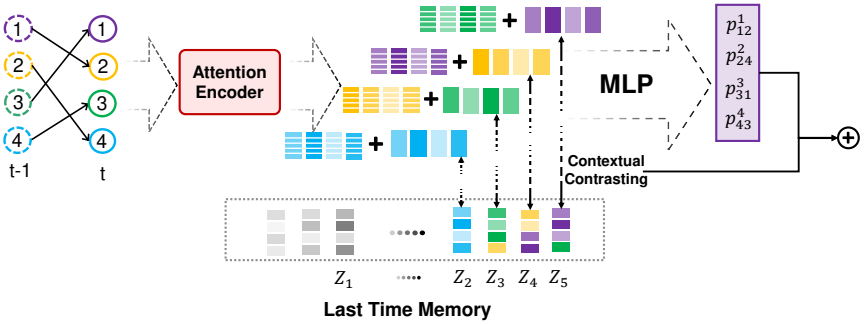

Fig. 2: Prposed general framework for a dynamic graph representation of temporal and context contrasting (DGTCC). Please note that we do not explicitly select negative sample pairs without interaction for comparison or use data enhancement to transform a sample into a sample of its kind in the temporal context contrast module. We treat the representation of the same nodes in the nearby temporal sub-graph as positive examples and only maximize the mutual information between positive samples.

interaction$\langle v_s, v_d, t\rangle$ by learning a mapping function f.

$$\mathbf{Z}_i(t) = \mathrm{emb}(i,t) = \sum_{d_i(0,\ t)} f\left(\ v_s(t), v_d(t)\right) \tag{1}$$

The computation of node v_i is obtained from the aggregation information of its l-hop temporal neighborhood neighbor through the graph attention layer embedded in the l layer.

$$\mathbf{Z}_i^l(t) = \mathbf{MLP}^l\left(\mathbf{Z}_i^{l-1}(t) \| \tilde{\mathbf{Z}}_i^l(t)\right) \tag{2}$$

$$\tilde{\mathbf{Z}}_i^l(t) = \ \mathbf{MultiHeadAttention}^{\ l}\left(\mathbf{Q}^l(t), \mathbf{K}^l(t), \mathbf{V}^l(t)\right) \tag{3}$$

$$\mathbf{Q}^l(t) = \mathbf{Z}_i^{l-1}(t) \tag{4}$$

$$\mathbf{K}^l(t) = \mathbf{V}^l(t) \tag{5}$$

$$\mathbf{V}^l(t) = \left[\mathbf{Z}_1^{l-1}(t) \,\|\mathbf{e}_{i1}\,(t_1)\|\, \boldsymbol{\phi}\,(t-t_1), \ldots, \mathbf{Z}_N^{l-1}(t) \,\|\mathbf{e}_{iN}\,(t_N)\|\, \boldsymbol{\phi}\,(t-t_N)\right] \tag{6}$$

The symbol || denotes the concatenation operation to link the source and destination nodes. The embedding of node v_i is obtained from the aggregation information of its $l-hop$ time neighborhood neighbor through the graph attention layer embedded in the l layer. Different from APAN, which uses positional encoding in order of the arrival of mails, we use a time coding function Φ and only care about the timespan $t - t_N$ between the two interactions, and each layer is equivalent to executing multiple attention layers, in which the query

$Q^l(t)$ is the query node (that is, the target node or its l-1 hop neighbor). The key K and value V are adjacent nodes.

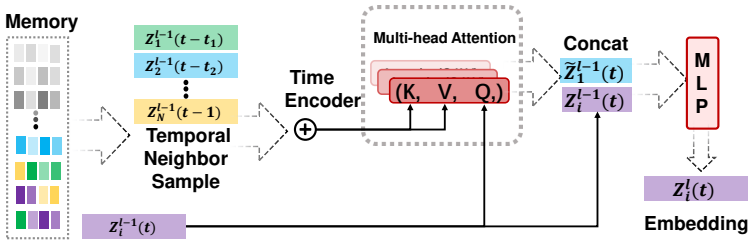

Fig. 3: The encoder network of DGTCC is a multi-head attention module. This attention module calculates the current node embedding $Z_i(t) \in R^d$ according to the relativity between the last updated embedding $Z_i^{l-1}(t) \in R^d$ and temporal neighbor information extracted from the memory module.

