Control-based Graph Embeddings with Data Augmentation for Contrastive Learning

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

***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], [2]. 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 [3]. Traditional network representation learning heavily relies on supervised learning, necessitating substantial labeled data for effective training [4]. 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 [5]. 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 [6]. 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 [7]. While existing GCRL approaches primarily focus on node-level embeddings [8], 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 [9]–[11]. 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 [11]. 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 [7]. 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 composed of interconnected entities is represented as a graph, denoted as $G = (V, E)$, where the set of vertices $V = V(G) = v_1, v_2, \ldots, v_N$ represents the

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

entities, and the collection of edges $E = E(G) \subseteq V \times V$ represents pairs of entities with established relationships. In this context, the terms 'vertex,' 'node,' and 'agent' are used interchangeably. The neighborhood of a vertex v_i is defined as $\mathcal{N}_i = v_j \in V : (v_j, v_i) \in E$. The *distance* between vertices v_i and v_j, denoted as $d(v_i, v_j)$, represents the shortest path length between them. The transpose of a matrix X is denoted as X^T. An N-dimensional vector with all its entries set to zero is represented as $\mathbf{0}_N$, and a vector with all its entries set to 1 is denoted as $\mathbf{1}_N$. Although our primary focus is on undirected graphs for simplicity in explanation, it is important to note that all methods and findings are equally applicable to directed graphs.

B. Problem Description

In this subsection, we introduce the problem of generating graph-level representations in an unsupervised manner. We begin by providing a formal definition of the problem and then present a contrastive learning-based solution. For graph-level embeddings, we develop a control-based graph embedding method in Section III

A graph embedding, denoted as $\phi(G) : \mathcal{G} \rightarrow \mathbb{R}^d$, maps graphs from the family $\mathcal{G}$ to a Euclidean space of dimension d. The main goal of graph embedding is to meet specific design criteria. Due to the limited availability of labeled data for large real-world benchmark datasets, we need a mechanism to learn graph representations without heavy reliance on labels. One crucial objective is to ensure that ϕ retains information about structural similarities between pairs of graphs, both at local and global scales. This means that if two graphs share structural similarities, their embeddings should yield vectors that are close to the target vector space, as measured by Euclidean distance. It's important to note that the notion of similarity between graphs depends on the specific application and the type of graphs being considered, such as chemical compounds or social networks.

Another crucial design objective for graph embeddings is scalability. An optimal graph embedding should not only map graphs of different sizes to a fixed-dimensional space but should also transcend graph size to capture underlying structural characteristics. For example, an effective graph embedding would position the mapping of a ten-node circulant graph closer to that of a twenty-node circulant graph compared to the mapping of a fifteen-node wheel graph. In this paper, we tackle the challenge of generating graph embeddings while adhering to these design objectives.

***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 approach to learning unsupervised representation of raw data is called self-supervised learning (SSL). One of the powerful techniques of SSL is contrastive learning which has achieved remarkable success across various domains, including computer vision [6]. In the realm of graph representation learning, researchers have introduced Contrastive Graph Representation Learning (CGRL) [12]. This approach operates on the principle of generating diverse augmented perspectives of the same data samples through pretext tasks. We propose to introduce a dynamical system on graphs, examining the control characteristics of this system, and subsequently crafting an embedding that utilizes these control attributes, as elaborated in Section III.

a) Graph Contrastive Representation Learning: Graph Contrastive Representation Learning (GCRL) offers distinct advantages over traditional unsupervised graph representation methods. GCRL encourages the model to bring similar nodes or subgraphs closer together in the embedding space while pushing dissimilar nodes or subgraphs farther apart [7], enhancing performance in various downstream tasks [13]. It excels in data efficiency, leveraging limited labeled data efficiently with unlabeled data sources [7]. GCRL is scalable, handling large-scale graph datasets effectively [7], [14]. It facilitates easy transfer of learned representations to diverse downstream tasks, including node classification, link prediction, and graph classification [7]. Lastly, GCRL often produces interpretable embeddings, aiding in the understanding and analysis of the learned representations [13].

b) Control-based Graph Contrastive Learning (CGCL): We introduce Control-based Graph Contrastive Learning (CGCL), which computes control-based graph-level features denoted as $CTRL(.)$. These features are utilized 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) [5]. Additionally, we propose various augmentation techniques designed to 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 is achieved using the NT-Xent loss proposed by Oord et al. [5]. The loss function encourages the similarity between the embeddings of the original graph and its transformed counterpart (positive pair) while minimizing the similarity with the transformed embeddings $z(.)$ of other graphs in the dataset (negative pairs). This self-supervised learning approach enables the model to capture meaningful representations effectively. The NT-Xent loss is employed as:

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

where $sim(z_G, z_{G'})$ represents the cosine similarity between the embeddings of the graph G and its augmentation G', $\mathcal{G}$ is the set containing all the graphs in the dataset, g' is the

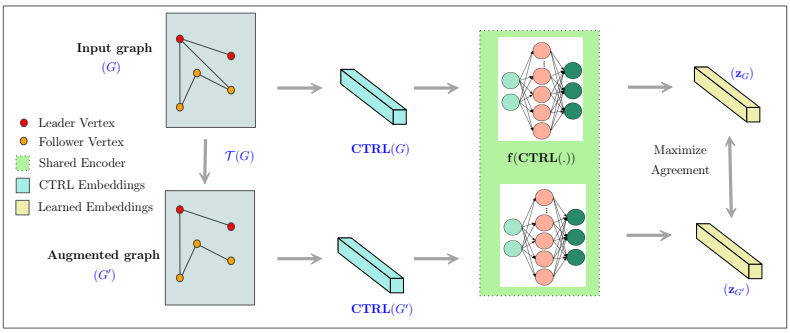


Fig. 1: Block diagram of the proposed CGCL approach

augmented version of graph g, and $\mathbb{E}$ is the expectation. The temperature hyperparameter τ is used to control the sharpness of the distribution.

In summary, CGCL leverages contrastive learning principles to generate expressive graph representations by considering control-based graph-level features and optimizing the similarity of positive pairs, demonstrating its potential for self-supervised graph representation learning.

III. Network Controllability and Graph Embeddings

In this section, we introduce a novel graph embedding approach rooted in the control properties of networks. We begin by presenting a network as a controllable dynamic system. We then provide a formal definition of network controllability and subsequently explore several metrics employed to quantify it. These metrics serve as the foundation for constructing control-based graph embeddings, denoted as $CTRL(G)$, for a given graph G [15].

A. Networks as Dynamical Systems

In the context of network dynamics, each agent, denoted as v_i, represents a dynamic unit with a *state* $x_i(t) \in \mathbb{R}$ at time t. These agents share their states with neighboring agents in $\mathcal{N}_i$ and update their states based on specific dynamics, like consensus dynamics. The collective system state at time t is represented as the vector $x(t) = [x_1(t)\ x_2(t)\ \dots\ x_N(t)]^T$.

To control the states of this dynamical system, we introduce external control signals applied to a subset of agents known as *leaders*. These leader agents have states that can be directly manipulated, expressed as $\dot{x}_l = u_l(t)$, where $u_l(t)$ represents the input signal. Conversely, non-leader agents, referred to as *followers*, update their states by aggregating information from their local neighborhoods. The dynamics of the followers, denoted as $\dot{x}_f(t)$ in this leader-follower system, can be expressed as:

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

where $\mathcal{M}(G)$ represents system matrices related to the followers' subgraph, and $\mathcal{H}(G)$ denotes the topological interactions between leader and follower agents. We use the Laplacian matrix for a matrix representation of a network.

For a network represented as a graph $G = (V, E)$, we partition the node set V into two groups: followers (V_f) and leaders (V_ℓ), where $|V_f| = N_f$ and $|V_\ell| = N_\ell$. We establish an ordered arrangement for the nodes, with V_f consisting of nodes $v_1, v_2, \dots, v_{N_f}$, and V_ℓ containing nodes $v_{N_f+1}, \dots, v_N$. The subgraph composed of follower nodes is referred to as the *follower graph* (G_f), represented mathematically using the Laplacian matrix L, which is partitioned as:

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

where $A \in \mathbb{R}^{N_f \times N_f}$, $B \in \mathbb{R}^{N_f \times N_\ell}$, and $C \in \mathbb{R}^{N_\ell \times N_\ell}$.

In this configuration, we introduce an external input signal u_l applied to leader agent $v_l \in V_\ell$. The follower nodes update their states according to the dynamics given by:

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

where $\boldsymbol{x_f}(t) \in \mathbb{R}^{N_f}$ represents the state vector of follower nodes at time t, and $u(t) = [u_{N_f+1}(t)\ \cdots\ u_N(t)]^T \in \mathbb{R}^{N_\ell}$ is the control signal at time t. The matrices $-A$ and $-B$ in these dynamics are derived from the network structure and leader agent selection.

