2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Control-based Graph Embeddings with Data Augmentation for Contrastive Learning

Obaid Ullah Ahmad, Anwar Said, Mudassir Shabbir, Xenofon Koutsoukos, and Waseem Abbas

Abstract— **In this paper, we study the problem of unsupervised graph representation learning by harnessing the control properties of dynamical networks defined on graphs. Our approach introduces a novel framework for contrastive learning, a widely prevalent technique for unsupervised representation learning. A crucial step in contrastive learning is the creation of 'augmented' graphs from the input graphs. Though different from the original graphs, these augmented graphs retain the original graph's structural characteristics. Here, we propose a unique method for generating these augmented graphs by leveraging the control properties of networks. The core concept revolves around perturbing the original graph to create a new one while preserving the controllability properties specific to networks and graphs. Compared to the existing methods, we demonstrate that this innovative approach enhances the effectiveness of contrastive learning frameworks, leading to superior results regarding the accuracy of the classification tasks. The key innovation lies in our ability to decode the network structure using these control properties, opening new avenues for unsupervised graph representation learning.**

I. Introduction

Networks serve as fundamental data structures for representing relationships, connectivity, and interactions across various domains, such as social networks, biology, transportation, brain connectivity, and recommendation systems [1]. Network representation learning plays a pivotal role in acquiring meaningful network representations, which find applications in tasks like node classification, link prediction, and community detection [2]. Traditional network representation learning heavily relies on supervised learning, necessitating substantial labeled data for effective training [2]. However, obtaining labeled network data is often challenging, expensive, and limited in availability.

Contrarily, contrastive learning (CL) has emerged as a prominent self-supervised learning (SSL) technique in unsupervised network representation learning [3]. CL methods operate by comparing augmented positive and negative samples with the original graph. The positive samples exhibit similarity, while the negative samples manifest dissimilarity. This framework empowers CL methods to acquire representations that capture the inherent network structure, even when labeled data is absent [4]. Graph Contrastive Representation Learning has recently gained attention in the context of graph representation learning, aiming to maximize agreement between similar subgraphs and produce informative embeddings that capture the graph structure [5]. While existing GCRL approaches primarily focus on node-level embeddings [6], our proposed architecture has the potential to generate graph-level embeddings suitable for SSL.

In this work, we introduce a novel approach that leverages control properties to design graph-level embeddings for self-supervised learning. Recent research has uncovered deep connections between network controllability and various graph-theoretic constructs, including matching, graph distances, and zero forcing sets [7], [8]. Additionally, significant progress has been made in characterizing the controllability of different families of network graphs, including paths, cycles, random graphs, circulant graphs, and product graphs [8]. These investigations shed light on the interplay between network structures and their controllability properties, enhancing our understanding of network dynamics. Our aim is to explore and harness the interconnections between network structures and their controllability properties to form a foundation for comprehensive graph representations.

Furthermore, we introduce systematic graph augmentation for creating positive and negative pairs in CL, in contrast to previous random edge perturbation methods [5]. Our systematic approach focuses on preserving the graph's control properties, leading to improved performance in downstream machine-learning tasks.

Our main contributions can be summarized as follows:

- We introduce a novel graph embedding—representing graphs as vectors— called CTRL, which is based on the control properties of networks defined on graphs, including meaningful metrics of controllability such as the spectrum of the Gramian matrix.
- We present the Control-based Graph Contrastive Learning architecture for unsupervised representation learning of networks, applicable to various downstream graph-level tasks.
- We devise innovative augmentation techniques that mainly preserve the controllability of the network.
- We conduct extensive numerical evaluations on real-world graph datasets, showcasing the effectiveness of our method in graph classification compared to several state-of-the-art (SOTA) benchmark methods.

II. Preliminaries and Problem Statement

A. Notations

A network, represented as a graph $G = (V, E)$, consists of interconnected entities denoted by vertices $V = V(G) =$

This material is based upon work supported by the National Science Foundation under Grant Nos. 2325416 and 2325417.

Obaid Ullah Ahmad is with the Electrical Engineering Department at the University of Texas at Dallas, Richardson, TX. Email: Obaidullah.Ahmad@utdallas.edu.