The attention model uses multi-head attention to form multiple sub-spaces so that the model can learn information from different angles. Finally, a layer of a multi-layer perceptron (MLP) is used to combine the reference node representation with aggregated information. Since some nodes may cease to be active for a while, the graph attention mechanism can better aggregate neighbor information based on node representation and time information.

3.2.2 Temporal Information Processing

Like Dyrep and Jodie, we introduce additional memory to store representations of adjacent nodes each time an interaction occurs. We use node-wise level node representation updates to model node representation in sequential dynamic graphs. The memory module remembers the long-term dependencies of each node in the graph. When a new node is encountered, the memory portion of the node is initialized to the 0 vectors. Then, it updates the model for each event involving that node. When nodes v_i and v_j have an interaction at time t, we calculate the message $\mathbf{m}_i(t)$ for each interacting node:

$$\mathbf{m}_i(t) = \mathrm{msg}_n\left(s_i\left(t^-\right), t, v_i(t)\right) \tag{7}$$

$s_i(t^-)$ is the memory representation of the last update of node i before time t. msg is a learnable message function. We use a simple MLP layer to extract information about the current interaction.

The best experimental effect of the CTDG model is to update the representation of nodes every time there is an interaction between nodes. However, due to efficiency issues, the batch size is usually not set to 1, so we add model training to the interaction of a batch simultaneously. For multiple interactions

on the same batch of internal nodes, we use the message aggregation function to summarize the interaction information to the node V_i in the current batch:

$$\mathbf{m}_i(t) = \mathbf{aggregate}(\mathrm{m}_i(t_k), ..., m_i(t_n)) \tag{8}$$

Message Aggregation. For the aggregation function, we can select the RNN model to store the information of multiple interactions of nodes v_i in the same batch, but to avoid additional computational overhead, it is similar to the mailbox mechanism of the continuous-time dynamic graph model TGN and APAN, When the node v_i in the same batch involves multiple interactions, we only compress all the information $m_i(t) \in R^{n*d}$ of the node v_i horizontally by $m_i(t) \in R^{1*d}$.

Memory Update. Every event involving the node itself updates the node's memory: $s_i(t) = mem$. For interaction events involving two nodes, v_i and v_j, the memory of both nodes is updated after the event occurs. Here, mem is a learnable memory update function, such as a recurrent neural network, i.e., a gate recurrent unit(GRU) or recurrent neural network(RNN).

3.2.3 Temporal Contextual Contrasting

Motivated by recent contrastive learning developments in visual representation learning[37], different graph augmentation strategies (i.e., node descent and edge perturbation) in static graph representation learning[38, 39] are proposed. However, in dynamic graph settings, individual representations usually evolve over time. A part of such methods[31] relies on the modeling accuracy of future interactions and could be vulnerable to noise in addition to maximizing the mutual information between the latent representations of interaction nodes in the future, ignoring characteristics of node evolution over time. Therefore, we instinctively assume that the evolution of node representation is a smooth process, meaning that node representation would not change greatly in the fine particle time slice rather than altering the original features of the original graph structure through perturbation. Based on the above assumptions, we propose a temporal contextual contrasting module. We use the current interaction sub-graph obtained by neighbor sampling, and the node representation after attention encoder $\{Z_1, Z_2, Z_3 \ldots, Z_n\}$, compared with the previous round of memory representation Z_i^{t-1}in the sub-graph.

As proved in [40], minimizing this training objective function is equivalent to maximizing the lower bound of mutual information between interactive nodes. We use a contrastive loss to distinguish the same node embedded with other nodes embedded in these two nearby temporal contextual views. We define $Sim(\cdot) \in R^{d\times d}$ as a discriminator function that takes two predictive representations as input and scores the consistency between them. θ is a cosine similarity and $g(\cdot)$ is a nonlinear projection to enhance the latent information of the node.

$$Sim\left(v_i, v_j\right) = \theta\left(g\left(v_i\right), g\left(v_j\right)\right) \tag{9}$$