From a control perspective, we are interested in assessing the feasibility of steering the system described by these dynamics from an initial state to a final state within a finite time interval t_1. If control is achievable, we aim to quantify the control energy $\mathcal{E}(u)$ required, as defined below:

$$\mathcal{E}(u) = \int_{\tau=0}^{t_1} \|u(\tau)\|^2 d\tau.$$

We also investigate the dimension of the subspace containing reachable states and the impact of changing leader agents. These inquiries provide insights into the underlying graph structure and guide the derivation of network controllability metrics for graph embeddings.

B. Network Controllability Metrics

The process of controlling a network entails the responsibility of directing it from an initial state to a desired final state by applying control inputs to specific leader nodes within the network. A state $\boldsymbol{x_f^*} \in \mathbb{R}^{N_f}$ is considered *reachable* when there exists an input that can propel the network from the origin $\mathbf{0}_{N_f}$ to the state $\boldsymbol{x_f^*}$ within a finite timeframe. The set comprising all such reachable states defines what we refer to as the *controllable subspace*. Importantly, in continuous linear time-invariant systems, such as the one described by equation (III-A), if a state $\boldsymbol{x_f}^*$ is reachable

from the origin, it is also reachable from any arbitrary initial state within any given duration of time.

The *dimension* of this controllable subspace $\gamma(G, V_\ell)$, is a pivotal concept in control theory. It can be determined by examining the *rank* of the *Controllability matrix*: $\mathcal{C} = \begin{bmatrix} -B & (-A)(-B) & \cdots & (-A)^{N_f-1}(-B) \end{bmatrix}$. The rank of this matrix hinges on the properties of matrices A and B, which, in turn, are contingent on the network's structure and the selection of leader nodes.

The *Controllability Gramian* serves as a significant mathematical construct that offers vital insights into the control characteristics of a network [16]–[18]. Utilizing the Controllability Gramian, we can quantitatively assess the ease of transitioning from one state to another, taking into account the necessary control energy as defined in equation (III-A).

For the system delineated in equation (III-A), the *infinite horizon controllability Gramian* is defined as follows:

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

If the system is stable, signifying that all eigenvalues of $-A$ have negative real parts, $\mathcal{W}$ asymptotically converges and can be computed through the Lyapunov equation:

$$(-A)\mathcal{W} + \mathcal{W}(-A)^T + (-B)(-B)^T = 0,$$

For a solution to exist for (III-B), it is necessary for $-A$ to be a stable matrix. This condition holds for connected graphs.

Lemma 1: If we partition the Laplacian matrix L of an undirected connected graph as shown in (III-A), the matrix A is positive definite [11].

In summary, 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. 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 [16]–[18]. Some of these statistics are further discussed below.

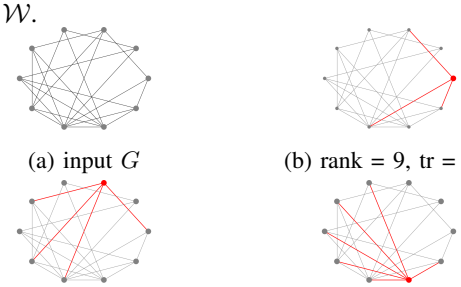
i **Trace of** $\mathcal{W}$: The trace of the controllability Gramian inversely correlates with the average control energy required to reach random target states. It also indicates the overall controllability across all directions within the state space.

ii **Minimum eigenvalue of** $\mathcal{W}$: This metric represents the worst-case scenario and demonstrates an inverse relationship with the control energy required to navigate the network in the least controllable direction.

iii **Rank of** $\mathcal{W}$: The rank of $\mathcal{W}$ corresponds to the dimension of the controllable subspace.

iv **Determinant of** $\mathcal{W}$: The metric $\text{ld}(\mathcal{W}) = \log\left(\prod_j \mu_j(\mathcal{W})\right)$, where $\mu_j(\mathcal{W})$ denotes a non-zero eigenvalue of $\mathcal{W}$, provides a volumetric assessment of the controllable subspace that can be accessed with one unit or less of control energy.