Anwar Said, Mudassir Shabbir and Xenofon Koutsoukos are with the Computer Science Department at the Vanderbilt University, Nashville, TN. Emails: anwar.said,mudassir.shabbir, xenofon.koutsoukos@vanderbilt.edu.

Waseem Abbas is with the Systems Engineering Department at the University of Texas at Dallas, Richardson, TX. Email: waseem.abbas@utdallas.edu

$v_1, v_2, \ldots, v_N$, with edges $E = E(G) \subseteq V \times V$ indicating established relationships between entities. The terms 'vertex,' 'node,' and 'agent' are used interchangeably. The neighborhood of a vertex v_i is $\mathcal{N}_i = v_j \in V : (v_j, v_i) \in E$. The shortest path length between vertices v_i and v_j is denoted as $d(v_i, v_j)$. The transpose of a matrix X is denoted as X^T. A zero vector of dimension N is represented as $\mathbf{0}_N$, and a vector of all ones is denoted as $\mathbf{1}_N$. While our primary focus is on undirected graphs, all methods apply equally to directed graphs.

B. Problem Description

In this subsection, we tackle the task of unsupervised graph-level representation learning by introducing a contrastive learning-based approach. Graph embeddings, denoted as $\phi(G) : \mathcal{G} \to \mathbb{R}^d$, are designed to map graphs from the family $\mathcal{G}$ to a Euclidean space of dimension d. Given the scarcity of labeled data, our goal is to learn representations that capture both local and global structural similarities. Scalability is another critical consideration, ensuring that embeddings can accommodate graphs of varying sizes while effectively capturing underlying structural features. Our approach addresses these challenges in generating graph embeddings.

***Problem** 1:* Given a graph G, generate unsupervised graph representations $\phi(G)$ that capture essential structural information and node relationships for subsequent machine learning tasks.

These learned representations $\phi(G)$ are intended to be semantically meaningful and effective for various downstream tasks, including node classification, link prediction, graph classification, and community detection.

C. Proposed Approach

A typical method for learning unsupervised representations of raw data is self-supervised learning (SSL). Contrastive learning, a powerful technique within SSL, has demonstrated remarkable success across various domains, including computer vision [4]. In the field of graph representation learning, researchers have introduced Contrastive Graph Representation Learning (CGRL) [5]. This approach operates by generating diverse augmented perspectives of the same data samples through pretext tasks. We propose the introduction of a dynamical system on graphs to examine the control characteristics of this system, followed by crafting an embedding that utilizes these control attributes, as detailed in Section III.

a) Graph Contrastive Representation Learning: Graph Contrastive Representation Learning (GCRL) offers several advantages over traditional unsupervised graph representation methods. It encourages the model to bring similar nodes or subgraphs closer together while pushing dissimilar ones farther apart [5], thereby enhancing performance in various downstream tasks [9]. GCRL is data-efficient, scalable to large-scale datasets [5], and facilitates easy transfer to diverse tasks, producing interpretable embeddings that aid in the understanding and analysis of learned representations [9].

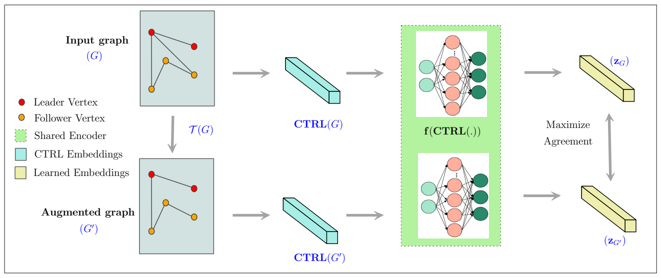

Fig. 1: Block diagram of the proposed CGCL approach

b) Control-based Graph Contrastive Learning (CGCL): We introduce Control-based Graph Contrastive Learning (CGCL), computing control-based graph-level features denoted as $CTRL(.)$. These features are used to bring an augmented version G' of a graph G closer together in a latent space $z(.)$ using the normalized temperature-scaled cross-entropy loss (NT-Xent) [3]. We propose augmentation techniques that preserve the $CTRL$ properties of the graph to a certain extent. This is illustrated in Figure 1.

For a given graph G, we apply a control-based augmentation $\mathcal{T}(G)$ to obtain G', creating a positive pair. We then compute control-based features $CTRL(.)$ for both G and G' and pass them through a learnable encoder $f(.)$ to transform them into a new latent space. The goal is to optimize the similarity of each positive pair in this latent space. This concept is illustrated in Figure 1, where the embeddings z_G and $z_{G'}$ are represented as yellow cuboids.

