Approaches and Techniques to Enhance Efficiency and Performance of Non-Contrastive Self-Supervised Learning Methods

by

Ali Saheb Pasand

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Self-supervised learning (SSL) methods have gained considerable attention in recent years due to their ability to learn useful representations of data without relying on labels during training. These methods have revolutionized research across various domains, including Natural Language Processing, Computer Vision, and Graph Deep Learning. SSL methods can be classified into three categories: Generative, Predictive, and Dual-Encoder. Among these, Dual-Encoder methods have gained significant popularity in many applications as, in contrast with generative methods, they do not require training a powerful decoder. Also, in contrast with predictive SSL methods, they do not need a careful design of the pre-training task. However, Dual-Encoder techniques suffer from collapse in representation space, since mapping all the samples into the same point in the embedding space is the trivial solution to their optimization problem. Based on their mechanisms for preventing this issue, those can be further divided into two classes: Contrastive and Non-Contrastive. Contrastive techniques prevent representation collapse using negative sampling, while Non-Contrastive techniques employ asymmetries such as stop-gradient or information maximization through Covariance/Cross-Covariance matrices whitening.

This thesis aims to enhance two classes of Non-Contrastive Dual-Encoder SSL methods - Asymmetry-Based and Covariance-Based. The proposed improvements are as follows:

- Covariance-Based methods: This thesis proposes techniques to enhance the efficiency and performance of Covariance-Based methods. Specifically, the invariance loss term's efficiency is improved through various data sampling techniques, and the covariance whitening term's efficiency is enhanced through random dimension selection, LSH bucketing, Nystrom approximation, and random projection. These techniques are thoroughly tested and validated.

- Asymmetry-Based methods: This thesis enhances the performance of Asymmetry-Based methods by preventing rank degradation and early termination of the update procedure for the target network.

Overall, this thesis proposes novel techniques to enhance the efficiency and performance of Non-Contrastive Dual-Encoder SSL methods. Proposed ideas tested and validated for benchmark static graph datasets. However, those are applicable to other applications and modalities as well.

Acknowledgements

I would like to express my gratitude to all those who have contributed to making this thesis possible. In particular, I would like to thank my supervisor, Professor Ali Ghodsi, whose expert guidance and mentorship have guided me through this academic journey. I would also like to extend my appreciation to my supervisors at HUAWEI Noah's Ark Lab, Dr.Biparva and Ms.Yingxue Zhang, whose support and invaluable assistance have been essential to the successful completion of this project. Lastly, I would like to acknowledge the contributions of my friends and colleagues, Benyamin Ghojogh, Mohammad Ali Alomrani, and Raika Karimi, who have generously shared their knowledge, insights, and comments with me throughout this endeavor.

Dedication

To my mother and to all scientists who have dedicated their lives to shaping our understanding of the world.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, self-supervised learning (SSL) has emerged as a powerful approach for training deep neural networks without the need for large labeled datasets. Unlike supervised learning, which relies on manually labeled data, SSL leverages the inherent structure and information present in the data to learn useful representations. In particular, Non-Contrastive Self-Supervised Learning (NCSSL) has shown great potential in various applications such as Computer Vision (CV), Natural Language Processing (NLP), and Graph Deep Learning which is the application of interest in this thesis.

Non-contrastive self-supervised learning aims to learn representations by minimizing the distance between different views of the same data instance, rather than contrasting different data instances. By doing so, these methods focus on learning the intrinsic properties of the data and reducing the impact of external factors, such as noise and outliers, on the learned representations. This can lead to more robust and generalizable representations, which can improve the performance of downstream tasks such as classification and clustering. Despite its potential benefits, NCSSL is still at early stages of development.Many techniques within this category, as described in the literature, lack theoretical justification and are characterized by ad hoc methods indicating the potential for significant improvements. This thesis aims to contribute to this research by proposing techniques to improve the efficiency and performance of different types of NCSSL methods.

The research questions which this thesis is trying to answer are:

- How the efficiency and the downstream performance of covariance-based non-contrastive methods can be improved?

- In asymmetry-based methods, what type of regularizer can improve the downstream performance with minimal computational overhead?
- In asymmetry-based methods, do we need to perform momentum-based model update procedure on the target network in all epochs?

In order to address these research questions, this thesis will conduct a comprehensive review and comparison of various SSL paradigms and relevant literature on non-contrastive/contrastive SSL techniques within the domain of graph deep learning. Then, novel approaches will be proposed to enhance both covariance-based and asymmetry-based methods separately. A set of experiments will be conducted on various static graph datasets to assess the effectiveness of the proposed methods in terms of their downstream performance compared to vanilla versions and other contrastive/non-contrastive SSL baseline methods.

Abbreviations and acronyms used in this thesis are listed in Table 1.1. The thesis is organized as follows:

- Chapter 2: This chapter focuses on general SSL, beginning with an introduction to various SSL paradigms. For each paradigm, the chapter will discuss several examples from diverse literature and identify their strengths and weaknesses. Furthermore, the chapter will provide justification for selecting NCSSL methods as the focus of this thesis, emphasizing their advantages over alternative approaches.
- Chapter 3: This chapter is focused on the Non-Contrastive/Contrastive SSL literature within the domain of Graph Deep Learning. First, a range of commonly used graph augmentation techniques will be introduced and explicated. Then, the chapter will discuss different levels of contrast, followed by an in-depth explanation of some recent works in the literature. Additionally, the chapter will provide details on the datasets used as benchmarks in this thesis, the downstream task, and the evaluation methodology.
- Chapter 4: This chapter proposes novel approaches for enhancing covariance-based non-contrastive SSL methods, which are presented and justified. A comprehensive set of experiments will be conducted to examine the impact of these techniques.
- Chapter 5: In this chapter, a novel approach is proposed for enhancing asymmetry-based non-contrastive SSL methods by regularizing the weights of the backbone encoder. The proposed methodology will be justified and evaluated through comprehensive experimentation.

Abbreviation	Definition
SSL	Self-Supervised Learning
CSSL	Contrastive Self-Supervised Learning
NCSSL	Non-Contrastive Self-Supervised Learning
CV	Computer Vision
NLP	Natural Language Processing
GRACE	GRAph Contrastive rEpresentation learning
MVGRL	Multi-View Graph Representation Learning
DGI	Deep Graph Infomax
BGRL	Bootstrapped GRaph Latents
GBT	Graph Barlow Twins
VICReg	Variance Invariance-Covariance Regularization
BYOL	Bootstrap your own latent
EMA	Exponential Moving Average
LSH	Locality Sensitive Hashing
RP	Random Projection
SVD	Singular Value Decomposition
GCN	Graph Convolution Network
GNN	Graph Neural Network
KD	Knowledge Distillation
HSIC	Hilbert-Schmidt independence criterion

Table 1.1: List of abbreviations and acronyms used in this thesis.

- Chapter 6: This chapter introduces a novel approach to enhance asymmetry-based non-contrastive SSL methods, which involves disabling the momentum update of the target after a turning point. The proposed methodology will be justified by insights from Knowledge Distillation literature.
- Chapter 7: This chapter presents a comparative analysis of the best methodologies presented in Chapters 4 to 6, as well as other baseline methods, and provides a conclusive evaluation.
- Chapter 8: In this chapter, future work for further improvements of non-contrastive/contrastive methods are discussed.
- Appendix: This chapter provides a summary of the experimental setups used in the experiments presented in Chapters 4 to 6.

Chapter 2

Self-Supervised Learning (SSL)

2.1 Chapter's Overview

This chapter will introduce the Self-Supervised Learning paradigm and discuss the key techniques employed for unsupervised learning, comparing and contrasting them with each other. Finally, it will be justified why Non-Contrastive Self-Supervised Learning (NCSSL) is selected as the primary method of focus in this thesis.

2.2 Introduction

Although supervised learning has demonstrated promising performance in various applications, it is often impractical and unfeasible. The process of obtaining supervised signals and annotating data samples can be expensive and time-consuming, limiting its scalability. For example, in the Drug Target Interaction task, which aims to identify proteins and drugs that interact with each other, testing the interaction between multiple drugs and proteins can be both costly and time-consuming. To address this issue, models must be trained under a low-shot regime, meaning with limited labeled data. The solution to the lack of labeled data problem is Transfer Learning. In this scenario, a model that is large enough and has the capacity to represent data accurately is pre-trained without the use of manually labeled data. This pre-trained model is then fine-tuned using the limited labeled data from tasks of interest. It has been shown that as the model has been pre-trained first with a large database of data, it is already capable enough to embed data samples

meaningfully. By the extra fune-tuning step, it will learn to tweak the embedding space in order to achieve proper representation for the task at hand.

Using large pre-trained models as initialization for the fine-tuning stage has revolutionized and improved performance in many tasks across different areas such as Computer Vision (CV), Natural Language Processing (NLP), and bioinformatics. Models such as Alexnet [34], VGG [53], and GoogLeNet [56] have significantly impacted the field of Computer Vision. The introduction of BERT [10], Roberta [38], and GPT [45] has rapidly transformed the NLP literature to the extent that 2018 is now considered the boundary between classic and modern NLP. Encoders for proteins such as ESM [37] have also emerged, revolutionizing the bioinformatics literature.

To pre-train these large models, a significantly large number of samples is required. However, labeling many samples is not always possible in many scenarios. Therefore, researchers have developed techniques by which a large model can be pre-trained using samples without having access to manually labeled data, or with only data samples which have been labeled automatically with the help of self-supervised signals. The procedure of training a model without using any manually labeled data samples is called self-supervised Learning. Self-supervised learning (SSL) techniques can be divided into three main categories:

- Generative SSL: These methods train an encoder-decoder architecture, where the encoder aims to learn the most representative representation for the data to help the decoder in generating higher quality samples.

- Predictive SSL: These methods define pre-training tasks that do not require manual annotation or label generation. The encoder learns informative representations of the data by solving these tasks.

- Dual-Encoder (Contrastive/Non-contrastive) SSL: These methods train the encoder on different versions of the same data sample, ensuring that their embeddings are close to each other. This similarity maximization without adopting any extra measures leads to a phenomenon called "Representation Collapse" which means encoding all data samples into the same point in the embedding space. These methods can be categorized as contrastive or non-contrastive, depending on their approach to preventing collapse in the embedding space. Contrastive methods prevent collapse by performing negative sampling, which leads to maximizing similarity between same-data embeddings, and minimizing similarity between different-data embeddings. Non-contrastive methods prevent collapse through different measures, computationally less complex and require less memory than negative sampling, such as covariance/cross-covariance

whitening, stop-gradient in one branch, and updating the weights of encoder in one branch by the weights of the other encoder (teacher-student framework).

There are also methods that combine these categories, but they will not be discussed in this thesis. Each category has its advantages and disadvantages, which will be pointed out in the following sections. Additionally, the focus of this thesis will be on non-contrastive SSL (NCSSL), and the reasons for this choice will be argued in subsequent sections.

2.3 Generative Self-Supervised Learning

Autoencoder structures form the basis of Generative Self-Supervised Learning techniques, in which a portion of the data (such as pixels of an image or words in a text) is typically masked or corrupted, and the decoder is trained to reconstruct the original image. Through this process, the encoder learns to map semantically similar samples to geometrically close points in the embedding space, in order to assist the decoder. Figure 2.1 illustrates one such example, which involves masking parts of the data and generating the missing patches using the decoder.

These techniques have been successfully applied in various fields such as Natural Language Processing [10, 38, 35], Computer Vision [22, 46], and Graph Deep Learning [26, 33]. One of the major advantages of these methods is their similarity to self-supervised learning in the human brain, which has a powerful generative capacity and an accurate model of the world. Additionally, these methods provide both an encoder and a generator, which can be useful in different applications.

However, training generative models is not a simple task, and it may not be justifiable if the goal is solely finding an appropriate embedding for classification or clustering. Moreover, the loss function for these methods is typically based on reconstruction loss in the general data space (e.g., the image space), which is not a proper metric for comparison as to compare the reconstructed data with the actual data, a proper metric in the actual data space is needed which is often difficult to obtain. This limitation makes these methods less justifiable since the ultimate goal of self-supervised learning is to find that metric.

2.4 Predictive Self-Supervised Learning

These methods involve defining effective pre-training tasks that can be formulated as classification tasks and whose labels can be easily obtained from the data. It is also essential

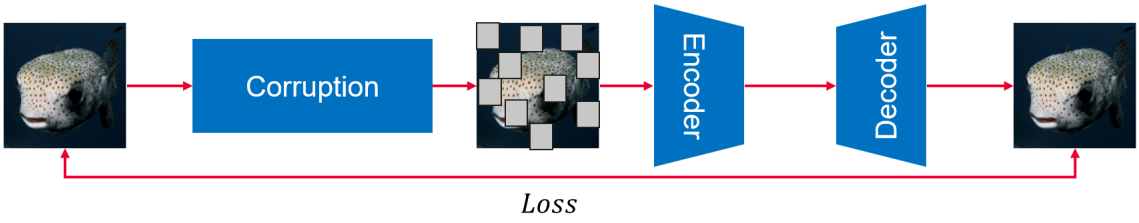

Figure 2.1: Overview of Generative Self-Supervised Learning methods. These techniques are based on corrupting part of the data and training an autoencoder structure which can reconstruct the actual data sample. The differences between these methods are mainly in the loss function and the type of corruption used during training.

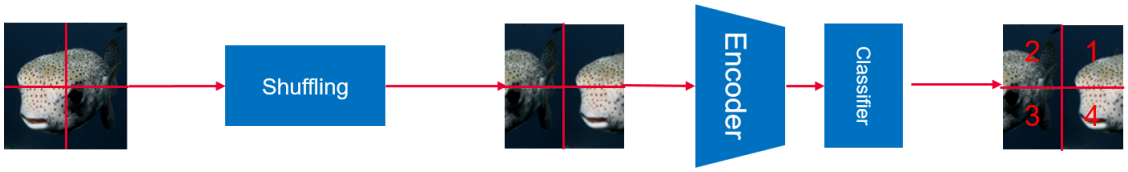

Figure 2.2: One example of Predictive Self-Supervised Learning methods. It is based on shuffling patches of a data sample and trying to predict the correct ordering of patches. The main differences between different methods in this category is the pre-training task used by them.

that these tasks provide benefits for various downstream tasks, as the relevance and generality of the pre-training task determine the quality of the resulting embedding. Figure 2.2 illustrates one example of this type of self-supervised learning, which involves shuffling patches of the data sample and defining the classification and predictive task as predicting the correct order of patches.

Some common examples of these techniques include Permuting patches and predicting order [75, 31], Image or Video Coloring [78, 44, 65], Next Sentence Prediction [10, 38, 35]. One major advantage of these techniques is that they can be formulated as classification with labels which can be obtained easily and automatically.

However, defining an appropriate pre-training task is crucial and can be challenging. The task should neither be difficult, as this would lead to non-informative representations, nor straightforward, as this would result in trivial representations.