Examples: We illustrate, through examples, that network controllability is influenced by both the network's topological configuration and the placement of leaders within it. The impact of leader selection on network controllability is visualized in Figure 2. In this scenario, we examine a network consisting of 10 agents, with one designated as the leader, resulting in $N_f = 9$. In Figure 2(b), the dimension of the controllable subspace is 9, indicating complete controllability of the follower network. The edges connecting the leader and follower nodes, which define the structure of the B matrix in (III-A), are highlighted in red. Transitioning to Figure 2(c), we opt for a different leader while preserving complete controllability; however, this results in a modified trace of $\mathcal{W}$.



(a) input G (b) rank = 9, tr = 1.5

(c) rank = 9, tr = 3.5 (d) rank = 8, tr = 7.1

Fig. 2: Controllability metrics vary with leader selection

To gather valuable insights into the graph structure, we employ an effective probing strategy. By varying the number and positions of leader nodes, we can observe the resulting controllability behavior, as quantified by metrics such as **tr**$(\mathcal{W})$, $\mu_j(\mathcal{W})$, **rank**$(\mathcal{W})$, and **ld**$(\mathcal{W})$. In our empirical evaluation in Section V, we primarily focus on the rank, trace, and both minimum and non-zero eigenvalues of the Gramian matrix, considering multiple leader set configurations.

IV. Control-Based Graph Augmentations

Contrastive Graph Representation Learning (CGRL) is a self-supervised technique that relies on augmented data to create positive and negative pairs. As discussed in the previous section, we have developed graph-level control embeddings that act as inputs to the encoder, minimizing the NT-Xent loss [5]. In this section, we will introduce several innovative methods for data augmentation.

The primary purpose of data augmentation is to generate new data that is logically consistent while preserving the semantic labels. As depicted in Figure 1, CGRL incorporates a contrastive module into the conventional Graph Machine Learning (GML) architecture, introducing a contrastive loss for fine-tuning the models. This process involves contrasting the original graph with an augmentation graph using positive–positive and positive–negative pairs. The core idea behind CGRL is that the original graph should, to some extent, be equivalent to the augmentation graph. Therefore, each graph in the dataset should have precisely one corresponding positive pair (an equivalent graph) in the augmented dataset, while the remaining augmented graphs serve as negative pairs. During training, CGRL strives to optimize the similarities of positive–positive pairs to approach unity and the similarities of positive–negative pairs to approach zero.

Since the graph-level embeddings used in our work are based on control properties, the primary goal is to devise an

augmentation technique that primarily preserves the control properties of the original graph in the augmented version.

***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.

While performing augmentation, our primary focus is on preserving one of the key features of our CTRL embedding, which is the rank of controllability denoted as $\gamma(G, V_\ell)$. However, preserving the exact rank of controllability is a highly complex task. Consequently, the literature has explored lower bounds on network controllability [9], [19], [20]. In our edge perturbation techniques, we employ a rigorous lower bound based on topological node distances and introduce algorithms to augment the original network while preserving this lower bound [9].

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:

$$D_i = \begin{bmatrix} d(\ell_1, v_i) & d(\ell_2, v_i) & \cdots & d(\ell_m, v_i) \end{bmatrix}^T \in \mathbb{Z}^m.$$

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 [9].

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 $[\mathcal{D}_i]_{\pi(i)}$.

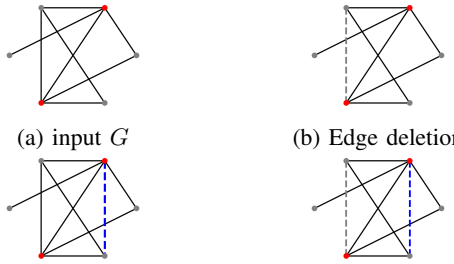
The length of the PMI sequence provides a precise lower bound on the dimension of the controllable subspace $\gamma(G, V_\ell)$. This is presented in the subsequent result.

Theorem 4.1: [9] If we denote the length of the longest PMI sequence of DL vectors in a network $G = (V, E)$ with V_ℓ leaders as $\delta(G, V_\ell)$ or simply $\delta(G)$, then we can establish the inequality: $\delta(G, V_\ell) \leq \gamma(G, V_\ell)$ Here, $\gamma(G, V_\ell)$ 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 3. 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.

A. Egde Deletion

We propose using the concept of discerning essential edges, referred to as controllability backbone edges that we introduced in [21], which do not decrease $\delta(G, V_\ell)$. We begin by introducing the concept of the distance-based controllability backbone and subsequently present an algorithm for computing an augmented graph with k-perturbed edges while utilizing the controllability backbone.