For any node v_i, the embeddings generated in one view v_i^- are anchors, the embeddings generated in the other temporal view v_i are positive samples, and the embeddings of non-v_i nodes in the two views are naturally regarded as negative samples. We define the pairwise target of each pair (v_i^-, v_i) that generates an interaction as

$$\mathcal{L}_{tc}\left(\boldsymbol{v}_i^-, \boldsymbol{v}_i\right) = \log \frac{e^{Sim\left(v_i^-, v_i\right)/\tau}}{e^{Sim\left(v_i^-, v_i\right)/\tau} + \sum_{k=1}^{N} \mathbb{1}_{[k \neq i]} e^{Sim(v_i, v_k)/\tau} + \sum_{k=1}^{N} \mathbb{1}_{[k \neq i]} e^{Sim\left(v_i^-, v_k^-\right)/\tau}} \tag{10}$$

where $\mathbb{1}_{[}k \neq i] \in \{0, 1\}$ is an indicator function equal to 1 when k is not equal to 1 and τ is a temperature parameter. In our work, we do not explicitly sample negative nodes. Instead, given a positive pair, we naturally define the negative sample as all other nodes in the two temporal interactive node batch [t-n,t],[t,t+n]. The second term of the denominator in the corresponding formula (10) is the source-target node pairs without interaction within an inter-view window. The third term corresponds to the negative pairs between intra-view windows. The overall goal to maximize is then defined as the average of all positive pairs.

$$\mathcal{K} = \frac{1}{2N} \sum_{i=1}^{N} \left[\mathcal{L}_{tc}\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right) + \mathcal{L}_{tc}\left(\boldsymbol{v}_i, \boldsymbol{u}_i\right)\right] \tag{11}$$

3.2.4 MLP Decoder

For the modeling of a temporal dynamic graph representation, we use the link prediction between nodes as the downstream task of MLP layer. In link prediction, the goal is to predict the probability of an edge appearing between two nodes at a given timestamp. We use the node interaction generated before time $\{\langle S_1, S_2, 1\rangle, \langle S_2, S_3, 2\rangle \ldots\}$ as the model training input. After multi-layer GNN model calculation, a universal mapping function $\phi(\cdot)$ representing the link probability score $y_{u,v}$ is finally learned. We use t moments after the link probability prediction model of ultimate use $\left\{P_{1,2}^t, P_{2,3}^{t+1}, ...P_{i,j}^{t_n}\right\}$ as the predicted output for the link.

4 EXPERIMENTS

The results of an empirical evaluation of the benchmark dataset to validate the effectiveness of the proposed DGTCC framework is reported here. We conducted link prediction experiments on four real-world dynamic network datasets and explored the role of each component of DGTCC on individual node representation. Moreover, the sensitiveness of the temporal graph attention layer and self-supervised auxiliary parameter β were also analyzed to show the robustness of the proposed method.

Table 1: Statistics of the datasets used in the experiments.

	Wikepedia	Reddit	ML-10M	Yelp
Node	9227	11000	20537	6569
Edge	157474	672447	43760	95361
Edge features	172	172	-	-
Timespan	30 days	30 days	12 snapshots	15 snapshots
Data Spilt	70%-15%-15%	70%-15%-15%	70%-15%-15%	70%-15%-15%

4.1 Experimental setup

4.1.1 datasets

The four datasets come from different real-world scenarios, and all these networks consist of a sequence of edges/links with timestamps. The statistical information of these datasets is summarized in Table 1. Although Reddit and Wikipedia have initial node vectors, following the settings in [15], we assigned the same zero feature vector to all nodes. For link prediction tasks, to generate a set of links with labels for training and testing, we sorted the links by time (ascending) and selected the top 70% as training data, 15% as validation data, and the remaining 15% as testing data for all datasets.

Wikipedia[1]. Wikipedia is widely used in many SOTA dynamic graph modeling tasks. Nodes represent wiki users and wiki pages, and interactive edges represent user edit pages. The dynamic label indicates whether the user is prohibited from posting a dynamic label. We used data from 1,000 frequently edited pages and 9,227 active users on Wikipedia, and a total of 157,474 interactions were generated at different timestamps as experimental data.

Reddit[1]. The bipartite interaction graph of Reddit users collects one month of user interaction data, which has approximately 11,000 nodes and 700,000 edges. The interaction in Reddit refers to the interaction between the user and the subreddit by posting. The dynamic temporal tag indicates whether the user is prohibited from posting under the subreddit.

ML-10M. ML-10M[41] is a bipartite temporal graph of user-tag interactions that consists of user-tag interactions in which the links connect users with the tags they applied to certain movies.

Yelp[2]. Yelp is a user enterprise bipartite graph. The dynamic graph is composed of links between the user and businesses, and these links represent the user's observational ratings of the businesses over time.