This optimization employs the NT-Xent loss by Oord et al. [3], encouraging similarity between the embeddings of the original graph and its transformed counterpart (positive pair) while minimizing similarity with transformed embeddings $z(.)$ of other graphs in the dataset (negative pairs). The loss function is:

$$\mathcal{L} = \mathbb{E}\left[-log\frac{exp(sim(z_G, z_{G'})/\tau)}{\frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}, g\neq G} exp(sim(z_G, z_{g'})/\tau)}\right],$$

Here, $sim(z_G, z_{G'})$ represents the cosine similarity between the embeddings of graph G and its augmentation G'. $\mathcal{G}$ denotes the set containing all graphs in the dataset, where g' represents the augmented version of graph g, and $\mathbb{E}$ denotes the expectation. The temperature hyperparameter τ controls the sharpness of the distribution.

In summary, CGCL utilizes contrastive learning principles to generate expressive graph representations by leveraging control-based features and optimizing the similarity of positive pairs. This highlights its potential for self-supervised graph representation learning.

III. Network Controllability and Graph Embeddings

In this section, we introduce a novel approach to graph embedding rooted in network control properties. We start by viewing a network as a controllable dynamic system and define network controllability. We then explore metrics used to measure it, forming the basis for constructing control-based graph embeddings denoted as $CTRL(G)$.

A. Networks as Dynamical Systems

In network dynamics, each agent v_i has a state $x_i(t) \in \mathbb{R}$ at time t, sharing states with neighbors in $\mathcal{N}_i$. Agents update states based on dynamics like consensus. The system state at t is $x(t) = [x_1(t)\ x_2(t)\ \ldots\ x_N(t)]^T$.

To control the system, we apply external signals to a subset of agents called *leaders*. Leader states can be directly manipulated ($\dot{x}_l = u_l(t)$), while *followers* update states based on local information. Follower dynamics ($\dot{x}_f(t)$) in this leader-follower system are given by:

$$\dot{x}_f(t) = -\mathcal{M}(G)x_f(t) + \mathcal{H}(G)u(t),$$

where $\mathcal{M}(G)$ represents follower subgraph matrices, and $\mathcal{H}(G)$ denotes leader-follower interactions using the Laplacian matrix.

In a graph $G = (V, E)$, nodes V are partitioned into followers (V_f) and leaders (V_ℓ). The follower graph (G_f) is represented by the Laplacian L partitioned as:

$$L = \left[\begin{array}{c|c} A & B \\ \hline B^T & C \end{array}\right],$$

with A, B, and C of appropriate dimensions.

An external signal u_l is applied to leader $v_l \in V_\ell$. Follower states update according to:

$$\dot{\boldsymbol{x}}_f(t) = -A\boldsymbol{x_f}(t) - Bu(t),$$

where $\boldsymbol{x_f}(t)$ is the follower state vector and $u(t)$ is the control signal. The matrices $-A$ and $-B$ are derived from network structure and leader selection.

We analyze reachable state subspace dimensionality and the influence of leader agent variation, offering insights into graph structure for deriving controllability metrics in graph embeddings.

B. Network Controllability Metrics

Controlling a network involves directing it from an initial state to a desired final state by applying control inputs to specific leader nodes. A state $\boldsymbol{x_f^*} \in \mathbb{R}^{N_f}$ is considered reachable if an input can propel the network from the origin to $\boldsymbol{x_f^*}$ within a finite timeframe, defining the controllable subspace. The dimension of this subspace is determined by the rank of the *Controllability matrix*: $\mathcal{C} = \left[\begin{array}{cccc} -B & (-A)(-B) & \cdots & (-A)^{N_f-1}(-B) \end{array}\right]$. The rank of this matrix hinges on the properties of matrices A and B, which, in turn, depend on the network's structure and the selection of leader nodes.