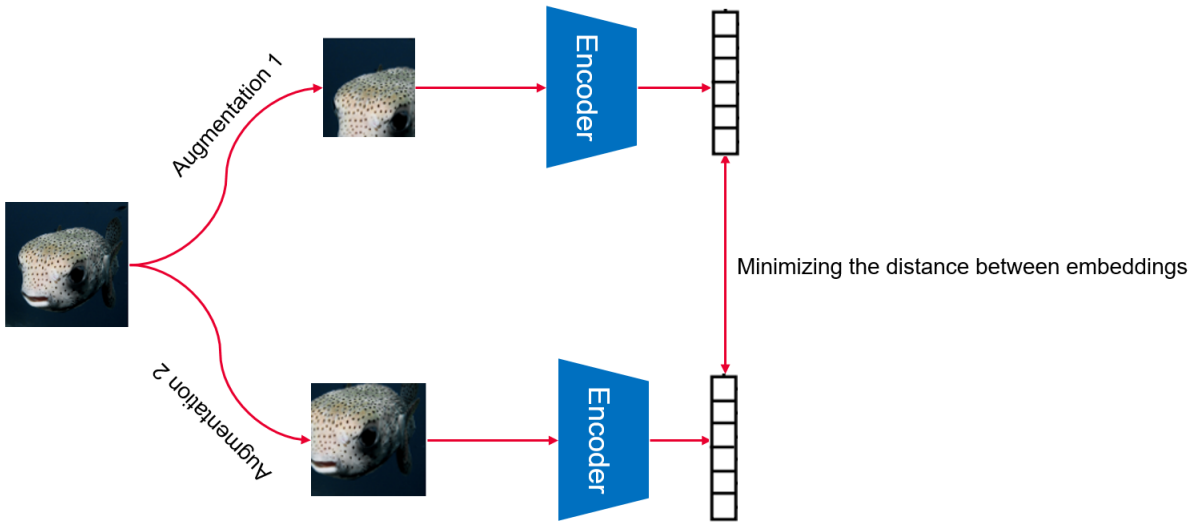

Figure 2.3: Overview of Dual-encoder Self-Supervised Learning methods. These methods are based on obtaining different versions of data by performing data augmentation, image cropping in this depicted case, and maximizing the similarity between their embedding.

2.5 Dual-Encoder Self-Supervised Learning

These methods are based on this assumption that different views of a data sample are semantically similar and should be mapped into the same point in the embedding space. Therefore, all methods under this category consist of two or multiple branches of encoders which encode different views of a data sample and try to maximize the similarity between the embedding vectors corresponding to views of the same data sample. One example of this scheme is shown in Figure 2.3.

Similarity maximization without any further considerations leads to a phenomenon called "Collapse in Representation" which corresponds to the case in which encoder learns to map all samples into the same point in the embedding space. Dual-encoder SSL techniques based on their mechanism for preventing collapse can be further divided into two classes: Contrastive or Non-Contrastive which are explained in the following sub-sections.

2.5.1 Contrastive Self-Supervised Learning (CSSL)

In these self-supervised learning techniques, the goal is to learn an embedding space where semantically similar samples are located close to each other while non-related samples are far apart. Since there are no explicit labels, the notion of semantic similarity is defined through data augmentation. The idea is to generate augmented versions of a data sample and minimize the distance between their representations in the embedding space if they belong to the same data point, and maximize the distance if they belong to different data instances. Examples of such techniques include SimCLR [6] and MoCo (memory bank) [23, 7].

The advantages of these methods are that they do not require a generator or decoder, and there is no need to define a specific task. Additionally, they can be theoretically explained using classic manifold learning techniques [2], where preserving closeness in the embedding space is equivalent to preserving the local structure of the manifold.

However, these methods require negative sampling, which can be challenging. To perform negative sampling effectively, a large number of negative samples is required, which can be memory-intensive. For example, SimCLR requires a large batch size of samples for effective negative sampling, which makes it impractical in many scenarios.

2.5.2 Non-Contrastive Self-Supervised Learning (NCSSL)

As discussed before, these methods adopt mechanisms different than negative sampling in order to prevent representation collapse. One class of these methods try to make embedding variables decorrelated, covariance or cross-covariance whitening, [3, 73, 4]. Other class of Non-Contrastive methods are based on imposing asymmetries in training procedure. For example, using stop-gradient in one branch and updating one encoder by the weights of the other one instead of weight sharing scheme [58, 19]. These SSL methods have become increasingly popular due to their desirable features, such as the ability to work with smaller batch sizes in contrast with contrastive SSL methods [16], their applicability to various types of data (discrete and continuous) [1, 3], and their potential for universal SSL frameworks for multimodal data [1]. However, these methods suffer from several drawbacks compared to contrastive ones, which are as follows:

1. Although non-contrastive methods are more efficient compared to their contrastive counterparts, there is still room for improvement in terms of efficiency by reducing redundant calculations.

2. As these methods do not perform negative sampling, they usually lead to rank degradation in the embedding space [2]. In VICReg, it has been shown that the rank of the output cannot be larger than a specific threshold [2, 3].

3. Techniques used in NCSSL methods to prevent collapse lead to inter-sample repulsion for data samples in the same batch. If we have similar points in the same batch, their distance will be maximized as they will be considered as non-neighbour samples [16].

4. Non-contrastive methods heavily rely on using local information of manifolds [2] instead of providing a global view, which may limit their effectiveness in some scenarios.

In the following sections, methods will be proposed and justified to address some of these issues, with a focus on graph modeling applications. The proposed techniques in this thesis are applicable to various types of data and can be used to improve the efficiency and effectiveness of any current or future successful NCSSL method.

2.6 Chapter's Conclusion

Based on the comparisons made between different SSL paradigms, it has been determined that for the focus of this thesis on modeling graphs, generative SSL paradigm is not suitable as finding a proper decoder to generate graphs is challenging. Additionally, determining a suitable predictive task for graphs is not straightforward. When comparing contrastive and non-contrastive methods, non-contrastive methods are more memory-efficient and less computationally complex. Moreover, they tend to perform better to model graph datasets [81]. In the next chapter, prior contrastive/non-contrastive methodologies proposed for graph datasets will be reviewed. Also, the baselines and the previous methods selected to be enhanced will be explained and compared.

Chapter 3

Contrastive/Non-Contrastive Self-Supervised Learning Methods for Graphs

3.1 Chapter's Overview

This chapter provides an in-depth exploration of Dual-Encoder Self-Supervised Learning (SSL) methods for graphs, both Contrastive and Non-Contrastive. It begins by introducing common augmentation types used in these methods, followed by a discussion of the different levels at which contrasting procedures can be applied.

The chapter then compares recent Contrastive/Non-Contrastive methods that have been chosen as baselines, based on their adopted method for collapse prevention.

3.2 Introduction

In the previous chapter, the key components of Dual-Encoder SSL methods were discussed, which include:

- Augmentation Type: The first step in Dual-Encoder methods is data augmentation, which aims to produce multiple versions of a given data sample while preserving its intrinsic characteristics. These versions are then mapped into the vicinity of each other in the embedding space.

- Scale of Contrast: Graph datasets can be contrasted in various scales, such as the whole graph, a part of the graph, or a single node. Each scale of contrast produces different embedding space and similarity maximization/minimization objectives.

- Collapse Prevention Mechanism: To prevent trivial embeddings where all data samples are encoded into the same point in the embedding space, Dual-Encoder SSL methods use different procedures to prevent representation collapse.

In the next sections of this chapter, different augmentation methods commonly used in various SSL techniques develops for graph will be introduces. Also, different scales of contrast common for graph SSL literature will be discussed. Finally, the collapse prevention procedures used in recent Dual-Encoder SSL methods will be explained. Through explaining collapse prevention mechanisms, the baselines used in this thesis will be explained as well. The categories mentioned in Augmentation and scale of contrast sections are inspired by Liu et al.'s survey on self-supervised learning for graphs [39].

3.3 Augmentation Methods

While there are some augmentation-free methods in the literature [36], data augmentation is an integral part of many Dual-Encoder Self-Supervised Learning (SSL) methods. However, in graphs, it is not straightforward to perform data augmentation without significantly altering the graph's structure. Nevertheless, in the SSL literature for graphs, several data augmentation methods have been frequently used and shown to be effective in learning proper graph representations. These methods are illustrated in Figure 3.1. The explanation of these augmentation techniques are as follows:

- Node Shuffling: One of the data augmentation methods used in SSL for graphs is Node Shuffling. In this method, the embeddings of different nodes are switched with each other, which is equivalent to permuting the indices of nodes in the adjacency matrix. Node Shuffling is particularly useful for Predictive SSL tasks that involve finding the correct indices of nodes after shuffling. Nevertheless, it has also been used in Dual-Encoder frameworks [47, 10, 43] and has shown to be effective in improving graph representations.

- Feature Masking: Another commonly used data augmentation technique in SSL for graphs is Node Feature Masking. This technique involves randomly masking the initial features of a subset of nodes within a given graph. The initial embeddings

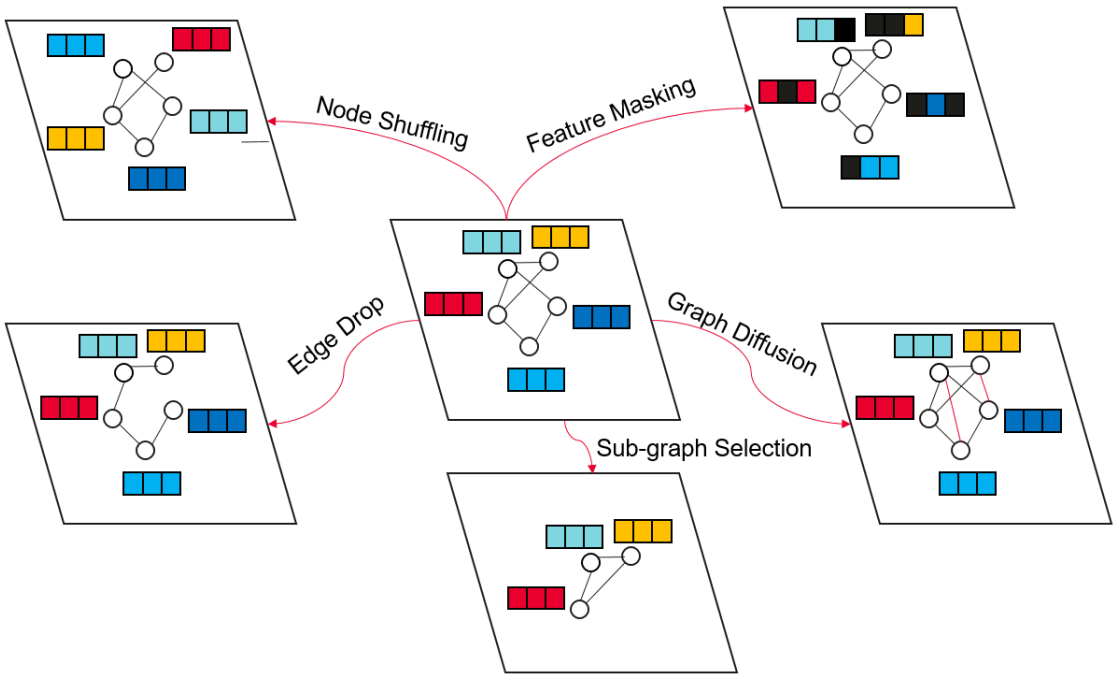

Figure 3.1: Different methods of augmentation used in Graph Deep Learning.

for nodes can be obtained in various ways, such as transforming indices to one-hot vectors, transforming random walks into embedding vectors as in DeepWalk [68], or using the structure of the graph as in node2vec [20]. The masking process can involve masking the entire embedding of a node [29, 25], or only a portion of its features [4, 58, 82, 28]. Furthermore, instead of random masking, masking can be done with higher probability for unimportant nodes, similar to masking in GCA [83]. These various approaches to Node Feature Masking have shown to be effective in improving graph representations in SSL.

- Edge Drop: The third augmentation method which is widely used in Dual-Encoder SSL techniques for graphs is edge removal. In this procedure, a portion of edges will be randomly [80, 4, 58, 72, 28, 77, 82] or adaptively [83] chosen to be removed.

- Graph Diffusion: This method is an edge appending method based on using the graph diffusion. Graph Diffusion shows how information propagates through the graph [17]. Incorporating and adding edges based on Graph Diffusion helps to provide SSL methods with global information [39] about graph. This augmentation method has

been used in some SSL methods such as [28], it has the potential to be used in many other methods which rely solely on the local information such as VICReg [3, 2].

- Sub-Graph Sampling: This type of augmentation is based on sampling a segment of the original graph as a new version of the graph. This augmentation is inspired by image cropping augmentation from Computer Vision literature. It has been used in various SSL methods such as [21, 27, 74]. While using image cropping is well established and justified in Computer Vision, as various downstream tasks rely on detecting type of the object and it will not change by cropping, it is not justifiable well in many types of tasks in graph deep learning. Especially the task of interest in this thesis which is node classification. This sub-graph sampling type of augmentation is useful in some applications such as graph classification which is not studied in this thesis.

In this thesis, the combination of Feature Masking and Edge Drop have been used as the augmentation method. The justification behind using the combination of these two specific method is that those are more common in different SSL methods and shown to be helpful in finding proper representations. However, Graph Diffusion could be helpful as well as this method can bring global information into the methods which lack global information [3, 2]. Using Graph Diffusion in combination with VICReg is left as the future work. In addition, while Sub-Graph Sampling is a well-established augmentation method, it has not been used in this thesis as it is not suitable when attempting to learn representations for a single graph at a node-level contrast.

3.4 Scale of Contrast

SSL (Semi-Supervised Learning) methods aim to maximize the similarity between semantically similar instances in the embedding space. This similarity maximization can be performed in various scales as shown in Figures 3.2 and 3.3. There are two main categories for the scale of contrast:

- Same-Scale Contrasting: The Same-Scale Contrasting methods maximize the similarity of the embedding of the same nodes in different versions of the original graph or the aggregated embedding of two views of the same graph. For instance, [58, 4, 82] use the first method, while [82, 74] utilize the second method.

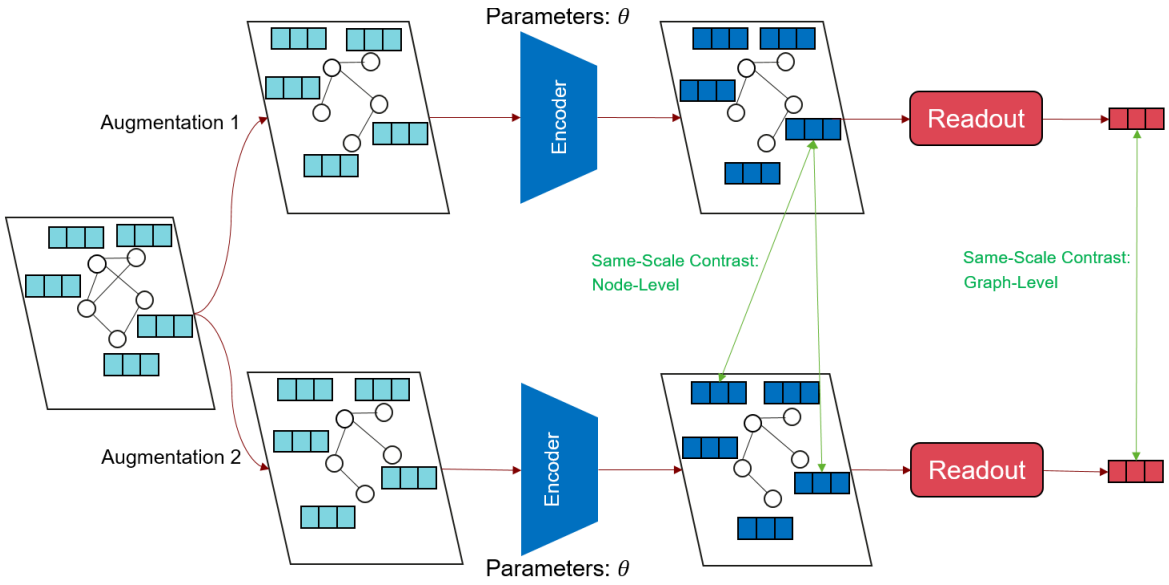

Figure 3.2: Contrasting at the same scale between nodes or final graph embedding.