4.1.2 Baselines

As a CTDG model, DGTCC updates the representation of the node whenever the interaction of the node occurs. We selected five dynamic graph representation methods and three static methods as DGTCC's main competitors. We introduced these baselines in detail in Section 2. Moreover, in Wikipedia and

[1] http://snap.stanford.edu/jodie
[2] https://www.yelp.com/dataset/challenge

Reddit datasets, the results of all baselines are strictly inherited from their original papers. We used the same data processing and splitting methods as the original paper. The baselines involved in the experiment are the following.

- **Node2vec**: Node2vec[21] is similar to DeepWalk, generating node sequences by a biased random-walk-based nodes sampling strategy. These sequences could be treated as text input skip-gram models to obtain vectors for each node.
- **GAT**: GAT[13] is a GNN-based graph representation learning method that proposes that the attention mechanisms assign different weights to different neighbor nodes.
- **Sage**: Sage[25] is an extended GNN that can learn the node representations for previously unseen data. It provides three aggregators: mean aggregator, long short-term memory (LSTM) aggregator, and pooling aggregator.
- **DySAT**: DySAT[26] is the first DTDG representation method that linearly extracts structural and temporal information based on the self-attention mechanism.
- **Jodie**: Jodie[31] is a representation learning framework that learns to forecast the embedding trajectories into the future to make predictions about the entities and their interactions.
- **Dyrep**: Dyrep[30] divides the change of graph structure into two processes: topological evolution and node interaction. Dyrep learns the topology evolution and activities between nodes, where the representation of node v_i is updated after an event involving v_i.
- **TGN**: TGN[15] combines the advantages of Jodie and TGAT and introduces node-level memory into the timing aggregation stage of TGAT.
- **APAN**: In APAN[16], an asynchronous mail propagation is introduced to decouple the graph query and calculation phases and use graph propagation to model the temporal graph structure. .

4.1.3 Parameter Settings

For all datasets, we used the Adam optimizer with a learning rate of 0.0001, a batch size of 200 for both training and testing, and early stopping with a patience of 5. When selecting a batch of interactive nodes, the same number of negative examples (not generated) link nodes were selected simultaneously to enhance the robustness of the model representation learning. The number of temporal layers was set to 2 and attention heads to 2. Note that our experiment setup closely followed those of TGN[15] and APAN [16]. The magnitude of the self-supervised task β was set to 0.25. In the experiments described next, we prove that the DGTCC method is not sensitive to hyper-parameters.

4.1.4 Evaluation Metrics

Because the model's performance under the inductive setting is more robust, it can reduce the impact of noise; we directly divided the unseen nodes in the test set into the test set and the verification set to evaluate the learning ability of

Table 2: Dynamic link prediction performance in AP and AUC. We use different colors to highlight the first and second best performing algorithms.

	Wikipedia		Reddit		ML-10M		Yelp	
	AP	AUC	AP	AUC	AP	AUC	AP	AUC
Node2vec	0.6905	0.6883	0.8034	0.8463	0.7538	0.7548	0.7905	0.7928
GAT	0.7401	0.7321	0.9060	0.8400	0.7727	0.8413	0.8366	0.7458
Sage	0.9109	0.9087	0.9627	0.9614	0.7824	0.7784	0.8463	0.8486
DySAT	0.9441	0.9244	0.9265	0.9470	0.8930	0.8610	0.8058	0.8379
Jodie	0.9322	0.9273	0.93485	0.9437	0.7837	0.7963	0.7398	0.7942
Dyrep	0.9712	0.9713	0.9709	0.9688	0.7512	0.7510	0.7904	0.8223
TGN	0.9786	0.9752	0.9847	0.9843	**0.8953**	**0.9123**	0.8827	0.9066
APAN	**0.9797**	**0.9802**	**0.9922**	**0.9879**	0.8602	0.8786	**0.8833**	**0.9092**
ours	**0.9824**	**0.9810**	**0.9832**	**0.9828**	**0.9190**	**0.9242**	**0.9131**	**0.9190**

the model. We used average precision (AP) and the area under the ROC curve (AUC) as evaluation indicators for the link prediction task, as was done for the traditional sequence graph representation task. Given a series of interactions generated in a time interval, we calculated the occurrence probability of node i, j interaction of each interaction and calculate AP and AUC based on the true label.