The Controllability Gramian $\mathcal{W}$ quantitatively assesses the transitioning between states, derived from the system dynamics. For the system delineated in equation (III-A), the *infinite horizon controllability Gramian* is defined as follows:

$$\mathcal{W} = \int_0^\infty e^{-A\tau}(-B)(-B)^T e^{-A^T\tau} d\tau; \in; \mathbb{R}^{N_f \times N_f}.$$

When partitioning the Laplacian matrix L of an undirected connected graph, as demonstrated in equation (III-A), the matrix A is revealed to be positive definite, ensuring the system's stability [8]. This stability enables the computation of the Controllability Gramian $\mathcal{W}$, which serves as a valuable measure of controllability in terms of energy-related quantification. It also facilitates the derivation of various controllability statistics [10]–[12], including the trace, minimum value, rank, and determinant of $\mathcal{W}$.

The trace of $\mathcal{W}$, $\mathbf{tr}(\mathcal{W})$, inversely relates to average control energy, indicating overall controllability. The minimum eigenvalue $\mu_j(\mathcal{W})$ reflects worst-case control energy needs. $\mathbf{rank}(\mathcal{W})$ corresponds to controllable subspace dimension, while $\mathbf{ld}(\mathcal{W})$ assesses volume accessible with minimal control energy. Refer to [10], [11], [13] for details. An example in [14] illustrates network controllability dependency on topology and leader placement. In our evaluation (Section V), we vary leader nodes to observe controllability, quantified by these metrics.

IV. Control-Based Graph Augmentations

Contrastive Graph Representation Learning (CGRL) is a self-supervised technique utilizing augmented data to create positive and negative pairs. Graph-level control embeddings serve as inputs to the encoder, minimizing the NT-Xent loss [3]. In this section, we introduce innovative methods for data augmentation.

The primary purpose of data augmentation is to generate logically consistent new data while preserving semantic labels. CGRL integrates a contrastive module into the Graph Machine Learning (GML) architecture, employing contrastive loss for fine-tuning models. Positive pairs include original and augmented graphs, while negative pairs involve other augmented graphs. CGRL aims to maximize similarity among positive pairs and dissimilarity among negative pairs during training. Given our embeddings' reliance on control properties, our augmentation technique prioritizes preserving these properties in the augmented graphs.

Problem IV.1: Given a graph $G = (V, E)$ and a leader set V_ℓ, perform an augmentation $\mathcal{T}(G)$ to obtain G' by perturbing k edges while ensuring that the controllability properties are preserved.

In this context, *edge perturbation* can refer to actions such as edge deletion, edge addition, or edge substitution.

During augmentation, our focus is on preserving the rank of controllability, denoted as $\gamma(G, V_\ell)$, in our CTRL embedding. However, exact preservation of this rank is complex, leading to exploration of lower bounds on network controllability in the literature [7], [15]. Our edge perturbation techniques employ a rigorous lower bound based on topological node distances and introduce algorithms to maintain this lower bound during network augmentation [7].

Assuming the presence of m leaders denoted as $V_\ell = \{\ell_1, \ell_2, \cdots, \ell_m\}$ within a leader-follower network $G = (V, E)$, we define the *distance-to-leader (DL) vector* for each vertex $v_i \in V$ as follows:

$$D_i = \left[\begin{array}{cccc} d(\ell_1, v_i) & d(\ell_2, v_i) & \cdots & d(\ell_m, v_i) \end{array}\right]^T \in \mathbb{Z}^m.$$

In this vector, the j^{th} component, denoted as $[D_i]_j$, represents the distance between leader ℓ_j and vertex v_i. We then proceed to define a *sequence* of distance-to-leader vectors, referred to as a *pseudo-monotonically increasing sequence* (PMI), as described in [7].

Definition (*Pseudo-monotonically Increasing Sequence (PMI))* A sequence $\mathcal{D} = [\mathcal{D}_1 \ \ \mathcal{D}_2 \ \ \cdots \ \mathcal{D}_k]$ of distance-to-leader vectors is a PMI if, for any vector $\mathcal{D}_i$ in the sequence, there exists a coordinate $\pi(i) \in \{1, 2, \cdots, m\}$ such that

$$[\mathcal{D}_i]_{\pi(i)} < [\mathcal{D}_j]_{\pi(i)}, \ \forall j > i.$$

In essence, the PMI property (IV) ensures that for each vector $\mathcal{D}_i$ in the PMI sequence, there is an index/coordinate $\pi(i)$ such that the values of all subsequent vectors at the coordinate $\pi(i)$ are strictly greater than $[\mathcal{D}_i]_{\pi(i)}$.