- Cross-Scale Contrasting: The Cross-Scale methods maximize the similarity between the final embedding of the whole graph with some nodes [63, 41], or by maximizing the similarity between the embedding of a version of a graph and some sections of the other version of the same graph [55, 67, 5].

In this thesis, as the downstream task of interest is node classification, the node-level same-scale contrast has been chosen as the scale of contrasting.

3.5 Collapse Prevention Mechanism

In the previous chapter, it was discussed that Dual-Encoder schemes are commonly used for Self-Supervised Learning, but they suffer from a major issue called "collapse in the representation". This refers to the phenomenon where the encoders learn to map all input samples into the same point or vicinity of a point in the embedding space. Therefore, any SSL method based on Dual-Encoder scheme must introduce a mechanism for preventing this phenomenon. Such methods can be classified into two categories: Contrastive and Non-Contrastive. Contrastive methods use negative sampling, while Non-Contrastive methods do not.

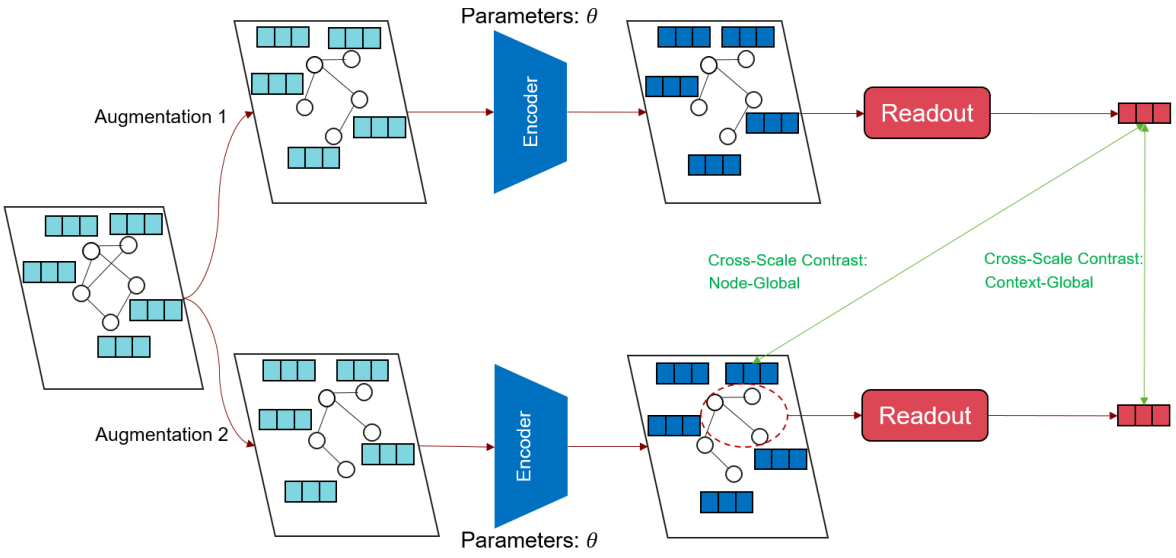

Figure 3.3: Contrasting at the different scales between nodes and final graph embedding or final graph embedding and part of the graph embedding.

In this section, we will discuss some of the recent and well-established methods for graph deep learning that have adopted various collapse prevention mechanisms. These methods have been chosen as baselines due to the frequency of being analyzed in different papers and the variety in their adopted mechanisms for preventing collapse.

3.5.1 Contrastive SSL Methods for Graphs

Contrastive SSL methods avoid collapse by utilizing negative sampling, which involves minimizing the similarity between different versions of different data samples while maximizing the similarity between the versions of the same sample. In the context of graphs and node-level same-scale contrast, this approach translates to maximizing the similarity between embeddings of the same node in different versions of the graph, while minimizing the similarity between embeddings of different nodes in the same version or between different versions. While these methods have proven to be effective, they lack diversity compared to non-contrastive methods. In the following section, some recent contrastive SSL methods developed for graphs, GRACE [82], DGI [63], and MVGRL [21], will be discussed.

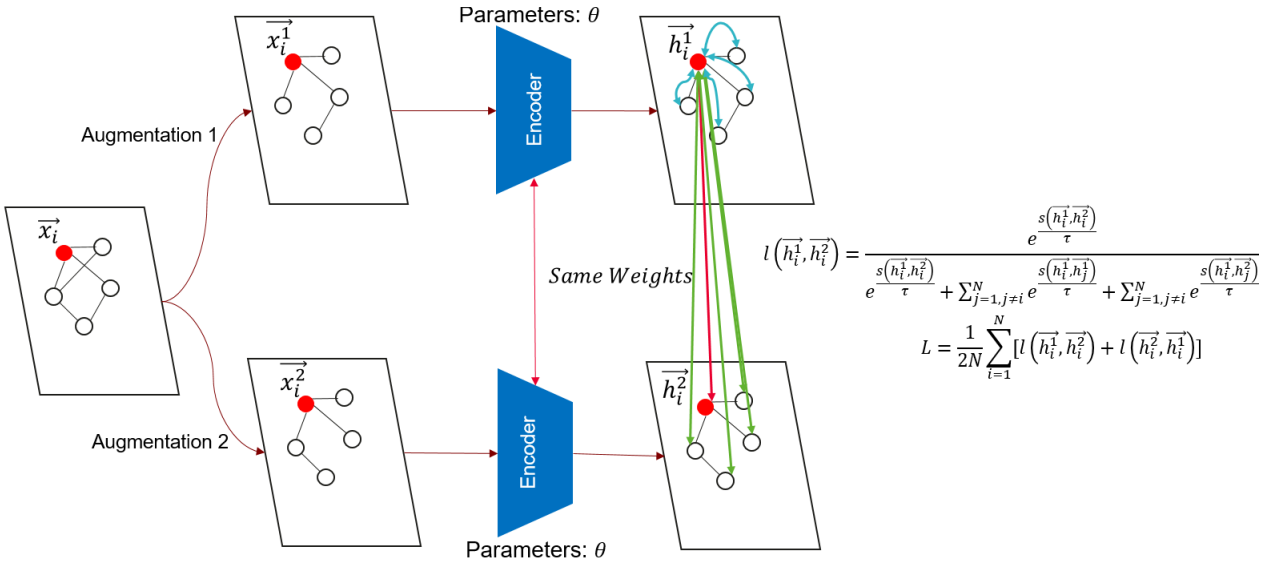

Figure 3.4: The overview of **GRACE**. Two views of the same graph are obtained through augmentation. Then, same-scale node-level contrast is applied to nodes. By doing that, the embedding of the same node in two versions become maximally similar to each other (Red Arrow) and minimally similar to the embedding of other nodes in the same view (Blue Arrows) or in other views (Green Arrows).

GRAph Contrastive rEpresentation learning (GRACE)

The overview of this technique depicted in Figure 3.4. This method, similar to other dual-encoder methods, obtain two versions of the graph by adopting data augmentation. The augmentation methods used in the paper are Feature Masking and Edge Drop. Then, the similarity between the embedding of the same node in two versions will be maximized (Red Arrow), while minimizing the similarity between different nodes of the same versions (Blue Arrows) and different versions (Green Arrows). The loss function minimized for each node in this method is as follows:

$$l(\vec{h_i^1}, \vec{h_i^2}) = \frac{e^{\frac{s(\vec{h_i^1}, \vec{h_i^2})}{\tau}}}{e^{\frac{s(\vec{h_i^1}, \vec{h_i^2})}{\tau}} + \sum_{j=1, j\neq i}^{N} e^{\frac{s(\vec{h_i^1}, \vec{h_j^1})}{\tau}} + \sum_{j=1, j\neq i}^{N} e^{\frac{s(\vec{h_i^1}, \vec{h_j^2})}{\tau}}} \tag{3.1}$$

where N is the number of nodes in graph, $s(.,.)$ is the cosine similarity between two vectors, τ is the temperature controlling the sensitivity of loss function into different levels of dissimilarity (smaller τ leads to higher sensitivity), and $\vec{h_i^k}$ is the embedding of node i in the k^{th} view. The total loss will be the summation over these loss terms for all nodes in both views of the graph:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} [l(\vec{h_i^1}, \vec{h_i^2}) + l(\vec{h_i^2}, \vec{h_i^1})] \tag{3.2}$$

As this method is inspired by its counterpart in computer Vision literature called SimCLR [6], it suffer from the same drawbacks of SimCLR which is requiring large memory footprint. As it will be shown later, GRACE is only applicable for small to mid-sized datasets which are of less interest especially in SSL literature.

Multi-View Graph Representation Learning (MVGRL)

This method, similar to GRACE, adopts negative sampling as a mechanism for collapse prevention. However, it uses Diffusion Graph as the augmentation method and also adopts cross-scale contrasting between embedding of graph and nodes [21]. As this method requires large memory capacity, it is not applicable even to mid-size datasets. Therefore, it is excluded from the baseline results in the following chapters.

Deep Graph Infomax (DGI)

This contrastive method proposed by Velickovic et. al [63]. The main motivation behind the paper is that most existing methods for self-supervised learning on graphs are based on node-level contrasting, which may not capture the full richness of the graph structure. The authors argue that mutual information maximization can be extended to graph-structured data by considering the mutual information between the graph and a set of global features. To implement this idea, the authors propose a neural network architecture called Deep Graph Infomax (DGI). The DGI model consists of two parts: a backbone graph encoder and a context encoder. The graph encoder takes a graph as input and outputs a set of node representations. The context encoder takes a set of global features as input and outputs a context representation. The mutual information between the two representations is then maximized using a contrastive loss function, negative sampling. As it will be shown later, this method does not perform better or even on par with other methods for node classification in single static graph datasets.

3.5.2 Non-Contrastive SSL Methods for Graphs

Non-contrastive methods prevent the collapse in representation with other mechanisms which are more efficient than negative sampling in memory footprint, computational complexity, or both. Some of these methods can be listed as stop-gradient in one branch, weight sharing via updating one encoder by delayed version of the other encoder (Exponential Moving Average mechanism or EMA), or covariance/cross-covariance whitening. In the following sub-sections, some important recent methods in Graph Deep Learning literature under NCSSL category are listed and explained. These methods are Bootstrapped Graph Latents (BGRL) [58], Graph Barlow Twins (GBT) [4], and our implementation of Variance-Invariance-Covariance Regularization (VICReg) [3] for graphs.

Bootstrapped GRaph Latents (BGRL)

This method is inspired from a well-known SSL method in Computer Vision called BYOL [19] which prevents collapse in representation by imposing asymmetry in dual-encoder architecture. The overview of this method is depicted in Figure 3.5. As it can be seen, one encoder will be considered as an online encoder and the other one the target encoder. There are three types of asymmetry in this scheme: 1. Stopping gradient in the target encoder's branch 2. Using predictor module in online (student) branch 3. Instead of weight sharing, the weights of the target encoder are updated by using exponential moving average mechanism. In other words, the new weights of the target model will be the combination of the previous weights with he new weights of the online encoder. The loss function in this method is the negative of the inner product between predictions from the online encoder branch and the output of the target encoder.

Although this method is significantly faster than NCSLL methods based on Covariance/Cross-Covariance whitening, it performs worse than them in almost all benchmark datasets in Graph literature. Chapters 5 and 6 of this thesis are dedicated to improving this method and pushing its performance to covariance-based counterparts while adding minimal computational complexity. The improvements proposed in those two chapters are based on weight regularization and early stopping the teacher's network training inspired from Knowledge Distillation [24] Literature.

Graph Barlow Twins (GBT) [4]

This method is inspired from a Self-Supervised Learning method proposed for Computer Vision tasks called Barlow Twins [73]. The idea stems from the redundancy-reduction

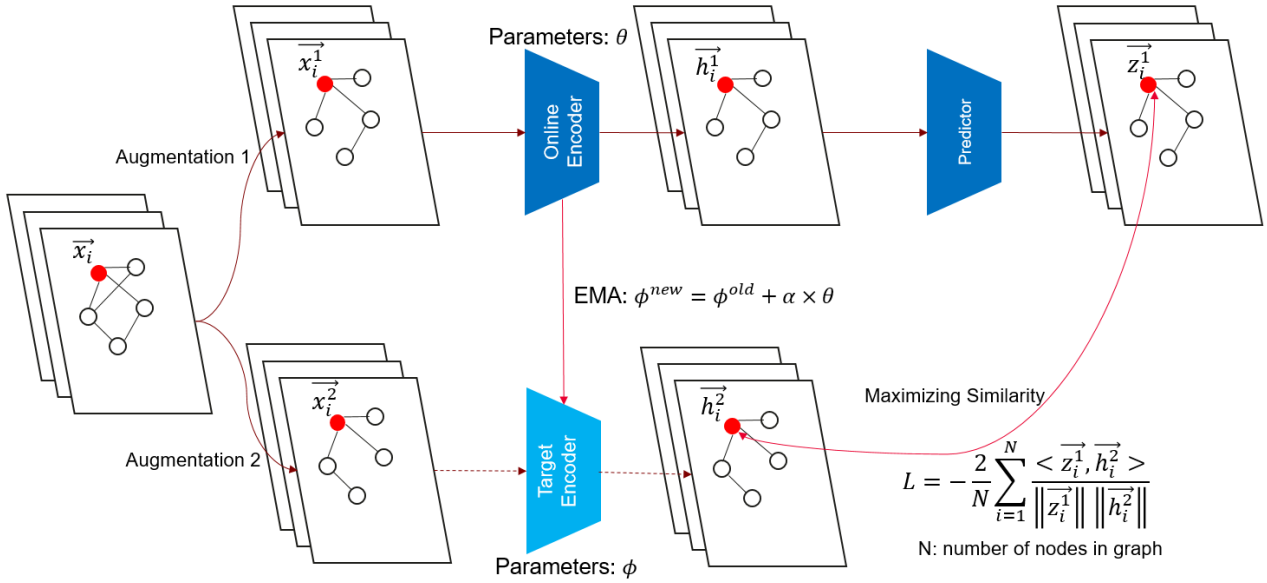

Figure 3.5: The overview of **BGRL**. Two views of the original graph obtained by augmentation. One of them is passed through online (student) encoder and the other one through the target (teacher) encoder. The weights of online encoder change by back-propagation, while the weights of target encoder updated by performing Exponential Moving Averaging(EMA). Finally, the embedding of the upper branch is passed through a predictor module which predicts the target embedding. The objective is maximizing the cosine similarity between $\vec{z_i^1}$ and $\vec{h_i^2}$. Dashed lines show stop-gradient.