4.2 Performance Comparison

We evaluated our method and the baselines for the temporal link prediction task. All experiments were repeated five times, and average results are reported in Table 2, from which we made the following observations. (1)DGTCC consistently outperforms all the competitors on Wikipedia, ML-10M, and Yelp. According to the analysis, the frequency of nodes in the event stream of Reddit is relatively sparse compared with the other three datasets. Nodes tend to have their next interaction within a longer interval. Therefore, it is challenging to use contrasting learning to amplify mutual information of nodes in the interaction between nodes with a long time interval (inactive nodes). We can ignore the maximum retrospective duration by setting a time upper bound for the problem that nodes are not active in the dynamic graph representation task. (2)We noticed that DGTCC performs better than other baselines on the Yelp and ML-10M datasets compared to Wikipedia. By analyzing the datasets, 88.4% interactions are repeated in Wikipedia. A similar situation does not exist on ML-10M or Yelp because users would generally only review a movie or business once. Therefore, we can conclude that richer node neighborhoods would perform better when the node representation is represented as a high-dimensional vector. Meanwhile, richer neighbor nodes can also enable the contrasting learning to have multiple forms of representations between different positive samples, which help capture the evolution mode of nodes and prevent the phenomenon of model collapse[35].

4.3 Ablation Experiments

Impact of Different Model Components The superiority of DGTCC detailed in the preceding subsection can be seen due to the efficient modeling of the topology and the temporal evolution trend. To further study the contribution of each module in DGTCC, we developed two variants of DGTCC: DGTCC-T and DGTCC-I. DGTCC-T represents traditional self-supervised contrasting learning methods in a dynamic graph that only samples the interaction in the future temporal subgraph to determine whether it is a positive or negative sample. In contrast, DGTCC-I represents maximizing the mutual information between the latent representations of nodes which interval a temporal view, which is different from contrasting two nearby temporal views of the same node identity. We compared them to the full DGTCC on the ML-10M and Wikipedia datasets.

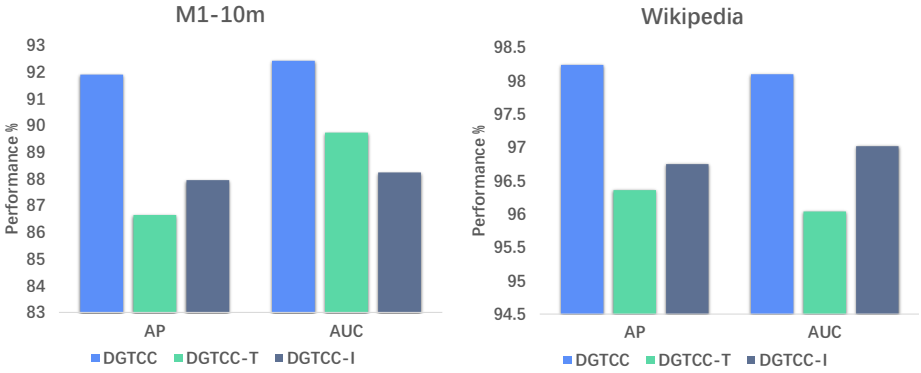

Fig. 4: Contribution of each component

As can be observed in Figure 4, the contributions of each component are different in the two datasets. We summarize our observations as follows: The result of maximizing the mutual information between two nearby temporal views of the same node identity is better than contrasting interval temporal views, which validates our assumption that coherence may be more important than graph modeling. We can also observe that DGTCC outperforms DGTCC-T by a large margin, demonstrating our optimization strategy's effectiveness in dynamic settings compared to traditional contrasting learning.

Impact of Self-Supervised Learning. Since β is an additional hyperparameter that we introduced to control the effects of auxiliary tasks, we attempted to evaluate how the size of the β in our temporal context module affects performance. We report the performance of DGTCC with a set of representative β values $\{0.001, 0.01, 0.02, 0.03, 0.05\}$. We observed that the DGTCC model only considered the interaction information of nodes in the temporal graph when learning the representation of the model when β was too small in most cases. However, the performance would be improved when β increases, and

the model can often take the similarity of nodes in temporal context information into better consideration when learning node representation. However, in some cases, over-constraining the user's representation changes can lead to performance degradation due to possible over-fitting problems.