The length of the PMI sequence provides a precise lower bound on the dimension of the controllable subspace $\gamma(G, V_\ell)$. This is presented in the subsequent result.

Theorem 4.1: [7] If we denote the length of the longest PMI sequence of DL vectors in a network G with V_ℓ leaders as $\delta(G, V_\ell)$ or simply $\delta(G)$, then we establish the inequality: $\delta(G, V_\ell) \leq \gamma(G, V_\ell)$, where $\gamma(G, V_\ell)$ represents the dimension of the controllable subspace.

We propose the following three sophisticatedly designed edge perturbation methods that maintain the lower bound $\delta(G, V_\ell)$ on the rank of controllability. They are illustrated in Figure 2. The red vertices represent leaders, the gray dashed edge can be removed, and the blue dashed edge can be added while ensuring that the bound $\delta(G, V_\ell) = N_f = 4$ is maintained for all augmented graphs.

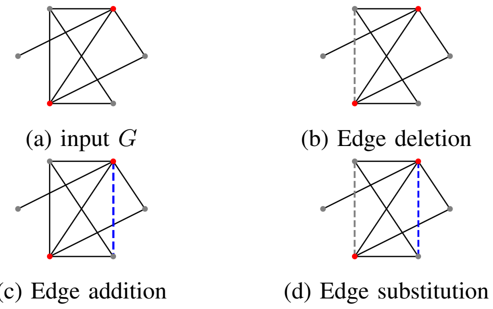

Fig. 2: Control-based graph augmentations where $\delta = \gamma = 4$ for original and augmented graphs.

A. Egde Deletion

We propose leveraging controllability backbone edges, introduced in [16], which preserve $\delta(G, V_\ell)$. We introduce the distance-based controllability backbone and present an algorithm for augmenting a graph with k-perturbed edges while incorporating the controllability backbone.

Definition (*Controllability Backbone*) [16] For a given graph $G = (V, E)$ and a set of leaders V_ℓ, the controllability backbone is represented as a subgraph $B = (V, E_B)$. In this subgraph B, the condition $\delta(G, V_\ell) \leq \delta(\hat{G}, V_\ell)$. holds for any subgraph $\hat{G} = (V, \hat{E})$ where the edge set satisfies $E_B \subseteq \hat{E} \subseteq E$.

Algorithm 1 outlines the graph augmentation process via edge deletion. Initially, we identify and preserve crucial edges from the controllability backbone to maintain the controllability bound, as computed using Algorithm 2 in [16]. Next, we identify a set of candidate edges in the original graph G that excludes those from the backbone graph B. Finally, we randomly remove k edges from the candidate set in G to obtain the augmented graph G'. If k exceeds the candidate set size, we delete all potential edges.

Algorithm 1 *Edge_Deletion*

```
Input: $G = (V, E)$, $V_\ell$, $k$
Output: $G' = (V, E')$, $|E| - |E'| = k$
1: Compute the distance-based controllability backbone
   $B = (V, E_B)$ for $G = (V, E)$ and $V_\ell$.
2: $pot\_edges \leftarrow E \setminus E_B$ % Set of potential
   edges.
3: $E_{pot} \leftarrow$ randomly selected $k$ edges from $pot\_edges$
4: $E' \leftarrow E \setminus E_{pot}$
5: return $G' = (V, E')$
```

Proposition 4.2: Given a graph $G = (V, E)$ and a leader set V_ℓ, Algorithm 1 returns an augmented graph $G' = (V, E')$, where $E' \subseteq E$, while ensuring $\delta(G, V_\ell) \leq \delta(G', V_\ell)$.

Proof: Available in [14]. ■

B. Egde Addition

Expanding on our prior work [17] regarding identifying removable edges while preserving the distance-based bound $\delta(G, V_\ell)$, we utilize an augmentation technique to identify edges that can be added to a given graph $G = (V, E)$ while maintaining $\delta(G, V_\ell)$. Following the approach outlined in [17], we identify potential edges that can be added to G without compromising $\delta(G, V_\ell)$. Subsequently, we randomly select k such potential edges and introduce them into G, resulting in the augmented graph $G' = (V, E')$.