principle proposed by Neuroscientist H. Barlow. It argues that to remove the redundancy in embedding space, we have the highest level of redundancy in the case of collapse in representation, it is enough to perform whitening on the cross-covariance matrix calculated by outer product of embeddings for two views. This could be viewed as negative sampling on dimensions instead of nodes. Recently, it has been shown that performing negative sampling in dimension space is similar to performing negative sampling in sample space [16]. The overall architecture of Graph Barlow Twins is shown in Figure 3.6. The goal is minimizing the difference between identity matrix and the cross-covariance matrix. Before calculating the cross-covariance matrix, embedding vectors will be normalized to have mean zero and the standard deviation equal to one. Then each index of cross-covariance matrix

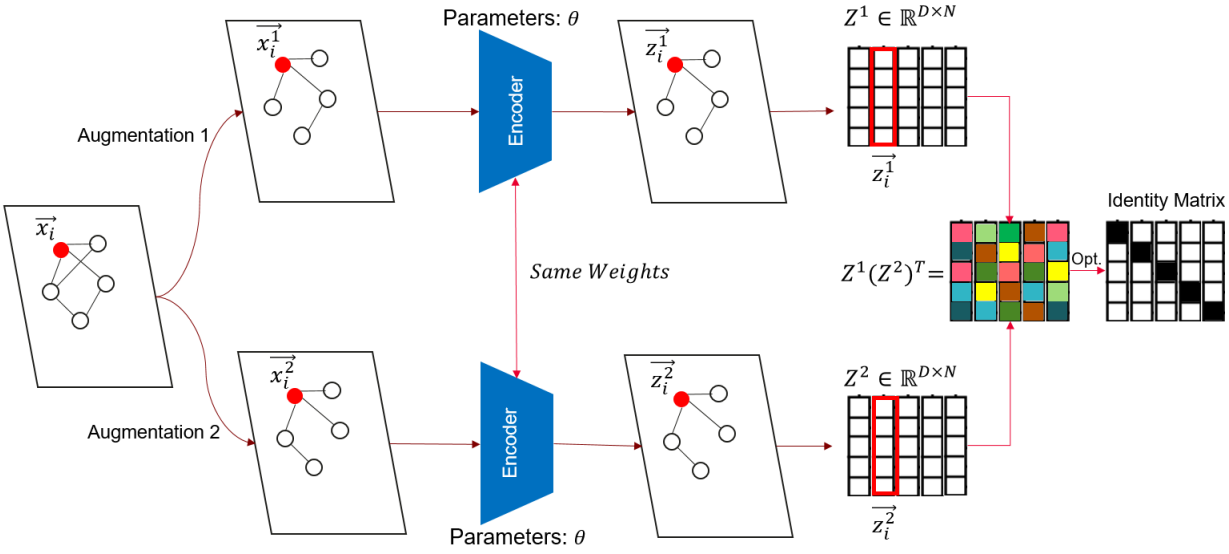

Figure 3.6: The overview of **GBT**. Given a static graph, two views of that are produced by augmentation and encoded into representations $\vec{z_i^1}$ and $\vec{z_i^2}$. If we concatenate these embeddings we will get two matrices Z^1 and Z^2. Then we need to normalize rows to have mean zero and variance one, we can calculate the cross-covariance matrix as follows $Z^1(Z^2)^T$. The goal is making this cross-covariance matrix close to identity, whitening procedure.

can be calculated as follows:

$$C_{i,j} = \frac{< \vec{z_i^1}, \vec{z_j^2} >}{||\vec{z_i^1}||\,||\vec{z_j^2}||} \tag{3.3}$$

where $\vec{z_i^k}$ is the embedding of the i^{th} node in the k^{th} view of the origianl graph. The total cross-covariance whitening loss will be as follows:

$$\mathcal{L} = \sum_{i=0}^{D}(1 - C_{ii}^2) + \lambda \sum_{i=0}^{D}\sum_{j\neq i} C_{ij}^2 \tag{3.4}$$

GBT outperforms GRACE and BGRL in all datasets. However, if we want to have large output dimension, its complexity is significantly higher than BGRL. GBT also has the following drawbacks:

1. Both branches must be matched. Therefore, it cannot be used in multi-modal cases or in the cases that we want to train two encoders with different capacities [3].

2. It only decorrelates dimensions instead of making them independent. Therefor, it is sub-optimal.

The next method which is going to be introduced is VICReg [3] which outperforms all other methods and also solve two mentioned problems in GBT. VICReg, to the best of our knowledge, has not been implemented for graphs. In this thesis, the results of VICreg trained on different graph datasets will be shown and improved.

Variance Invariance-Covariance Regularization (VICReg) [3]

The overall structure of this method is shown in Figure 3.7 .The idea behind this method is similar to Barlow Twins and it is based on the redundancy-reduction-principle indicating that we can prevent collapse by maximizing the information content of embedding space. Barlow Twins and Graph Barlow Twins papers achieve that by whitening the cross-covariance matrix whitening. However, VICReg achieves that by three loss terms:

1. Variance (V): This loss term, by the help of hinge loss, tries to keep the variance of each embedding variable maintained above a threshold.

2. Invariance (I): This term tries to minimize the distance between two embedding calculated from two views.

3. Covariance (C): Tries to decorrelate the pairs of embedding variables. In other words, it tries to minimize the norm difference between identity matrix and the covariance matrix of embedding variables.

However, VICReg has some major differences compared to Barlow Twins. First of all, it calculates covariance of embedding variable for each branch separately, in contrast with Barlow Twins which calculates cross-covariance of branches. This way of covariance calculation provides VICReg with the possibility of training multi-modal (different types of data in different branches) or different types of encoders. Second of all, VICReg training architecture contains a module called expander which maps the output of encoder into higher dimensional space. As the output of this module has significantly higher dimension than the output of encoder and it maps data non-linearly into the higher dimensional space, it makes dimensions not solely decorrelated though independent as well.

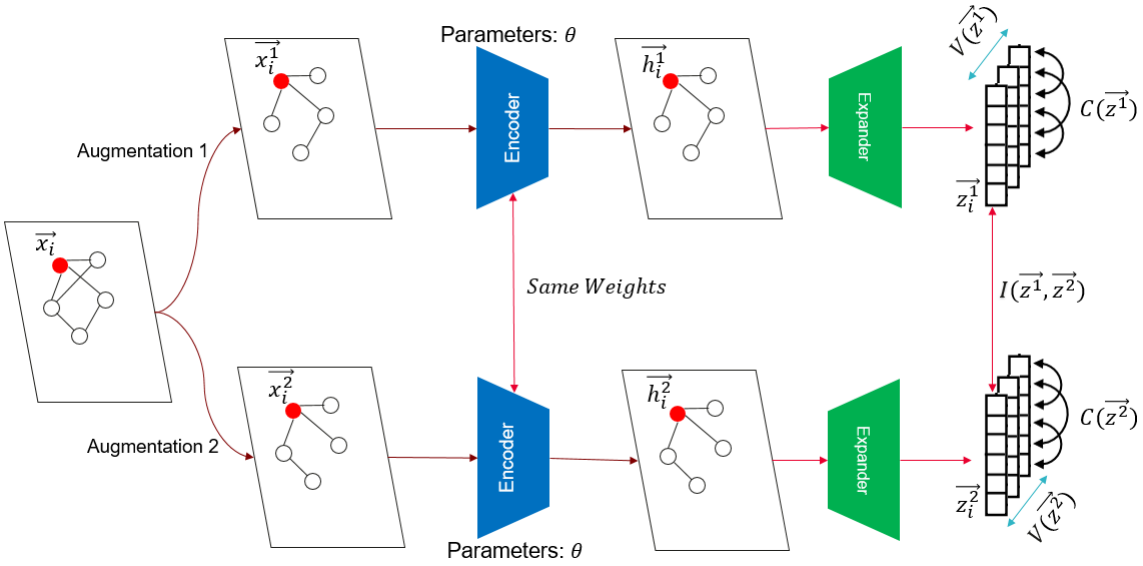

Figure 3.7: The overview of **VICReg**. Given a static graph, two views of that are produced by augmentation and encoded into representations $\vec{h_i^1}$ and $\vec{h_i^2}$. Then, the representations are fed to an expander module $\vec{z_i^1}$ and $\vec{z_i^2}$. This module, up-project the output of encoders into a space with higher dimension. Finally, the distance between two embeddings from the same node in the graph is minimized, the variance of each embedding variable is maintained above a threshold, and the covariance between pairs of embedding variables are attracted to zero, which results in decorrelating the variables from each other.

VICreg has been proposed an implemented for image datasets. One of the contributions of this thesis is testing VICReg on graph datasets. It will be shown that VICReg outperforms all of the mentioned successful baselines. However, it is more complex than them. In chapter 4, different methods will be proposed which can improve efficiency and the performance of VICReg for graph deep learning. Although, ideas have been tested on graph datasets, those are applicable in any other applications and modalities.

3.6 Detail of Graph Datasets

There are some publicly available static graph datasets which are frequently used in Graph Self-Supervised Learning papers as benchmarks. The detail of these datasets summarized

	Nodes	Edges	Features	Classes
Cora	2,708	10,556	1,433	7
WikiCS	11,701	216,123	300	10
Amazon Computers	13,752	245,861	767	10
Amazon Photos	7,650	119,081	745	8
Coauthor CS	18,333	81,894	6,805	15
Coauthor Physics	34,493	247,962	8,415	5

Table 3.1: Detail of Graph Datasets

in Table 3.1. These datasets contain some nodes, edges, initial feature vectors, and classes. The detailed explanation of these datasets are as follows:

1. Cora: Presented by Yang et. al. [71]. In this dataset, nodes represent documents and edges represent citations. Features are bag-of-words representation of documents.

2. WikiCS: Presented by Mernyei and Cangea [42]. The dataset Wikipedia-based and it consists of nodes corresponding to Computer Science articles, with edges based on hyperlinks and 10 classes representing different sub-fields of Computer Science.

3. Amazon Photo/Computers: Presented by Shchur et. al. [51]. Nodes represent products and edges represent that two products are frequently bought together. Given product reviews as bag-of-words node features, and class as the category of product.

4. Coauthor CS/Physics: Presented by Shchur et. al. [51]. In this graph, authors are represented as nodes, which are connected by edges if they have co-authored a paper. The node features of each author represent the paper keywords of their respective papers. Furthermore, each node has a class label that indicates the most active fields of study for the author.

3.7 Downstream Tasks

In static graph literature, tasks of interests are node classification or edge prediction. In the first task, the goal is finding out the label of node. For example in Coauthor datasets, the task is detecting the field of the author by having the links of paper coauthored with

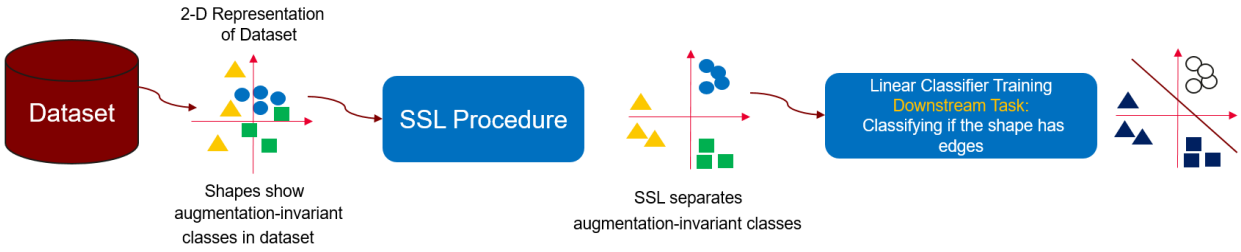

Figure 3.8: Linear probing procedure for evaluating Self-Supervised Learning Methods. The SSL procedure separates the classes which are augmentation-invariant. Then with a freezed encoder, freezed representation, a linear classifier will be trained to learn labels for a specific downstream task. For example, in this figure the task is classifying if the shape has edges.

other scholars. In link prediction, the goal is predicting if between two nodes there is an edge or not. If the dataset contains multiple graph, one other task of interest could be classifying the graph.

In this thesis, the focus is on node classification. The goal is finding a proper representation for the graph by using Self-Supervised Learning methods. Then, using the encoder and representation learnt by that as the embedding space for training a linear classifier. If the encoder has learnt a proper representation, a linear classifier must be able to classify nodes properly as the manifold of data has been unfolded. Therefore, the general procedure in all experimental results brought in the next chapters depicted in Figure 3.8. This procedure is called Linear Probing which is one of the methods for evaluating the quality of representation learnt by SSL procedure. The procedure separates the classes which are augmentation-invariant. The learnt encoder will be freezed and a linear classifier will be trained to learn labels for a specific downstream task. For example, in this figure the task is classifying if the shape has edges. In graph datasets, the downstream task could be classifying the node. It can be seen from the figure that if the augmentation-invariant classes are not aligned with the downstream task, SSL procedure will never lead to a proper representation [12]. For example, if the task is detecting location of the object and augmentation is cropping, it is not possible to achieve proper downstream performance by SSL procedure because of the type of augmentation.

F1 Micro and Macro for the node classification task has been used as measures of the effectiveness of the linear classification model (linear probing procedure). F1 micro is calculated by considering the total number of true positives, false positives, and false

negatives across all classes. This means that it gives equal importance to each observation regardless of its class. F1 macro is calculated by taking the average of the F1 scores for each individual class. This means that it treats each class equally, regardless of its frequency in the data. The proposed method in this thesis provides different levels of improvement for F1 Micro/Macro. Therefore, both are reported in all tables.

3.8 Chapter's Conclusion

In this chapter, different types of augmentation and scale of contrast in SSL methods developed for graphs were discussed. Then, some important baselines were introduced and compared to each other. Finally, the datasets and downstream task chosen for experiments in the next chapters were introduced. In summary, for all experiments in the next chapters, the augmentation type will be the combination of Edge-Drop and Feature-Masking. The scale of contrast will be Same-Scale Node-Level. The baselines for comparison will be GRACE, BGRL, GBT, GVICRec (implementation of VICReg for graph datasets). Also, BGRL and GVICReg are chosen to be improved. The datasets will be Cora, WikiCS, Amazon Photo, Amazon Computers, Coauthor CS, and Coauthor Physics. The downstream task will be Node Classification which is the most important task in graph literature for single graph datasets. The method for evaluating the baseline SSL methods and proposed ideas is linear probing with fixed encoder and the metrics will be F1 Micro/Macro. In the next chapters, first some methods for improving GVICreg in the sense of performance and efficiency will be proposed (Chapter 4). Then, two techniques will be proposed and tested for improving BGRL (Chapters 5 and 6). Finally, the best results from different chapter will be compared and some future directions will be proposed (Chapters 7 and 8).

Chapter 4

Improving The Efficiency and Performance of Covariance Based Non-Contrastive Self-Supervised Learning Methods

4.1 Chapter's Overview

In this chapter, different methods are proposed and justified which can be used to improve the efficiency of Non-Contrastive Self-Supervised Learning methods which are based on Covariance/Cross-Covariance whitening. These proposed ideas are tested on VICReg [3] and used for performing SSL for different benchmark graph datasets. The proposed techniques have resulted in performance improvement to VICReg, while being 2 times faster.