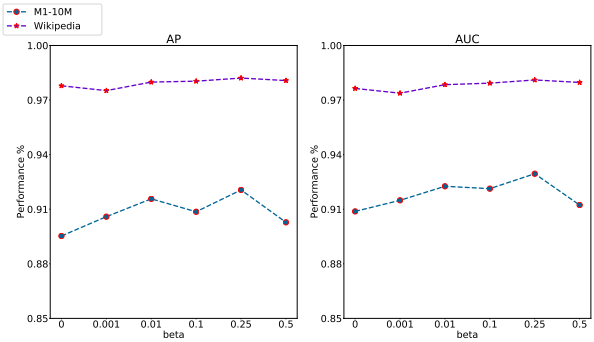

Fig. 5: Performance of DGTCC w.r.t different values of β.

Number of sampled graph attention layers. To study the impacts of the temporal graph attention layer, we ranged the number of layers of the network within $\{1, 2, 3, 4, 5\}$. According to the results presented in Figure 5, DGTCC is not very sensitive to the number of layers. Nevertheless, aggregating a large amount of historical neighbor information would introduce additional noise and reduce model performance, which has been similarly demonstrated in TGAT[14].

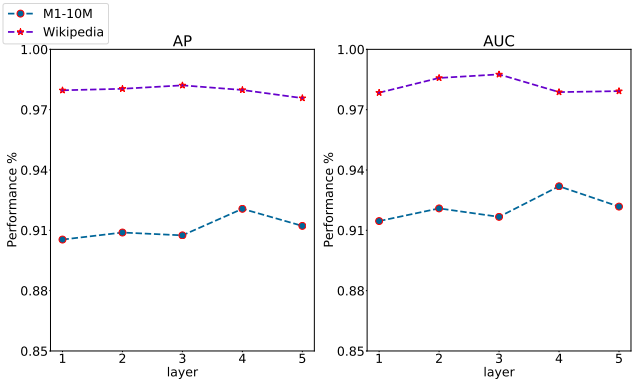

Fig. 6: Performance of DGTCC w.r.t different values of *layer*.

5 Conclusions

In this study, we propose a dynamic graph representation method based on temporal and context contrasting (DGTCC), a continuous-time dynamic graph framework for temporal graph representation. Inspired by the characteristic of time consistency in dynamic graph representation, we consider the trend that the changes between nodes over time always tend to be smooth. We introduce contrasting learning to build a temporal and context contrasting module and combine the topology information in the graph structure. As an auxiliary task, we only maximize the agreement of the same nodes in two temporal views without explicit negative sampling, which could capture the smoothing trend of node evolution and achieve better results. Experimental results of downstream tasks with link prediction demonstrate the superiority of the proposed model, and the ablation study results validate the effectiveness and rationale of the model parameters.

In the proof of [42], uniformity loss encourages the separation of independent samples in representation space. In traditional contrasting learning methods, if the concept of "uniformity" is not introduced, the features of all points are as evenly distributed as possible on the sphere. This would lead to the representation collapse of all nodes, while only the uniformity component will not generate any meaningful aggregation of nodes with similar characteristics. Eventually, each node is nearly the same, and the performance of the model decreases. Whether the same predictor module can be built without negative samples to bring the two networks closer to each other in the representation of a dynamic graph can be targeted in future research.

6 Declarations

Ethical Approval and Consent to participate
Not applicable.
Human and Animal Ethics
Not applicable.
Consent for publication
All authors have reviewed the final version of the manuscript and approved it for publication.
Availability of supporting data
These data were derived from the following resources available in the public domain: http://snap.stanford.edu/jodie/, https://www.yelp.com/dataset/challenge, https://doi.org/10.1145/2827872.
Competing interests
All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.
Funding
This work was supported by the Natural Science Foundation of Guangdong Province, China under Grant No. 2022A1515010148.

Authors' contributions
All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Jin Huang, Wentai Zhu and Jing Xiao. The first draft of the manuscript was written by Tian Lu and Weihao Yu and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Acknowledgements
This work was supported by the Natural Science Foundation of Guangdong Province, China under Grant No. 2022A1515010148.

Authors' information
Jin Huang, South China Normal University, huangjin@m.scnu.edu.cn;
Wentai Zhu, South China Normal University, 2020023065@m.scnu.edu.cn;
Jing Xiao, South China Normal University, xiaojing@scnu.edu.cn;
Tian Lu, South China Normal University, tianlu@m.scnu.edu.cn;
Weihao Yu, Research Institute of China Telecom Corporate Ltd., yuwh3@chinatelecom.cn.