(a) input G (b) Edge deletion (c) Edge addition (d) Edge substitution

Fig. 3: Control-based graph augmentations where $\delta = \gamma = 4$ for original and augmented graphs.

Definition (*Controllability Backbone*) [21] 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 $E_B \subseteq \hat{E} \subseteq E$.

In other words, for any subgraph $\hat{G} = (V, \hat{E})$ of G that includes the backbone edges E_B, it ensures that at least the same level of controllability as the original graph G is maintained. Essentially, retaining the backbone edges guarantees that controllability remains unchanged or improves within any subgraph $\hat{G}$.

Now, we present Algorithm 1 to perform graph augmentation by deleting edges from a graph. First, we compute the important edges (of the controllability backbone) that need to be preserved to maintain the minimum bound on controllability. The controllability backbone can be found by using Algorithm 2 of [21]. Then, we calculate a set of potential edges in the given graph G that does not include any edges from the backbone graph B. Finally, we randomly select k edges from the set of potential edges and remove them from the original graph G to obtain an augmented graph G'. If k is larger than the set of potential edges, we delete all the edges of the potential edge set.

Algorithm 1 *Edge_Deletion*

Input: $G = (V, E)$, V_ℓ, 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_ℓ.
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: Let $G = (V, E)$ be a graph with a set of leaders V_ℓ. The backbone graph $B = (V, E_B)$ is defined as the smallest set of essential edges required to maintain the necessary distances within the graph for establishing the lower bound on controllability, as formally described in Theorem 4.2 of [21]. The theorem establishes that for any edge e present in G but absent in B, the removal of edge e does not diminish the lower bound on $\gamma(G, V_\ell)$. Therefore, any edge that is not part of the backbone graph B can be eliminated from G without reducing the lower bound on the rank of the controllability matrix. Furthermore, this property implies that the removal of any subset of these non-backbone edges from G continues to maintain the bound on the rank of the controllability matrix. ■

B. Egde Addition

Building on the concept of recognizing removable edges while preserving the distance-based bound $\delta(G, V_\ell)$, as proposed in our previous work [22], we use an augmentation method that determines the edges that can be added to a given graph $G = (V, E)$ while still maintaining the distance-based bound $\delta(G, V_\ell)$. We employ the same approach outlined in [22] to identify edges that can be added to G without diminishing the distance-based bound $\delta(G, V_\ell)$. After determining such potential edges that can be safely incorporated into G without reducing the controllability rank $\gamma(G, V_\ell)$, we randomly select k such potential edges and introduce them into G, resulting in the augmented graph $G' = (V, E')$.

Proposition 4.3: [22] If $\delta(G, V_\ell)$ serves as 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 the edges that preserve the 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 [22].

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: Let $\mathcal{D}$ be the longest PMI sequence of length $\delta(G, V_\ell)$ in $G = (V, E)$ and $V_\mathcal{D} \subseteq V$ are the nodes whose DL vectors are in $\mathcal{D}$. If we remove k edges from G to form $\bar{G} = (V, \bar{E})$, then by Proposition 4.2, $\delta(\bar{G}, V_\ell) \leq \delta(G, V_\ell)$ i.e. the longest PMI sequence of G is a subsequence of the longest PMI sequence of $\bar{G}$ as the distances between leaders and nodes in $V_\mathcal{D}$ are exactly the same in G and $\bar{G}$.

Algorithm 2 *Edge_Substitution*

Input: $G = (V, E)$, V_ℓ, 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_ℓ.
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** $G' = (V, E')$

TABLE I: Statistics of the datasets. Number of graphs, average number of nodes and edges, range of number of vertices, and the number of classes.

Dataset	#Graphs	avg.|V|	avg.|E|	Range(|V|)	#Classes
MUTAG	188	17.93	19.79	10-28	2
PTC	344	14.29	14.69	2-109	2
PROTEINS	1113	39.06	72.82	4-620	2
DD	1178	284.32	715.66	30-743	2
COLLAB	5000	284.32	715.66	32-492	3
IMDB-B	1000	19.77	96.53	12-136	2
IMDB-M	1500	13.00	65.94	7-89	3

Now, from Algorithm 1 of [22], we find a maximal set of edges E_{max} for the original graph $G = (V, E)$. This edge set contains all the edges that can be added to G while preserving the distances between leaders and nodes in $V_\mathcal{D}$. Hence, any edge added to G from E_{max} will maintain the lower bound $\delta(G, V_\ell)$ for $\gamma(G, V_\ell)$. Hence, by deleting k edges that are mutually exclusive from the controllability backbone and adding k edges that keep the distances between leaders and nodes in $V_\mathcal{D}$ the same, we can substitute k edges in G for given V_ℓ such that $\delta(G, V_\ell) \leq \delta(G', V_\ell)$. ■

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 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 [23]. 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 I.