4.2 Introduction

As discussed in the previous chapters, one of the mechanisms for collapse prevention in NCSSL methods is covariance/cross-covariance whitening. It was also mentioned that covariance-based methods outperform other types of NCSSL in various applications and scenarios. However, their complexity is significantly higher than non-contrastive methods adopting other mechanisms for preventing collapse since covariance-based techniques are

based on pairwise dimension vector orthogonalization which leads to quadratic complexity with respect to the dimension of the output.

In this chapter, VICReg [3] has been chosen as the representative of covariance-based methods and its performance in representing graph datasets will be experimented. The issue of efficiency is a critical concern in VICReg since it has been shown in the original paper and it will be re-examined in this thesis as well that the output dimension of the Expander module must be significantly large. This alongside the quadratic complexity of the covariance loss term with respect to the dimension shows the necessity of adopting methods for improving its complexity. In the following sections, the performance of VICReg in graph deep learning will be experimented, the importance of having a large output dimension for the VICReg will be shown, and finally, methods for improving its performance and efficiency will be proposed. The contributions of this thesis are as follows:

- Evaluating the VICReg SSL method for node classification task in benchmark static graph datasets.
- Evaluating the effect of the expander module and increasing its output dimension.
- Proposing Methods for improving the efficiency and performance of VICReg.

4.3 Methodology

This section aims to analyze the loss terms used in VICReg, along with their corresponding complexity. Furthermore, novel methods for improving the complexity and performance of VICReg will be proposed. These methods have demonstrated better performance and significantly lower computational complexity compared to VICReg in some cases.

4.3.1 Overview on VICReg loss function and training procedure

The overview of VICReg training procedure for graph datasets is shown in Figure 4.1. The training procedure and scheme is similar to Graph Barlow Twins (GBT) [4] with some main differences which are as follows [3]:

1. In graph Barlow twins, the decorrelation is done between the embedding variables cross two views. In other words, it is based on whitening the cross-covariance matrix. In contrast, VICReg performs covariance whitening independently on each branch,

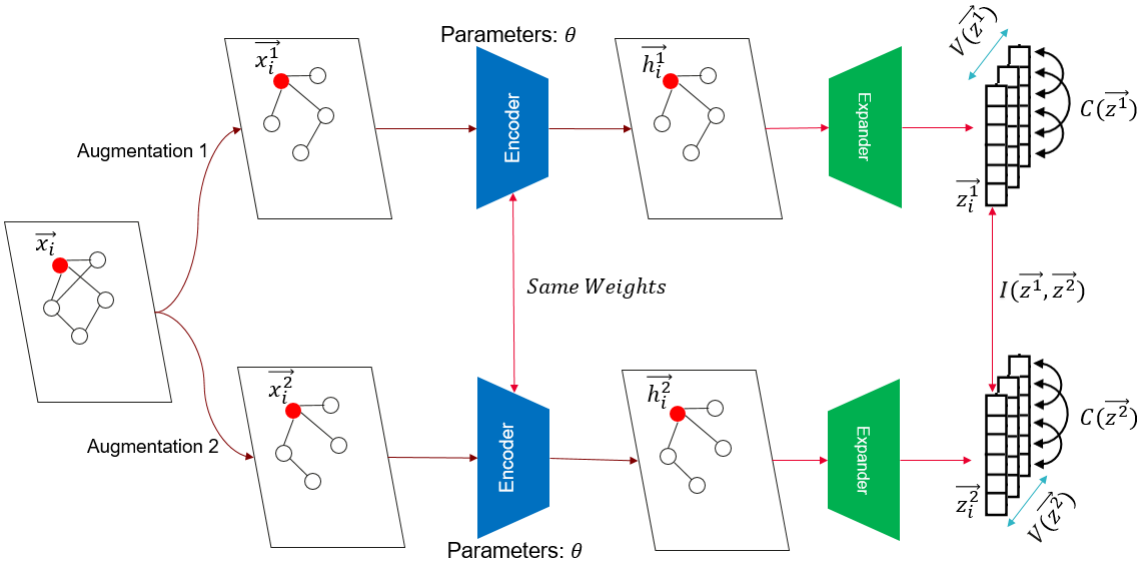

Figure 4.1: Overview of the training procedure. First, two views of a static graph are generated through augmentation, and then each node are encoded into two separate representations, $\vec{h_i^1}$ and $\vec{h_i^2}$ are the embedding of node i after this step. These representations are subsequently fed to an expander module. This expander module up-projects the output of encoders into a space with higher dimension resulting in $\vec{z_i^1}$ and $\vec{z_i^2}$ for node i. Finally, the distance between two representation of the same node in two views of the graph is minimized, the variance of each embedding variable is maintained above a threshold over representations of nodes in two different view separately, and the covariance between pairs of embedding variables become zero, which results in decorrelating the variables from each other.

allowing for multi-modal SSL and the use of different encoder architectures in each branch.

2. In VICReg, there is an extra module called "Expander" which maps data into high-dimensional space with a non-linear learnable mapping. This non-linear mapping into high-dimensional space, correlation becomes an indication of dependency. Therefore, de-correlating dimensions in that space leads to pairwise independence of them.

In Figure 4.1, three loss terms are shown. These terms are calculated as follows:

1. Variance (V): This loss term, by the help of hinge loss, keeps the variance of each

embedding variable maintained above a threshold. Mathematically:

$$V(\vec{z^k}) = \frac{1}{D}\sum_{j=1}^{D} max\Big(0, \gamma - \sqrt{Var(\vec{z^k}_{(.,j)})}\Big) \tag{4.1}$$

where $\sqrt{Var(\vec{z^k}_{(.,j)})}$ corresponds to the standard deviation of j^{th} variable of embedding over all nodes in the graph and D is the expander's output dimension. The complexity of this term is of order $O(ND)$.

2. Invariance (I): Minimizes the distance between two embeddings calculated from two views for the same node in the graph. Mathematically:

$$I(\vec{z^1}, \vec{z^2}) = \frac{1}{N}\sum_{i=1}^{N} ||\vec{z_i^1} - \vec{z_i^2}||_2 \tag{4.2}$$

where N is the number of nodes in the graph and $\vec{z_i^k}$ is the embedding of node i calculated based on the k^{th} view of the graph. The complexity of this term is of order $O(ND)$.

3. Covariance (C): Decorrelates the pairs of embedding variables. In other words, it tries to minimize the norm difference between identity matrix and the covariance matrix of embedding variables. Mathematically, if we define the covariance of embedding variables as follows:

$$Cov(\vec{z^k}) = \frac{1}{N-1}\sum_{i=1}^{N}(\vec{z_i^k} - \bar{z^k})(\vec{z_i^k} - \bar{z^k})^T \tag{4.3}$$

where $\vec{z_i^k} \in \mathbb{R}^{D\times 1}$, and $\bar{z^k} = \frac{1}{N}\sum_{i=1}^{N}\vec{z_i^k}$. Therefore, the loss term will be as follows:

$$C(\vec{z^k}) = ||Cov(\vec{z^k}) - I_{D\times D}||_F \tag{4.4}$$

Where $||.||_F$ corresponds to Frobenius norm. As the variance is controlled by Variance loss term shown in equation 4.1, diagonal terms should be discarded. Therefore, simplified covariance loss term can be written as follows:

$$C(\vec{z^k}) = \frac{1}{D}\sum_{i\neq j} Cov(\vec{z^k})_{i,j}^2 \tag{4.5}$$

the complexity of this term is of order $O(ND^2)$.

By putting all three loss terms together for two views, the total loss can be written as follows:

$$\mathcal{L} = \lambda I(\vec{z^1}, \vec{z^2}) + \mu[V(\vec{z^1}) + V(\vec{z^2})] + \zeta[C(\vec{z^1}) + C(\vec{z^2})] \tag{4.6}$$

where λ, μ, and ζ are hyperparameters controlling the importance of each term in training.

4.3.2 Reducing Complexity by Improving Node Efficiency

To calculate the three terms shown in Equation 4.6, considering all nodes of the graph is not necessary. We can estimate these terms by sampling a fraction of nodes during each epoch the same as recent methods in Curriculum Learning showing bootstrapping data will not lead to significant degradation of performance if we repeat sampling during each epoch [79]. It must be noted that all nodes are present during the message passing and graph encoding. However, only a fraction of them are used to calculate the loss terms. In this section, uniform sampling and sampling based on Forman Ricci Curvature are proposed as two methods for reducing the complexity via sampling nodes to be used in loss terms calculation. If we consider ρ as the node sampling ratio, the complexity of all three terms will be reduced by factor ρ. To sample nodes, we need a metric which is a valid probability distribution showing the importance of each node. The sampling probability for the i^{th} node is named Pr_i. In the next two sub-sections, two metrics which can be transformed into the sampling probability distribution are proposed and discussed.

Uniform Node Sampling

As a baseline node sampling method, uniform node sampling is considered. In this scenario, during each epoch, a ρ fraction of nodes will be chosen based on uniform probability distribution, $Pr_i = \frac{1}{N}$, where N is the number of nodes in graph. It means that, all nodes will have the same chance to be chosen. After that, three loss terms will be calculated considering only those nodes' embeddings. The important point is that the sampling will be repeated in the beginning of each epoch. As a result, all nodes will have chance to contribute in the loss calculation during next epochs if those have not be chosen for an epoch.

Node Sampling Based on Forman Ricci Curvature

To improve the performance of uniform sampling, we can adopt more sophisticated sampling procedures. One idea would be sampling nodes based on the amount of change in

their embedding after different graph augmentations since if the change is small, that node will not have a significant contribution in the loss term calculation. However, the exact calculation of this metric requires evaluating encoder for all nodes during all epochs, which leads to computationally expensive sampling procedure. Therefore, we need a metric which shows the sensitivity of the embedding of a node by only considering the original graph's structure and it can be calculated before starting the SSL procedure. Forman Ricci Curvature proposed by [14] has been chosen as the sampling metric. However, this method is biased towards the negative curvatures [60]. Therefore, the balanced methods proposed by [60] were used to calculate the Ricci curvature of each edge. Then, the total Ricci metric for each node will be calculated as the summation of Forman Ricci Flow of edges connected to that node. According to the theorems and observations by [60], edges with negative Ricci Flow act as bottlenecks that hinder the flow of information between different parts of the graph. Based on this, the hypothesize is that nodes with negative or small positive Ricci flows will experience less significant changes to their embeddings after the augmentation procedure. To obtain a valid probability distribution for the sampling metric, the Ricci flows of nodes were normalized by shifting them to the right to obtain non-negative values, and then normalizing them such that they sum to one. In other words, the probability of choosing node i is $Pr_i = \frac{Ricci(Node_i) - min[Ricci(.)]}{\sum_{j=1}^{N} Ricci(Node_j)}$, where $min[Ricci(.)]$ is the minimum Ricci curvature among all N nodes of the graph. By adopting the probability distribution obtained from normalizing the Ricci flows, nodes will be sampled less frequently if they have negative or small positive Ricci flows. Further justification for using the Ricci flow as the sampling metric is provided in Section 4.4.1. Specifically, this section demonstrates a strong and positive correlation between changes in the embedding after augmentations and the Ricci flow metric.

4.3.3 Reducing Complexity by Improving Dimension Efficiency

The complexity of the covariance loss term is quadratic with respect to the output dimension of expander module. As this dimension should be high in order to improve performance, reducing the complexity of covariance whitening term is essential. In this section, four methods are introduced which can decrease the complexity of covariance term significantly without causing degradation in performance. Before introducing these methods, it should be noted that the covariance loss term tries to make embedding variables pairwise orthogonal. The main goal behind all of the following ideas is doing this orthogonalization with less computational cost.

Uniform Dimension Sampling

A simple approach to reduce complexity is to uniformly sample a fraction of output dimensions during each epoch and only make those vectors orthogonal. The rationale behind this sampling method is that orthogonalization does not need to be performed on all pairs of vectors during all epochs. Instead, performing orthogonalization on a fraction of embedding variable vectors during each epoch, and repeating the sampling process, may lead to nearly the same level of performance. In this scenario, we sample a fraction β of dimensions based on uniform probability $Pr_i = \frac{1}{D}$, where D is the output dimension of the expander. Importantly, this sampling must be repeated at the start of each epoch to ensure that all embedding variables have an equal chance to contribute to the loss terms calculation.

Dimension Bucketing With Locality Sensitive Hashing (LSH)

Another method for reducing the complexity of pairwise orhogonalization is using fast methods to find vectors which have large similarity. One method for doing that is using Locality Sensitive Hashing (LSH). To perform LSH, m random hyperplane will be passed in the N-dimensional space, where N is the number of nodes in the graph, and it will be checked in which side of those planes each vector lie on. It can be determined by considering the sign of the inner product between the hyperplane's normal vector and the vector which should be placed in a bucket. If the sign is negative for the m^{th} hyperplane, we consider "0" as the m^{th} bit corresponding to that specific vector and "1" otherwise. In the end, the decimal number from those m bits will be the index of bucket at which the vector should be placed. Finally, pairwise orthogonalization is done only between vectors inside each bucket instead of all D vectors. In other words, if buckets contain the same number of vectors, the complexity of the covariance loss term will be $O(N(\frac{D}{2^m})^2) + BucketingCost = O(N(\frac{D}{2^m})^2) + O(mND)$. This procedure is depicted in Figure 4.2. For the sake of visualization, the procedure is shown in 2-dimensional scenario. More detailed justification for this method can be found in section 4.4.2.

Random Projection

Another method for decreasing the complexity of covariance whitening procedure is using random projection. Instead of orthogonalizing D vectors in N dimensional space, we can down-project them into M dimensional space, where $M << N$, by adopting random projection and performing orthogonalization in that space. By doing that, the complexity

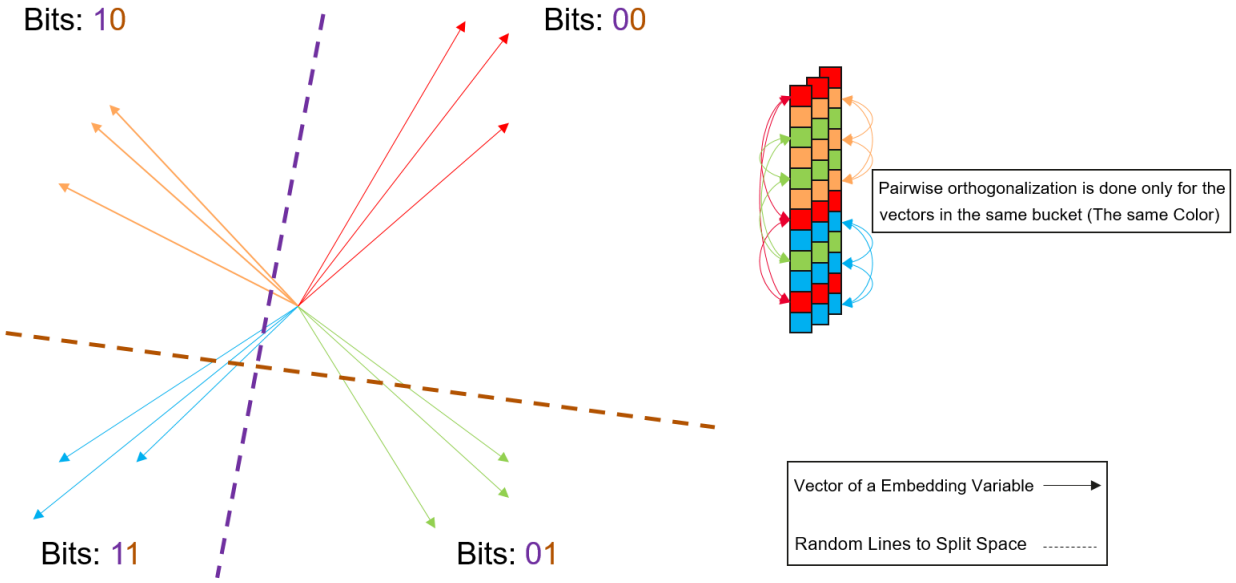

Figure 4.2: The procedure of using LSH in order to determine for which vectors the pairwise orthogonalization should be done. Orthogonlaization will be done for the N-dimensional vectors with the same bucket index (have the same color in this figure).

of covariance term will be reduced to $O(MD)$. The reason behind using random projection is that the pairwise angles will be preserved after projection. To see more detailed justification behind this method, refer to Section 4.4.3.