References

[1] Kazemi, S.M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., Poupart, P.: Representation learning for dynamic graphs: A survey. J. Mach. Learn. Res. **21**(70), 1–73 (2020)

[2] Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data mining and knowledge discovery **29**(3), 626–688 (2015)

[3] Bongini, P., Bianchini, M., Scarselli, F.: Molecular generative graph neural networks for drug discovery. Neurocomputing **450**, 242–252 (2021)

[4] Fortunato, S.: Community detection in graphs. Physics reports **486**(3-5), 75–174 (2010)

[5] Divakaran, A., Mohan, A.: Temporal link prediction: A survey. New Generation Computing **38**(1), 213–258 (2020)

[6] Xue, G., Zhong, M., Li, J., Chen, J., Zhai, C., Kong, R.: Dynamic network embedding survey. Neurocomputing **472**, 212–223 (2022)

[7] Skardinga, J., Gabrys, B., Musial, K.: Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey. IEEE Access (2021)

[8] Fu, D., Zhou, D., He, J.: Local motif clustering on time-evolving graphs. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 390–400 (2020)

[9] Hou, C., Zhang, H., He, S., Tang, K.: Glodyne: Global topology preserving dynamic network embedding. IEEE Transactions on Knowledge and Data Engineering (2020)

[10] Chang, X., Liu, X., Wen, J., Li, S., Fang, Y., Song, L., Qi, Y.: Continuous-time dynamic graph learning via neural interaction processes. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 145–154 (2020)

[11] Zhou, D., Zheng, L., Han, J., He, J.: A data-driven graph generative model for temporal interaction networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 401–411 (2020)

[12] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems **29**, 3844–3852 (2016)

[13] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)

[14] Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., Achan, K.: Inductive representation learning on temporal graphs. arXiv preprint arXiv:2002.07962 (2020)

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

[16] Wang, X., Lyu, D., Li, M., Xia, Y., Yang, Q., Wang, X., Wang, X., Cui, P., Yang, Y., Sun, B., *et al.*: Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In: Proceedings of the 2021 International Conference on Management of Data, pp. 2628–2638 (2021)

[17] Zhang, J., Chen, K., Wang, Y.: Pre-training on dynamic graph neural networks. arXiv preprint arXiv:2102.12380 (2021)

[18] Wang, L., Chang, X., Li, S., Chu, Y., Li, H., Zhang, W., He, X., Song, L., Zhou, J., Yang, H.: Tcl: Transformer-based dynamic graph modelling via contrastive learning. arXiv preprint arXiv:2105.07944 (2021)

[19] Yu, W., Cheng, W., Aggarwal, C.C., Chen, H., Wang, W.: Link prediction with spatial and temporal consistency in dynamic networks. In: IJCAI, pp. 3343–3349 (2017)

[20] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social

representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)

[21] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)

[22] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077 (2015)

[23] Yi, Y., Wang, J., Zhou, W., Zheng, C., Kong, J., Qiao, S.: Non-negative matrix factorization with locality constrained adaptive graph. IEEE Transactions on circuits and systems for video technology **30**(2), 427–441 (2019)

[24] Li, J., Zhou, G., Qiu, Y., Wang, Y., Zhang, Y., Xie, S.: Deep graph regularized non-negative matrix factorization for multi-view clustering. Neurocomputing **390**, 108–116 (2020)

[25] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)

[26] Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: 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, pp. 519–527 (2020)

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

[28] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., Leiserson, C.: Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5363–5370 (2020)

[29] Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)

[30] Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: Learning representations over dynamic graphs. In: International Conference on Learning Representations (2019)

[31] Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1269–1278 (2019)

[32] Zhu, Q., Xiao, Y.: Adagnn: A multi-modal latent representation meta-learner for gnns based on adaboosting. arXiv preprint arXiv:2108.06452 (2021)

[33] Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. ICLR (Poster) **2**(3), 4 (2019)

[34] Tian, Y., Chen, X., Ganguli, S.: Understanding self-supervised learning dynamics without contrastive pairs. arXiv preprint arXiv:2102.06810 (2021)

[35] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., et al.: Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733 (2020)

[36] Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15750–15758 (2021)

[37] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning, pp. 1597–1607 (2020). PMLR

[38] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems **33**, 5812–5823 (2020)

[39] Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021, pp. 2069–2080 (2021)

[40] Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670 (2018)

[41] Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis) **5**(4), 1–19 (2015)

[42] Wang, T., Isola, P.: Understanding contrastive representation learning

through alignment and uniformity on the hypersphere. In: International Conference on Machine Learning, pp. 9929–9939 (2020). PMLR