Proposition 4.3: [17] If $\delta(G, V_\ell)$ is a lower bound for the dimension of the controllable subspace $\gamma(G, V_\ell)$ of a graph $G = (V, E)$ with leaders $V_\ell \subset V$, then it also serves as a lower bound for $\gamma(G', V_\ell)$ of an augmented graph $G' = (V, E')$, where $E \subseteq E'$ and E' contains edges preserving distances of DL vectors in the longest PMI sequence of G.

C. Egde Substitution

Next, we propose a novel approach that combines the edge deletion and edge addition methods while preserving the size of the edge set $|E|$ of the given graph $G = (V, E)$ and the bound $\gamma(G, V_\ell)$. Algorithm 2 outlines this approach. First, we remove k edges from G to create $\bar{G} = (V, \bar{E})$ using Algorithm 1. Then, we introduce k distinct edges into $\bar{G}$ using the method described in IV-B, resulting in $G' = (V, E')$, where $|E| = |E'|$.

The maximal edge set E_{max} can be computed by from Algorithm 1 of our previous work [17].

Proposition 4.4: Given a graph $G = (V, E)$ and a leader set V_ℓ, Algorithm 2 yields an augmented graph $G' = (V, E')$, where $|E'| = |E|$, ensuring that $\delta(G, V_\ell) \leq \delta(G', V_\ell)$,

Proof: Available in [14]. ■

These edge perturbation methods are employed to create positive pairs. Subsequently, we use the CTRL embeddings to generate graph representations for these pairs and apply the NT_Xent loss for unsupervised learning of representations for each graph within the dataset. In the following

Algorithm 2 *Edge_Substitution*

```
Input: $G = (V, E)$, $V_\ell$, $k$
Output: $G' = (V, E')$
1: $\bar{G} = (V, \bar{E}) \leftarrow Edge\_Deletion(G, V_\ell, k)$
2: Compute the maximal edge set $E_{max}$ for $G = (V, E)$ and $V_\ell$.
3: $pot\_edges \leftarrow E_{max} - E$ % Set of potential edges.
4: $E_{pot} \leftarrow$ randomly selected $k$ edges from $pot\_edges$
5: $E' \leftarrow \bar{E} \cup E'$.
6: return $G' = (V, E')$
```

section, we conduct an empirical assessment using real-world graph datasets. Our proposed approach is numerically evaluated through the task of graph classification and compared with state-of-the-art methods.

V. Numerical Evaluation

A. Benchmark Datasets

Datasets: We conducted experiments on 7 standard graph classification benchmark datasets, which include MUTAG, PTC_MR, PROTEINS, and DD, representing bioinformatics datasets, as well as IMDB-BINARY, IMDB-MULTI, and COLLAB, representing social network datasets [18]. The bioinformatics datasets provide descriptions of small molecules and chemical compounds. Among the social network datasets, IMDB-BINARY and IMDB-MULTI describe actors' ego-networks, while COLLAB is a scientific collaboration dataset where graphs consist of researchers as nodes and their collaborations as edges. Basic dataset statistics are provided in Table 1 of [14].

B. The Role of Data Augmentation in Graph CL

We evaluate the effectiveness of our proposed framework for graph classification using the TUDataset benchmark [18]. We utilize the CL method to unsupervisedly learn representations $z(.)$ from CTRL embeddings, followed by the evaluation of these representations for graph-level classification. This evaluation involves training and testing a linear SVM classifier using the acquired representations. We employ a 10% label rate and 10-fold cross-validation, conducting experiments over 5 repetitions and reporting the evaluation accuracy as a mean value along with the standard deviation.

As a baseline reference, we directly employ the CTRL embeddings for training the SVM classifier. The results for four distinct bioinformatics datasets are summarized in Table I. Edge deletion yields the best result on MUTAG and PTC, while on PROTEINS and DD, edge addition and substitution provide the best results, respectively. We vary the number of edges perturbed from 1 to 3. Our augmentation techniques, combined with contrastive learning, consistently yield higher classification accuracies across all datasets compared to the baseline approach.

C. Comparison with the State-of-the-art Methods

The effectiveness of CGCL is assessed in the context of unsupervised representation learning, following the approach

TABLE I: Graph Classification accuracy (%) with 10% label rate. The baseline involves a linear SVM trained directly on CTRL embeddings.