In this thesis, random projection has been adopted in two ways:

1. Randomly projecting embeddings from $D \times N$ to $D \times M$ where $M << N$. By doing that, embedding of nodes will be combined and the number of nodes will be reduced in all three loss terms. In other words, it is similar to randomly combining nodes' embedding in order to get new artificial nodes which are combined version of actual nodes.

2. Random projection is only performed for the covariance loss term in order to perform orthoganilzation in a space with significantly smaller dimension compared to the number of nodes in graph.

In both cases, to perform random projection, a random matrix with dimensions $N \times M$ is generated whose elements are drawn from standard normal distribution in and i.i.d manner.

To preserve the norm as well, all elements must be divided by $\sqrt{M}$. Random matrix with Guassian elements, which has been used in this thesis, is one type of random projection. Trying different types of random projection in this scenario left to future work.

Nystrom Approximation

As the covariance matrix is a positive semi-definite matrix, methods for approximating this type of matrices can be adopted to reduce the cost of covariance matrix calculation. One method for approximating positive semi-definite matrices is Nystrom approximation. It states that we can reconstruct a symmetric positive semi-definite matrix by having access to only two blocks from that matrix [69]. This method of approximation has primarily been adopted in literature to save memory. It achieves this by only storing selected portions of a semi-definite matrix. Additionally, it can enhance efficiency by re-ordering the order of multiplications in cases such as the attention mechanism [61] where a positive definite matrix is multiplied by a vector [70]. Nystrom approximation is used in this thesis to reduce the covariance whitening complexity as this approximation can be seen from the angle of choosing landmark and non-landmark vectors and performing the iterative orthogonalization for each group separately. For detailed Justification, refer to Section 4.4.4.

To formulate the method mathematically, we write covariance matrix as follows:

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

Nystrom approximation states that the C block in this matrix can be written and approximated by the help of A and B blocks. In other words, it can be written as follows [69]:

$$Cov_{D\times D} = \left[\begin{array}{c|c} A & B \\ \hline B^T & C \end{array}\right] \approx \left[\begin{array}{c|c} A & B \\ \hline B^T & B^T A^{\dagger} B \end{array}\right]$$

where $A \in \mathbb{R}^{M\times M}$, $B \in \mathbb{R}^{M\times(D-M)}$, D is the number of embedding dimensions, $M << D$, and $A^{\dagger}$ is the pseudo-inverse of block A. In our scenario, as the number of nodes in graph is usually larger than M, A does not have rank deficiency and therefore we can replace pseudo-inverse with inverse. This formulation can be used in three ways:

1. Making the A block identity and also reducing the Frobenious norm of B block while making it orthonormal $B^T B \approx I$.

2. Only Making block matrices A and $B^T A^{-1} B$ identity.

3. Making the A block identity, and minimizing the norm of B block.

The first option leads to almost the same complexity compared to vanilla VICReg for M values significantly smaller than D, and even higher complexity for not significantly small M values as the cost of inverting A will be dominant. The second option decreases complexity to $O(NM^2) + O(N(D-M)^2)$ which is a small reduction in the cost for small M. The last option provides the maximum save in time. However, that makes the norm of $B^T A^{-1} B$ zero which increases the norm of the difference between the covariance matrix and identity matrix. The results of the first and the second methods will be shown in the results section.

To perform the first or second option, during each epoch, M out of D embedding variables will be chosen uniformly as the landmarks, A block. This sampling repeats in the beginning of all epochs. Other $D - M$ vectors will be non-landmarks. Then, covariance can be calculated as follows:

1. A: Pairwise inner products of landmark vectors.

2. B: Pairwise inner products between landmark and non-landmark vectors.

After calculating these two blocks, we can use them in order to adopt any of mentioned techniques for covariance matrix whitening.

4.4 Justification Behind Proposed Methods

4.4.1 Justification Behind Using Ricci Flow as the Sampling Metric for Nodes

The sign of Ricci flow shows if the shape of a part of manifold is hyperbolic or spherical. As shown in Figure 4.3, if the Ricci flow is negative, the shape is hyperbolic and spherical in other case. Therefore, bottlenecks of a manifold or a graph can be detected by calculating Ricci flows. The edges with negative Ricci flows are the bottlenecks [60]. These bottleneck

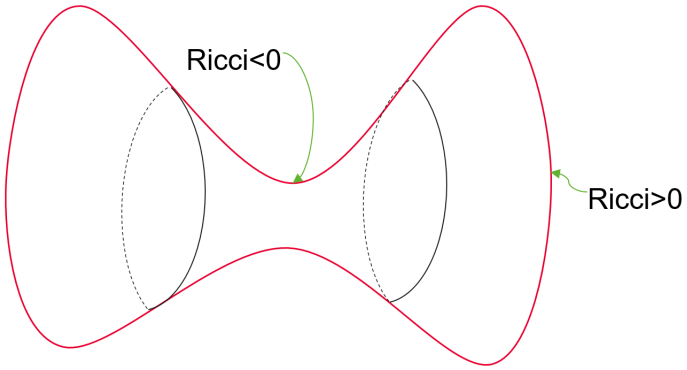

Figure 4.3: The sign of Ricci flow indicates if the specific part of manifold is Hyperbolic (bottleneck) or Spherical.

edges hinder the flow of information during the message passing in Graph Neural Network (GNN) [60]. In other words, if a node is connected to many edges with negative Ricci flow, the change in embedding and structure of different parts of graph will not be propagated to that node. Therefore, nodes connected to many bottleneck edges can be neglected from VICReg loss term as their contribution in the invariance term is limited and negligible since their embedding will not change significantly after augmentations.

In the proposed sampling method based on Ricci flow, first the balanced Forman Ricci Flow [14, 60] is calculated for all edges. Then, the Ricci metric for each node will be calculated as the summation of Ricci Flows of Edges [60]. Finally, this metric will be normalized to be a valid probability distribution to be used in order to sample fraction of nodes during each epoch to be used in VICReg loss calculation. By doing so, we hypothesize that the nodes which will be sampled more frequently are the ones which their embedding will change significantly after augmentations. To test the hypothesis, the correlation between normalized Ricci Flow and the invariance loss term in VICReg loss function will be calculated. Then a connection between this correlation and the level of improvement in performance compared to uniform sampling will be made. Finally, a hypothesis will be tested arguing that the reason behind no improvement in some datasets by using Ricci flow as a metric for sampling could be explained by uniformness of the metric for those datasets.

Correlation Between the change in embedding after augmentation and ricci curvature

Correlation Between the change in embedding after augmentation and Ricci curvature is shown in Table 4.1. As it can be seen in the table, the correlation is always positive which is aligned with the hypothesis. Also, the correlation is low for two datasets, Cora and WikiCS, and it is aligned with the observation in Table 4.7 showing that sampling based on Ricci does not lead to better performance compared to uniform sampling or without sampling for these two datasets. We hypothesize that for these two datasets, the shape of manifold is spherical and uniform in almost all places and that is why uniform sampling perfroms at the same level as sampling based on Ricci flows.

Dataset	Correlation Value
Cora	0.056
WikiCS	0.139
Amazon-Photo	0.223
Amazon-Computers	0.250
Coauthor-CS	0.385
Coauthor-Physics	0.338

Table 4.1: Correlation between $Ricci(Node_i)$ and $I(\vec{z_i^1}, \vec{z_i^2}) = ||\vec{z_i^1} - \vec{z_i^2}||_2$

KL divergence of Ricci and uniform for different datasets

To test the mentioned hypothesis that the reason behind no improvement in performance for some datasets is the uniformity of Ricci flow in them, the KL divergence between probabilities from normalized Ricci Flows and uniform distribution are calculated. The results are shown in Table 4.2. It is verified that for Cora and WikiCS datasets, the distribution is closest to the uniform distribution compared to other datasets. However, to accept this hypothesis, graph rewiring methods should be adopted which can make other datasets spherical as well. This is left to future work.

Dataset	KL-divergence (Divided by 1000)
Cora	2.145
WikiCS	109.710
Amazon-Photo	168.449
Amazon-Computers	131.087
Coauthor-CS	360.461
Coauthor-Physics	180.073

Table 4.2: KL divergence between uniform distribution $Pr_i = \frac{1}{N}$ and distribution from Ricci flows $Pr_i = \frac{Ricci(Node_i) - min[Ricci(.)]}{\sum_{j=1}^{N} Ricci(Node_j)}$.

4.4.2 Justification Behind LSH Bucketing to Achieve Dimension Efficiency

The covariance term in Equation 4.6 tries to make vectors of embedding variables pairwise orthogonal. To reduce the cost, the goal is to find vectors which have smaller pairwise angles and perform the orthogonalization only between them. The reason is that the vectors with smaller pairwise angles are the dominant terms in covariance loss term as those have higher correlation values. To find such vectors, we can calculate all angles and perform clustering based on those angles. However, it will lead to the same complexity as making all vectors pairwise orthogonal. Therefore, we need a faster method to find geometrically close embedding variable vectors. LSH can find such vectors in the way explained before with minimal extra complexity. To see how this method detects vectors with small pairwise angles, it is helpful to look at LSH from the angle of random rotation instead of hyperplane passing. In Figure 4.4, it is shown how LSH works based on random rotation. Instead of considering LSH as a method which passes hyperplanes and finds the side on which vectors lie, it can be considered as rotating vectors randomly and considering the sign of components of resulted vector as the bits to indicate the index of bucket. After rotation and considering the location of vectors based on quarters, the vectors with the same bit string are the ones which have been in the same quarter after all those random rotations. Therefore, the angle between them is small.

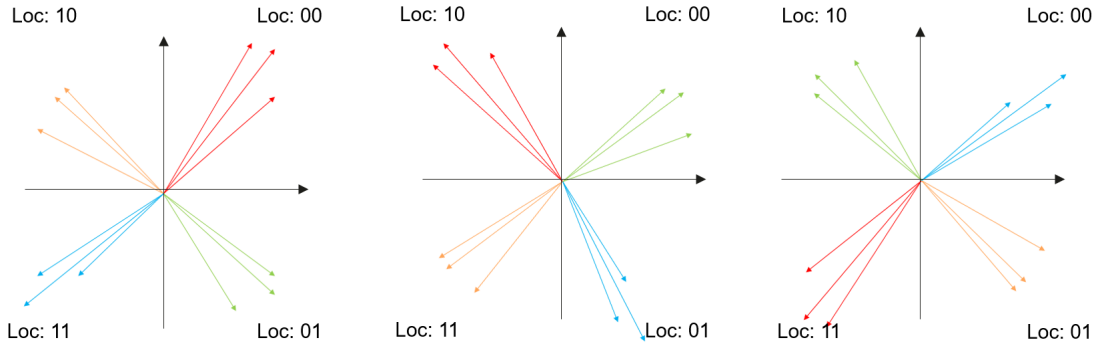

Figure 4.4: Explaining LSH from random rotation viewpoint.

4.4.3 Justification Behind Adopting Random Projection to Reduce the Cost of Pairwise Orthogonalization

Pairwise orthogonalzing of D N-dimensional vectors, especially if N is large, is costly. However, instead of performing orthogonalization in N-dimensional space, we can down-project vectors into M-dimensional space and perform orthogonalization in that space. If we have adopted Gaussian Random Projection for projecting vectors from N-dimensional space to M-dimensional one, pairwise angles will be preserved. Refer to Johnson–Lindenstrauss lemma [30] and Theorem 4.4.1.

Theorem 4.4.1 (Gaussian Random Projection Preserves Pairwise Angles [52]) *For any $w, x \in \mathbb{R}^N$, and any random standard Gaussian matrix $T \in \mathbb{R}^{N \times M}$ as defined in section 4.3.3, for any $\epsilon \in (0,1)$, if $< w, x >> 0$, then with probability at least*

$$1 - 6e^{\left(\frac{-M}{2}(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3})\right)} \tag{4.9}$$

the following holds

$$\frac{1+\epsilon}{1-\epsilon}\frac{< w, x >}{||w||\,||x||} - \frac{2\epsilon}{1-\epsilon} \leq \frac{< Tw, Tx >}{||Tw||\,||Tx||} \leq 1 - \frac{\sqrt{1-\epsilon^2}}{1+\epsilon} + \frac{\epsilon}{1+\epsilon} + \frac{1-\epsilon}{1+\epsilon}\frac{< w, x >}{||w||\,||x||} \tag{4.10}$$

for the proof, refer to [52]

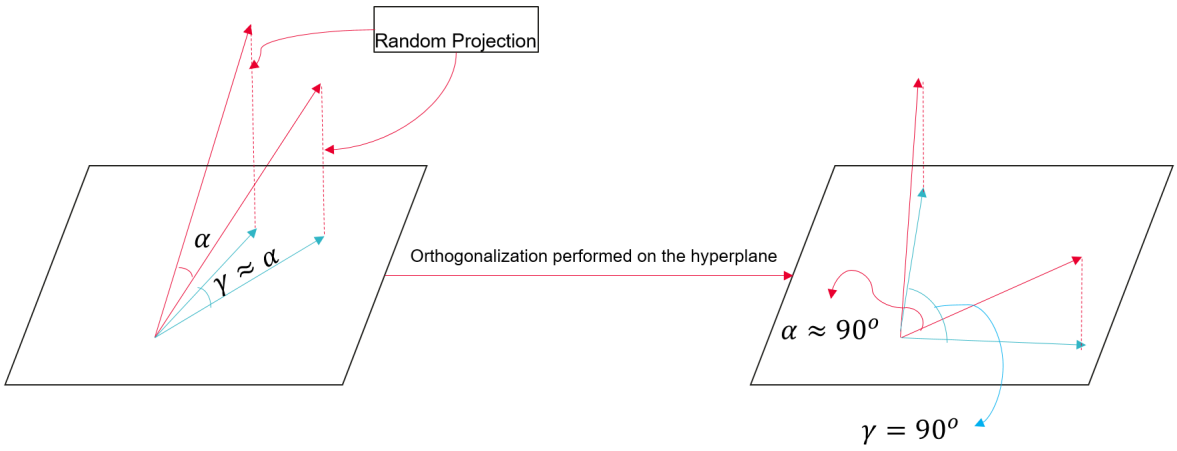

Figure 4.5: The procedure of down projecting vectors and imposing pairwise orthogonality on the projected vectors. As random projection preserves the pairwise angles, making projected version of two vectors orthogonal (Blue arrows) will make the non-projected versions of them (Red arrows) approximately orthogonal as well.