B. The Role of Data Augmentation in Graph CL

We evaluate the effectiveness of our proposed framework for graph classification using the TUDataset benchmark [23]. 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 II. 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.

TABLE II: 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

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 outlined in [13], [24]. We closely adhere to the established approach within Graph CL for graph classification [7], [13]. We compare our results with the following kernel-based, unsupervised, and self-supervised methods.

a) Graph Kernels Methods: Graph kernel-based techniques represent traditional approaches to graph classification. They engage directly with graph data by crafting kernel functions that preserve the graph's structural information. For the purpose of this comparative analysis, we have chosen four well-established graph kernel-based methods: Graphlet kernel (GK) [25], Weisfeiler-Lehman sub-tree kernel (WL) [26], and deep graph kernels (DGK) [27].

b) Unsupervised Methods: Unsupervised techniques for graph representation learning leverage sub-graph and node similarity scores to guide the learning process, all without relying on label information. In this comparative analysis, we have chosen three prominent unsupervised benchmarks methods: node2vec [28], sub2vec [29], and graph2vec [24].

c) Self-Supervised Methods: Unlike traditional unsupervised graph representation techniques, the self-supervised approach to graph representation harnesses the capabilities of contrastive learning to unearth profound insights between data samples, leading to significant improvements in graph representation learning. In this comparative study, we've chosen two state-of-the-art contenders, InfoGraph [13] and GraphCL [7], which both employ graph neural networks as their foundational architecture.

The results of graph classification are presented in Table III. 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 III, 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 [7]. 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 IV 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

In this work, we introduced Control-based Graph Contrastive Learning (CGCL), a novel framework for unsupervised graph representation learning that leverages graph controllability properties. We employed advanced edge augmentation methods to create augmented data for contrastive learning while preserving the controllability rank of graphs. Our extensive experiments on standard graph classification benchmarks showcased CGCL's effectiveness, outperforming SOTA unsupervised and self-supervised methods on multiple datasets. We also compared CGCL with a random edge augmentation approach, underscoring the significance of our controllability-driven augmentation strategy. The success of CGCL suggests that incorporating domain-specific structural knowledge, like controllability, can significantly enhance graph representation learning, opening avenues for further research. Overall, CGCL presents a promising approach for applications requiring informative graph representations. In the future, we aim to refine our graph augmentation techniques to preserve all relevant control features.

TABLE III: 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 IV: 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

REFERENCES

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

[2] A. Said, R. G. Bayrak, T. Derr, M. Shabbir, D. Moyer, C. Chang, and X. Koutsoukos, "Neurograph: Benchmarks for graph machine learning in brain connectomics," *arXiv preprint arXiv:2306.06202*, 2023.

[3] A. Said, S.-U. Hassan, S. Tuarob, R. Nawaz, and M. Shabbir, "Dgsd: Distributed graph representation via graph statistical properties," *Future Generation Computer Systems*, vol. 119, pp. 166–175, 2021.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] N. Monshizadeh, S. Zhang, and M. K. Camlibel, "Zero forcing sets and controllability of dynamical systems defined on graphs," *IEEE Transactions on Automatic Control*, vol. 59, pp. 2562–2567, 2014.

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

[12] H. Duan, C. Xie, B. Li, and P. Tang, "Self-supervised contrastive graph representation with node and graph augmentation," *Neural Networks*, 2023.

[13] 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.

[14] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International conference on machine learning*. PMLR, 2020, pp. 4116–4126.

[15] A. Said, O. U. Ahmad, W. Abbas, M. Shabbir, and X. Koutsoukos, "Network controllability perspectives on graph representation," *IEEE Transactions on Knowledge & Data Engineering*, no. 01, pp. 1–12, 2023.

[16] 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.

[17] 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.

[18] 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.

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

[20] N. Monshizadeh, K. Camlibel, and H. Trentelman, "Strong targeted controllability of dynamical networks," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 4782–4787.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.