Method	MUTAG	PTC	PROTEINS	DD
Baseline	75.86 ± 11.0	52.85 ± 9.5	58.72 ± 11.9	59.10 ± 13.9
CGCL	79.54 ± 11.0	56.10 ± 8.3	69.97 ± 4.5	63.22 ± 11.1

outlined in [9], [19]. We closely adhere to the established approach within Graph CL for graph classification [5], [9]. We compare our results with various kernel-based, unsupervised, and self-supervised methods. For *graph kernel-based* methods, we consider Graphlet kernel (GK) [20], Weisfeiler-Lehman sub-tree kernel (WL) [21], and deep graph kernels (DGK) [22]. *Unsupervised* techniques include node2vec [23], sub2vec [24], and graph2vec [19], while *self-supervised* methods consist of InfoGraph [9] and GraphCL [5], both employing graph neural networks.

The results of graph classification are presented in Table II. When compared to the top-performing unsupervised methods, our proposed approach exhibits significant improvements across several datasets, including MUTAG, PTC, PROTEINS, COLLAB, and IMDB-B, resulting in gains of 6.76%, 6.17%, 0.83%, 13.0%, and 1.6%, respectively. Notably, our method consistently outperforms all unsupervised competitors across all datasets except IMDB-M.

In contrast to self-supervised counterparts, our proposed method surpasses the current SOTA on MUTAG, PROTEINS, and COLLAB, and achieves the second-best accuracies on DD and IMDB-B datasets. Specifically, in MUTAG, PROTEINS, and COLLAB, our approach outperforms the existing standards by margins of 0.90%, 4.64%, and 3.74%, respectively. However, on PROTEINS and IMDB-M, our method falls short by 0.31% and 1.17%, respectively, compared to the best self-supervised approaches.

In summary, as demonstrated by Table II, our proposed method outperforms its competitors on two out of seven graph classification datasets by a considerable margin and achieves top-two accuracy rankings for five out of seven datasets. For the remaining two datasets, our method achieves accuracy levels within 2% of the SOTA methods.

We also perform an evaluation of our proposed CGCL approach using the edge augmentation method proposed by You [5]. We follow an i.i.d. uniform distribution to add/drop edges instead of the systematic edge perturbation methods mentioned in Section IV. We call this approach Random-CGCL. Table III presents the results for graph classification accuracies for all seven datasets under consideration. It can be seen that the accuracy of Random-CGCL is comparable to both GraphCL and CGCL. However, CGCL outperforms Random-CGCL on all the datasets. These results suggest that a sophisticated augmentation technique, as employed in CGCL, is essential for effectively leveraging control-based embeddings in graph contrastive learning.

VI. Conclusion

We introduced Control-based Graph Contrastive Learning (CGCL), a novel framework for unsupervised graph representation learning that leverages graph controllability prop-

TABLE II: Comparing classification accuracy on top of graph representations learned from graph kernels, SOTA representation learning method. The top two results are highlighted by First, Second. The numerical values presented for comparison are obtained from the respective papers, following the identical experimental configurations.

Methods	MUTAG	PTC	PROTEINS	DD	COLLAB	IMDB-B	IMDB-M
Kernel Approaches							
GK	81.70 ± 2.1	57.30 ± 1.4	-	-	72.80 ± 0.3	65.90 ± 1.0	43.90 ± 0.4
WL	80.63 ± 3.1	56.91 ± 2.8	72.92 ± 0.6	-	78.90 ± 1.9	72.30 ± 3.4	47.00 ± 0.5
DGK	87.44 ± 2.7	60.10 ± 2.6	73.30 ± 0.8	-	-	66.96 ± 0.6	44.60 ± 0.5
Unsupervised Approaches							
node2vec	72.63 ± 10.2	58.85 ± 8.0	57.48 ± 3.6	-	55.70 ± 0.2	50.20 ± 0.9	36.0 ± 0.7
sub2vec	61.05 ± 15.8	59.99 ± 6.4	53.03 ± 5.6	-	62.10 ± 1.4	55.26 ± 1.5	36.7 ± 0.8
graph2vec	83.15 ± 9.2	60.17 ± 6.9	73.30 ± 2.1	-	59.90 ± 0.0	71.10 ± 0.5	50.40 ± 0.9
Self-Supervised Approaches							
InfoGraph	89.01 ± 1.1	61.70 ± 1.4	74.44 ± 0.3	72.85 ± 1.8	70.65 ± 1.1	73.03 ± 0.9	49.70 ± 0.5
GraphCL	86.80 ± 1.3	61.30 ± 2.1	74.39 ± 0.5	78.62 ± 0.4	71.36 ± 1.2	71.14 ± 0.4	48.58 ± 0.7
CGCL	89.91 ± 6.4	66.34 ± 7.9	74.13 ± 2.8	75.33 ± 3.3	75.10 ± 1.8	72.70 ± 4.4	48.53 ± 3.1