Theorem 4.4.1 indicates that Gaussian Random Projection preserves angles with negligible error. That error can be decreased by increasing the target dimension, which is M in Equations. Therefore, we can adopt Gaussian Random Projection in order to perform pairwise orthoginalization in a space with dimension significantly less than the actual dimension and draw the conclusion that in the original space vectors will be pairwise orthogonal as well. This procedure is depicted in Figure 4.5,

4.4.4 Justification Behind Using Nystrom Approximation for Dimension Efficiency

As discussed in Section 4.3.3, Nystrom approximation can be adopted in two ways. The first way is almost the same as vanilla VICReg with the difference that the pairwise orthogonalization between non-landmark vectors is preformed after projecting those vectors into the sub-space spanned by landmark vectors. The second method, first performs orthogonalization between landmark points. Then, it performs orthogonalization between non-landmark points. However, instead of performing orthogonalizing in the actual space, it will be performed in the sub-space spanned by landmark points. Therefore, it can be considered similar to random projection method with the difference that projection is not

random in this scenario and it is based on the landmark vectors.

Although Nystrom approximation is analyzed in this thesis, it should be noted that this approximation method is not suitable for identity matrices. This is due to the fact that the approximation error is large when the size of M, number of landmark vectors, is small. Additionally, based on the results, this method is not favorable in cases where full orthogonalization (Method 1) is required, as it does not significantly reduce processing time. Furthermore, in cases where partial orthogonalization is required (Method 2), this method does not provide a considerable improvement in performance.

4.5 Results

The results of some recent Contrastive/Non-Contrastive SSL methods developed for graphs is summarized in Tables 4.3 and 4.4. The row GVICReg in these tables is the implementation of VICReg loss function for graphs by the author of this thesis. As it can be seen in these tables, GVICReg performs better than all other methods for all the datasets except Coauthor-CS, for which GBT performs better. However, it provides superior performance by the cost of higher complexity compared to other baseline methods, especially the ones based on asymmetry such as BGRL [58]. The main reason behind this inefficiency is the expander module and the observation that we need significantly higher dimension than the output of encoder for the output of the expander module.

Method	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics
GVICReg	$\mathbf{81.9}_{0.3}/\mathbf{81.2}_{0.4}$	$\mathbf{79.0}_{0.6}/\mathbf{75.8}_{1.8}$	$\mathbf{93.2}_{0.4}/\mathbf{92.2}_{0.9}$	$\mathbf{88.5}_{0.6}/\mathbf{86.0}_{0.6}$	$90.8_{0.5}/87.1_{1.4}$	$\mathbf{94.7}_{0.3}/\mathbf{92.8}_{0.5}$
BGRL [58]	$74.8_{1.8}/72.2_{1.9}$	$77.2_{1.5}/74.4_{1.5}$	$91.2_{0.8}/89.3_{1.1}$	$85.7_{0.7}/83.4_{1.4}$	$90.6_{0.9}/87.3_{1.4}$	$94.4_{0.4}/92.6_{0.5}$
GBT [4]	$75.8_{0.8}/74.9_{1.1}$	$77.6_{1.0}/74.9_{1.5}$	$91.7_{1.2}/90.1_{1.7}$	$87.0_{0.5}/85.4_{0.8}$	$\mathbf{91.7}_{0.4}/\mathbf{88.2}_{1.0}$	$94.3_{0.2}/92.5_{0.1}$
DGI [63]	$80.4_{4.2}/78.1_{3.8}$	$34.9_{2.6}/12.7_{2.6}$	$88.9_{1.6}/83.5_{3.1}$	$75.6_{1.5}/61.2_{3.9}$	$90.6_{0.7}/86.7_{2.0}$	$93.8_{0.7}/91.6_{1.0}$
GRACE [82]	$75.4_{2.4}/72.3_{2.3}$	$26.9_{1.9}/10.2_{1.9}$	$88.5_{1.5}/85.1_{2.0}$	$74.4_{3.7}/65.2_{8.7}$	OOM	OOM

Table 4.3: F1 Micro/Macro for different Contrastive Non-contrastive SSL methods. Values are averaged over 5 different runs with 5 seed values. F1 is calculated for a Linear Logistic Regression node classifier trained on the embedding when the encoder is frozen to the weights at the end of the SSL training.

Method	Average(WikiCS Excluded)	Average(WikiCS, Coauthor-CS/Physics Excluded)
GVICReg	**89.8/87.9**	**87.9/86.5**
BGRL [58]	87.3/85.0	83.9/81.6
GBT [4]	88.1/86.2	84.9/83.5
DGI [63]	85.9/80.2	81.7/74.3
GRACE [82]	N/A	79.4/74.2

Table 4.4: Comparison between Non-contrastive/Contrastive SSL methods developed for Graphs averaged over different datasets

Configuration	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics
Dim = 7K Node: 1 Dim: 1	$\mathbf{81.9}_{0.3}/\mathbf{81.2}_{0.4}$	$\mathbf{79.0}_{0.6}/\mathbf{75.8}_{1.8}$	$\mathbf{93.2}_{0.4}/\mathbf{92.2}_{0.9}$	$88.0_{0.7}/85.9_{0.4}$	$89.7_{0.5}/85.1_{0.7}$	$93.1_{0.5}/90.9_{0.7}$
Dim = 4K Node: 1 Dim: 1	$77.6_{0.5}/73.8_{1.3}$	$78.5_{1.0}/75.5_{1.4}$	$92.5_{0.5}/91.3_{0.9}$	$\mathbf{88.5}_{0.6}/\mathbf{86.0}_{0.6}$	$90.0_{0.7}/86.6_{1.4}$	$93.7_{0.2}/91.7_{0.4}$
Dim = 1K Node: 1 Dim: 1	$76.6_{1.4}/75.3_{2.2}$	$78.5_{1.0}/75.3_{1.3}$	$91.5_{0.6}/90.2_{0.9}$	$87.5_{1.0}/85.4_{1.4}$	$\mathbf{90.8}_{0.5}/\mathbf{87.1}_{1.4}$	$\mathbf{94.7}_{0.3}/\mathbf{92.8}_{0.5}$
Without Expander Module	$76.1_{1.7}/74.9_{1.7}$	$76.1_{0.8}/70.8_{1.9}$	$91.1_{0.7}/89.4_{0.4}$	$86.9_{0.5}/85.7_{0.6}$	$90.6_{0.3}/86.6_{1.7}$	$94.3_{0.3}/92.4_{0.4}$

Table 4.5: The effect of Expander Module and the Output Dimension

4.5.1 The Effect of Expander Module and The Output Dimension of The Expander

To justify the usage of expander, different scenarios have been tested for GVICReg. The results are summarized in Table 4.5. In this table, we can see the downstream performance when we do not use the expander, the last row, and the effect of increasing the output dimension of the Expander. "Node: 1" and "Dim: 1" indicates that we are not performing any node or dimension sampling in these scenarios. These results showing that using expander is necessary and the output of expander must have relatively large dimension. It must be noted that the optimum output dimension is different for different datasets.

4.5.2 Uniform Sampling Strategy for Nodes and Dimensions

As shown in the previous tables, we need to have an expander module with relatively large output dimension. This module is the main source of the complexity in GVICReg training. The goal behind methods proposed in Section 4.3 and justified in Section 4.4 is improving the efficiency and the performance of this expander module. Before seeing the performance of proposed methods, we need to analyze the results of uniform node, dimension, or both

Configuration	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics
Node: 1 Dim: 0.57	$79.9_{0.6}/77.5_{1.3}$	$\underline{\mathbf{79.4}}_{0.8}/\underline{\mathbf{76.3}}_{1.0}$	$92.9_{1.5}/91.9_{2.1}$	$88.3_{0.5}/85.9_{0.6}$	$91.1_{0.5}/87.2_{0.9}$	$94.5_{0.4}/92.6_{0.4}$
Node: 1 Dim: 0.14	$80.2_{2.5}/78.8_{2.1}$	$64.7_{1.7}/52.7_{0.9}$	$92.9_{1.5}/91.9_{2.1}$	$88.8_{0.6}/86.8_{0.4}$	$91.6_{0.2}/88.5_{0.8}$	$95.1_{0.2}/93.5_{0.3}$
Node: 0.2 Dim: 1	$80.8_{2.8}/80.2_{2.8}$	$79.1_{0.8}/75.7_{1.6}$	$92.9_{0.8}/91.6_{1.3}$	$88.8_{0.5}/87.1_{0.2}$	$91.6_{0.6}/88.8_{0.7}$	$94.6_{0.2}/92.9_{0.3}$
Node: 0.5 Dim: 1	$79.9_{4.0}/79.2_{3.3}$	$79.0_{0.7}/75.6_{1.6}$	$92.7_{0.9}/91.5_{1.5}$	$88.7_{0.8}/86.7_{1.0}$	$91.0_{0.2}/87.8_{0.4}$	$94.5_{0.3}/92.7_{0.3}$
Node: 0.2 Dim: 0.57	$80.8_{3.0}/79.9_{2.1}$	$74.2_{1.0}/68.3_{1.1}$	$92.8_{1.2}/91.5_{1.8}$	$88.7_{0.8}/86.8_{0.9}$	$91.7_{0.7}/89.0_{0.8}$	$94.7_{0.3}/93.1_{0.3}$
Node: 0.2 Dim: 0.14	$\underline{81.7}_{1.0}/\underline{80.9}_{0.8}$	$73.1_{1.9}/65.9_{3.5}$	$\underline{93.1}_{1.2}/\underline{91.9}_{1.8}$	$\underline{\mathbf{88.9}}_{0.5}/\underline{\mathbf{87.5}}_{0.4}$	$\underline{\mathbf{91.9}}_{0.4}/\underline{\mathbf{89.3}}_{0.8}$	$\underline{\mathbf{95.3}}_{0.0}/\underline{\mathbf{93.9}}_{0.1}$
Node: 0.5 Dim: 0.57	$80.4_{4.0}/79.4_{2.6}$	$79.2_{0.6}/75.6_{1.6}$	$92.8_{1.0}/91.6_{1.7}$	$88.4_{0.6}/86.4_{0.7}$	$91.4_{0.3}/88.1_{0.5}$	$94.4_{0.2}/92.5_{0.2}$
Node: 0.5 Dim: 0.14	$80.7_{0.7}/80.1_{1.5}$	$70.0_{0.2}/60.2_{3.9}$	$92.8_{1.1}/91.7_{1.5}$	$88.9_{0.6}/87.2_{0.6}$	$91.7_{0.2}/88.8_{0.8}$	$95.1_{0.1}/93.6_{0.2}$
Dim = 7K Node: 1 Dim: 1	$\mathbf{81.9}_{0.3}/\mathbf{81.2}_{0.4}$	$79.0_{0.6}/75.8_{1.8}$	$\mathbf{93.2}_{0.4}/\mathbf{92.2}_{0.9}$	$88.0_{0.7}/85.9_{0.4}$	$89.7_{0.5}/85.1_{0.7}$	$93.1_{0.5}/90.9_{0.7}$
Dim = 4K Node: 1 Dim: 1	$77.6_{0.5}/73.8_{1.3}$	$78.5_{1.0}/75.5_{1.4}$	$92.5_{0.5}/91.3_{0.9}$	$88.5_{0.6}/86.0_{0.6}$	$90.0_{0.7}/86.6_{1.4}$	$93.7_{0.2}/91.7_{0.4}$
Dim = 1K Node: 1 Dim: 1	$76.6_{1.4}/75.3_{2.2}$	$78.5_{1.0}/75.3_{1.3}$	$91.5_{0.6}/90.2_{0.9}$	$87.5_{1.0}/85.4_{1.4}$	$90.8_{0.5}/87.1_{1.4}$	$94.7_{0.3}/92.8_{0.5}$
Without Expander Module	$76.1_{1.7}/74.9_{1.7}$	$76.1_{0.8}/70.8_{1.9}$	$91.1_{0.7}/89.4_{0.4}$	$86.9_{0.5}/85.7_{0.6}$	$90.6_{0.3}/86.6_{1.7}$	$94.3_{0.3}/92.4_{0.4}$

Table 4.6: Results of Uniform Sampling Baselines.

sampling as baselines. The results can be seen in Table 4.6. In this table, "Node: α" and "Dim: β" indicates that during each epoch only αN fraction of nodes and βD fraction of dimensions are used to calculate the loss function of VICReg. As mentioned in Section 4.3, the sampling procedure repeats in the beginning of each epoch. Therefore, nodes and dimensions will have chance to contribute in loss calculations.

The interesting conclusion form Table 4.6 is that for some datasets, node and feature sampling will improve the results compared to using all nodes and all features from the output of the expander.

4.5.3 Node Sampling Based on Ricci Flow

As discussed in Section 4.3, one proper metric for sampling nodes is Ricci Curvature. The result of using this metric instead of uniform distribution can be found in Table 4.7. As it can be seen, using Ricci as the sampling metric outperforms uniform sampling and without sampling cases in all datasets except Cora and WikiCS. In the justification section for Ricci