TABLE III: Graph Classification accuracies using different Augmentation methods. The top accuracies are **highlighted**.

Methods	MUTAG	PTC	PROTEINS	DD	COLLAB	IMDB-B	IMDB-M
GraphCL	86.80 ± 1.3	61.30 ± 2.1	**74.39 ± 0.5**	**78.62 ± 0.4**	71.36 ± 1.2	71.14 ± 0.4	**48.58 ± 0.7**
Random-CGCL	87.81 ± 7.4	65.73 ± 6.6	73.23 ± 1.8	75.15 ± 2.9	75.04 ± 1.8	71.40 ± 4.0	47.40 ± 2.7
CGCL	**89.91 ± 6.4**	**66.34 ± 7.9**	74.13 ± 2.8	75.33 ± 3.2	**75.10 ± 1.8**	**72.70 ± 4.4**	48.53 ± 3.1

erties. Using advanced edge augmentation methods, we created augmented data while preserving graph controllability. Extensive experiments on graph classification benchmarks demonstrated CGCL's effectiveness, outperforming state-of-the-art methods. Incorporating domain-specific structural knowledge, like controllability, can significantly enhance graph representation learning. CGCL presents a promising approach for applications requiring informative graph representations. Future work will focus on refining graph augmentation techniques to preserve all relevant control features.

References

[1] M. Newman, *Networks*. Oxford University Press, 2018.

[2] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[3] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[5] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in neural information processing systems*, vol. 33, pp. 5812–5823, 2020.

[6] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang, "Hypergraph contrastive collaborative filtering," in *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 70–79.

[7] A. Yazıcıoğlu, W. Abbas, and M. Egerstedt, "Graph distances and controllability of networks," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 4125–4130, 2016.

[8] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.

[9] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *arXiv:1908.01000*, 2019.

[10] F. Pasqualetti, S. Zampieri, and F. Bullo, "Controllability metrics, limitations and algorithms for complex networks," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 40–52, 2014.

[11] T. H. Summers, F. L. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 91–101, 2015.

[12] E. Wu-Yan, R. F. Betzel, E. Tang, S. Gu, F. Pasqualetti, and D. S. Bassett, "Benchmarking measures of network controllability on canonical graph models," *Journal of Nonlinear Science*, pp. 1–39, 2018.

[13] A. Said, O. U. Ahmad, W. Abbas, M. Shabbir, and X. Koutsoukos, "Network controllability perspectives on graph representation," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[14] O. U. Ahmad, A. Said, M. Shabbir, X. Koutsoukos, and W. Abbas, "Control-based graph embeddings with data augmentation for contrastive learning," *arXiv:2403.04923*, 2024.

[15] A. Chapman and M. Mesbahi, "On strong structural controllability of networked systems: A constrained matching approach," in *American Control Conference*, 2013, pp. 6126–6131.

[16] O. U. Ahmad, M. Shabbir, and W. Abbas, "Controllability backbone in networks," in *IEEE Conference on Decision and Control (CDC)*, 2023, pp. 2439–2444.

[17] W. Abbas, M. Shabbir, H. Jaleel, and X. Koutsoukos, "Improving network robustness through edge augmentation while preserving strong structural controllability," in *American Control Conference (ACC)*, 2020, pp. 2544–2549.

[18] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

[19] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *arXiv preprint arXiv:1707.05005*, 2017.

[20] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 488–495.

[21] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[22] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and data mining*, 2015, pp. 1365–1374.

[23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[24] B. Adhikari, Y. Zhang, N. Ramakrishnan, and B. A. Prakash, "Sub2vec: Feature learning for subgraphs," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2018, pp. 170–182.