Configuration	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics
Node: 1 Dim: 0.57	$79.9_{0.6}/77.5_{1.3}$	$\underline{\mathbf{79.4}}_{0.8}/\underline{\mathbf{76.3}}_{1.0}$	$92.9_{1.5}/91.9_{2.1}$	$88.3_{0.5}/85.9_{0.6}$	$91.1_{0.5}/87.2_{0.9}$	$94.5_{0.4}/92.6_{0.4}$
Node: 1 Dim: 0.14	$80.2_{2.5}/78.8_{2.1}$	$64.7_{1.7}/52.7_{0.9}$	$92.9_{1.5}/91.9_{2.1}$	$88.8_{0.6}/86.8_{0.4}$	$91.6_{0.2}/88.5_{0.8}$	$95.1_{0.2}/93.5_{0.3}$
Node: 0.2 Dim: 1	$80.8_{2.8}/80.2_{2.8}$	$79.1_{0.8}/75.7_{1.6}$	$92.9_{0.8}/91.6_{1.3}$	$88.8_{0.5}/87.1_{0.2}$	$91.6_{0.6}/88.8_{0.7}$	$94.6_{0.2}/92.9_{0.3}$
Node: 0.5 Dim: 1	$79.9_{4.0}/79.2_{3.3}$	$79.0_{0.7}/75.6_{1.6}$	$92.7_{0.9}/91.5_{1.5}$	$88.7_{0.8}/86.7_{1.0}$	$91.0_{0.2}/87.8_{0.4}$	$94.5_{0.3}/92.7_{0.3}$
Node: 0.2 Dim: 0.57	$80.8_{3.0}/79.9_{2.1}$	$74.2_{1.0}/68.3_{1.1}$	$92.8_{1.2}/91.5_{1.8}$	$88.7_{0.8}/86.8_{0.9}$	$91.7_{0.7}/89.0_{0.8}$	$94.7_{0.3}/93.1_{0.3}$
Node: 0.2 Dim: 0.14	$\underline{81.7}_{1.0}/\underline{80.9}_{0.8}$	$73.1_{1.9}/65.9_{3.5}$	$93.1_{1.2}/91.9_{1.8}$	$88.9_{0.5}/87.5_{0.4}$	$91.9_{0.4}/89.3_{0.8}$	$95.3_{0.0}/93.9_{0.1}$
Node: 0.5 Dim: 0.57	$80.4_{4.0}/79.4_{2.6}$	$79.2_{0.6}/75.6_{1.6}$	$92.8_{1.0}/91.6_{1.7}$	$88.4_{0.6}/86.4_{0.7}$	$91.4_{0.3}/88.1_{0.5}$	$94.4_{0.2}/92.5_{0.2}$
Node: 0.5 Dim: 0.14	$80.7_{0.7}/80.1_{1.5}$	$70.0_{0.2}/60.2_{3.9}$	$92.8_{1.1}/91.7_{1.5}$	$88.9_{0.6}/87.2_{0.6}$	$91.7_{0.2}/88.8_{0.8}$	$95.1_{0.1}/93.6_{0.2}$
Dim = 7K Node: 1 Dim: 1	$\mathbf{81.9}_{0.3}/\mathbf{81.2}_{0.4}$	$79.0_{0.6}/75.8_{1.8}$	$93.2_{0.4}/92.2_{0.9}$	$88.0_{0.7}/85.9_{0.4}$	$89.7_{0.5}/85.1_{0.7}$	$93.1_{0.5}/90.9_{0.7}$
Dim = 4K Node: 1 Dim: 1	$77.6_{0.5}/73.8_{1.3}$	$78.5_{1.0}/75.5_{1.4}$	$92.5_{0.5}/91.3_{0.9}$	$88.5_{0.6}/86.0_{0.6}$	$90.0_{0.7}/86.6_{1.4}$	$93.7_{0.2}/91.7_{0.4}$
Dim = 1K Node: 1 Dim: 1	$76.6_{1.4}/75.3_{2.2}$	$78.5_{1.0}/75.3_{1.3}$	$91.5_{0.6}/90.2_{0.9}$	$87.5_{1.0}/85.4_{1.4}$	$90.8_{0.5}/87.1_{1.4}$	$94.7_{0.3}/92.8_{0.5}$
Without Expander Module	$76.1_{1.7}/74.9_{1.7}$	$76.1_{0.8}/70.8_{1.9}$	$91.1_{0.7}/89.4_{0.4}$	$86.9_{0.5}/85.7_{0.6}$	$90.6_{0.3}/86.6_{1.7}$	$94.3_{0.3}/92.4_{0.4}$
Sampling Metric: Forman Ricci	$79.4_{0.2}/78.0_{0.5}$	$78.9_{0.4}/76.1_{1.4}$	$\underline{\mathbf{93.5}}_{0.3}/\underline{\mathbf{92.5}}_{0.6}$	$\underline{\mathbf{89.3}}_{0.3}/\underline{\mathbf{87.5}}_{0.4}$	$\underline{\mathbf{92.1}}_{0.6}/\underline{\mathbf{89.7}}_{0.7}$	$\underline{\mathbf{95.6}}_{0.2}/\underline{\mathbf{94.3}}_{0.2}$

Table 4.7: Results of Sampling Nodes Based on Forman Ricci Flow as the Sampling Metric Compared to With and Without Uniform Sampling Baselines.

flow 4.4.1, it has been shown that for these datasets the Ricci flow is more uniform and less correlated with the change in the embedding after augmentation. It must be noted that these results are based on using Ricci as the sampling metric in the best uniform sampling scenario. It means that, for all datasets except WikiCS, we perform dimension sampling as well. The only difference is that, the metric for sampling nodes is the Ricci flow instead of uniform distribution.

4.5.4 Methods for Reducing the Covariance Loss Term's Complexity

In section 4.3.3, some methods were introduced which can potentially reduce the cost of covariance whitening procedure. The empirical results of these methods have been brought

in the Table 4.8. The explanation of new rows in this table is as follows:

- Nystrom (Partial) Dim: α: The second method (Whitening A and $B^T A^{-}1B$) from the Nystrom approximation section has been used and $M = \alpha \times D$.
- Nystrom (Full) Dim: α: The First method (Whitening A, $B^T A^{-}1B$, and minimizing the norm of B) from the Nystrom approximation section has been used and $M = \alpha \times D$.
- LSH (LE, α) Bits: K: LSH bucketing is used in order to find which vectors must become pairwise orthogonal. In order to further reducing the complexity, after sorting buckets based on their size, $1 - \alpha$ percent of largest buckets are chosen for orthogonalization. Number of buckets is 2^K. In other words, K random hyperplane been considered during the procedure.
- LSH (GE, α) Bits: K: LSH bucketing is used in order to find which vectors must become pairwise orthogonal. In order to further reducing the complexity, after sorting buckets based on their size, α percent of smallest buckets are chosen for orthogonalization. Number of buckets is 2^K. In other words, K random hyperplane been considered during the procedure.
- RP (Outside): The random projection is done before calculating all three loss terms. Therefore, even in invariance term, we minimize the randomly combine version of the node embeddings instead of each node embedding.
- RP (Inside) Node: α: Random projection is only done for covariance matrix calculation. D vectors which are N-dimensional will be down-projected into a αN-dimensional space.

As it can be seen in Table 4.8, there is no method working well for all datasets. However, we can conclude that on average the best performing methods are as follows:

1. Node: 0.2, Dim: 0.14
2. Nystrom (Full), Dim: 0.1
3. LSH (LE, 0.25), Bits: 5
4. RP (Inside), Node: 0.01
5. Best Uniform Sampling Case + Ricci Flow

Configuration	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics
Node: 1 Dim: 0.57	$79.9_{0.6}/77.5_{1.3}$	$79.4_{0.8}/76.3_{1.0}$	$92.9_{1.5}/91.9_{2.1}$	$88.3_{0.5}/85.9_{0.6}$	$91.1_{0.5}/87.2_{0.9}$	$94.5_{0.4}/92.6_{0.4}$
Node: 1 Dim: 0.14	$80.2_{2.5}/78.8_{2.1}$	$64.7_{1.7}/52.7_{0.9}$	$92.9_{1.5}/91.9_{2.1}$	$88.8_{0.6}/86.8_{0.4}$	$91.6_{0.2}/88.5_{0.8}$	$95.1_{0.2}/93.5_{0.3}$
Node: 0.2 Dim: 1	$80.8_{2.8}/80.2_{2.8}$	$79.1_{0.8}/75.7_{1.6}$	$92.9_{0.8}/91.6_{1.3}$	$88.8_{0.5}/87.1_{0.2}$	$91.6_{0.6}/88.8_{0.7}$	$94.6_{0.2}/92.9_{0.3}$
Node: 0.5 Dim: 1	$79.9_{4.0}/79.2_{3.3}$	$79.0_{0.7}/75.6_{1.6}$	$92.7_{0.9}/91.5_{1.5}$	$88.7_{0.8}/86.7_{1.0}$	$91.0_{0.2}/87.8_{0.4}$	$94.5_{0.3}/92.7_{0.3}$
Node: 0.2 Dim: 0.57	$80.8_{3.0}/79.9_{2.1}$	$74.2_{1.0}/68.3_{1.1}$	$92.8_{1.2}/91.5_{1.8}$	$88.7_{0.8}/86.8_{0.9}$	$91.7_{0.7}/89.0_{0.8}$	$94.7_{0.3}/93.1_{0.3}$
Node: 0.2 Dim: 0.14	$81.7_{1.0}/80.9_{0.8}$	$73.1_{1.9}/65.9_{3.5}$	$\underline{93.1}_{1.2}/\underline{91.9}_{1.8}$	$\underline{\mathbf{88.9}}_{0.5}/\underline{\mathbf{87.5}}_{0.4}$	$91.9_{0.4}/89.3_{0.8}$	$95.3_{0.0}/93.9_{0.1}$
Node: 0.5 Dim: 0.57	$80.4_{4.0}/79.4_{2.6}$	$79.2_{0.6}/75.6_{1.6}$	$92.8_{1.0}/91.6_{1.7}$	$88.4_{0.6}/86.4_{0.7}$	$91.4_{0.3}/88.1_{0.5}$	$94.4_{0.2}/92.5_{0.2}$
Node: 0.5 Dim: 0.14	$80.7_{0.7}/80.1_{1.5}$	$70.0_{0.2}/60.2_{3.9}$	$92.8_{1.1}/91.7_{1.5}$	$88.9_{0.6}/87.2_{0.6}$	$91.7_{0.2}/88.8_{0.8}$	$95.1_{0.1}/93.6_{0.2}$
Dim = 7K Node: 1 Dim: 1	$81.9_{0.3}/81.2_{0.4}$	$79.0_{0.6}/75.8_{1.8}$	$\mathbf{93.2}_{0.4}/\mathbf{92.2}_{0.9}$	$88.0_{0.7}/85.9_{0.4}$	$89.7_{0.5}/85.1_{0.7}$	$93.1_{0.5}/90.9_{0.7}$
Dim = 4K Node: 1 Dim: 1	$77.6_{0.5}/73.8_{1.3}$	$78.5_{1.0}/75.5_{1.4}$	$92.5_{0.5}/91.3_{0.9}$	$88.5_{0.6}/86.0_{0.6}$	$90.0_{0.7}/86.6_{1.4}$	$93.7_{0.2}/91.7_{0.4}$
Dim = 1K Node: 1 Dim: 1	$76.6_{1.4}/75.3_{2.2}$	$78.5_{1.0}/75.3_{1.3}$	$91.5_{0.6}/90.2_{0.9}$	$87.5_{1.0}/85.4_{1.4}$	$90.8_{0.5}/87.1_{1.4}$	$94.7_{0.3}/92.8_{0.5}$
Without Expander Module	$76.1_{1.7}/74.9_{1.7}$	$76.1_{0.8}/70.8_{1.9}$	$91.1_{0.7}/89.4_{0.4}$	$86.9_{0.5}/85.7_{0.6}$	$90.6_{0.3}/86.6_{1.7}$	$94.3_{0.3}/92.4_{0.4}$
Nystrom (Partial) Dim: 0.01	$81.2_{2.4}/80.6_{1.4}$	$78.9_{1.1}/75.3_{1.7}$	$92.4_{1.0}/91.3_{1.7}$	$87.6_{0.8}/86.1_{0.6}$	$92.2_{0.5}/89.9_{0.5}$	$95.5_{0.2}/94.1_{0.2}$
Nystrom (Partial) Dim: 0.1	$81.2_{2.4}/80.6_{1.4}$	$79.1_{0.8}/75.6_{1.8}$	$92.4_{1.2}/91.1_{1.7}$	$88.3_{0.8}/86.3_{1.0}$	$92.1_{1.4}/89.5_{1.6}$	$94.4_{0.3}/92.7_{0.2}$
Nystrom (Full) Dim: 0.01	$81.4_{2.5}/80.7_{1.2}$	$77.1_{1.2}/72.8_{2.0}$	$92.6_{1.1}/91.0_{1.3}$	$87.7_{0.9}/86.2_{0.5}$	$92.3_{0.6}/90.0_{0.5}$	$95.6_{0.3}/94.2_{0.2}$
Nystrom (Full) Dim: 0.1	$80.4_{3.2}/79.6_{3.2}$	$78.9_{1.0}/75.4_{1.5}$	$92.4_{1.2}/91.2_{1.6}$	$87.1_{1.0}/85.9_{1.4}$	$\underline{\mathbf{92.7}}_{0.3}/\underline{\mathbf{90.2}}_{0.6}$	$94.5_{0.3}/92.7_{0.3}$
LSH (LE, 0.25) Bits: 5	$83.9_{1.8}/82.8_{2.3}$	$\underline{\mathbf{79.6}}_{0.7}/\underline{\mathbf{76.8}}_{0.6}$	$93.0_{1.4}/91.9_{1.9}$	$88.5_{0.6}/86.7_{0.5}$	$91.8_{0.3}/89.0_{0.7}$	$95.3_{0.1}/93.9_{0.1}$
LSH (GE, 0.25) Bits: 5	$82.6_{2.0}/81.7_{2.2}$	$76.8_{2.0}/72.4_{3.3}$	$92.9_{1.2}/91.7_{1.7}$	$88.6_{0.6}/86.9_{0.8}$	$92.1_{0.7}/89.4_{0.9}$	$95.6_{0.3}/94.2_{0.3}$
LSH (LE, 0.50) Bits: 5	$83.6_{0.9}/82.6_{1.7}$	$79.3_{1.1}/76.1_{2.3}$	$93.0_{1.4}/91.7_{1.8}$	$88.5_{0.4}/86.7_{0.7}$	$91.6_{0.5}/88.7_{1.0}$	$95.2_{0.1}/93.8_{0.1}$
LSH (GE, 0.50) Bits: 5	$83.2_{0.9}/82.1_{1.2}$	$74.5_{2.7}/68.4_{3.4}$	$93.0_{1.1}/91.8_{1.7}$	$88.9_{0.4}/87.4_{0.4}$	$91.9_{0.6}/89.1_{0.9}$	$95.6_{0.1}/94.3_{0.1}$
LSH (LE, 0.25) Bits: 8	$84.4_{0.9}/83.1_{1.6}$	$79.2_{0.8}/76.2_{1.9}$	$92.9_{1.6}/91.6_{2.0}$	$88.8_{0.2}/86.8_{0.4}$	$92.0_{0.5}/89.6_{0.9}$	$95.5_{0.2}/94.1_{0.1}$
LSH (GE, 0.25) Bits: 8	$80.9_{3.3}/79.6_{2.8}$	$79.2_{1.0}/75.6_{1.8}$	$92.2_{1.3}/91.1_{1.8}$	$87.6_{0.8}/86.2_{0.8}$	$91.9_{0.6}/89.1_{0.8}$	$95.4_{0.2}/93.9_{0.2}$
LSH (LE, 0.50) Bits: 8	$84.3_{1.4}/83.2_{1.6}$	$79.3_{0.9}/76.1_{2.0}$	$92.8_{1.1}/91.5_{1.6}$	$88.4_{0.5}/86.6_{0.7}$	$91.9_{0.5}/89.4_{1.0}$	$95.6_{0.2}/94.2_{0.2}$
LSH (GE, 0.50) Bits: 8	$84.3_{1.9}/83.0_{2.3}$	$79.5_{0.6}/76.4_{0.9}$	$92.5_{1.1}/91.0_{1.6}$	$87.7_{0.7}/86.1_{0.8}$	$91.9_{0.8}/89.5_{1.0}$	$95.5_{0.1}/94.1_{0.2}$
RP (Outside)	$78.2_{2.6}/76.8_{1.8}$	$77.7_{1.1}/73.9_{1.9}$	$92.2_{1.4}/90.8_{2.2}$	$86.9_{1.2}/84.6_{1.1}$	$90.1_{0.4}/85.8_{0.6}$	$94.2_{0.3}/92.3_{0.3}$
RP (Inside) Node: 0.01	$\underline{\mathbf{85.1}}_{0.9}/\underline{\mathbf{84.0}}_{1.5}$	$77.8_{1.5}/73.6_{2.1}$	$92.8_{1.5}/91.7_{2.1}$	$\underline{\mathbf{88.9}}_{1.1}/\underline{\mathbf{87.5}}_{1.0}$	$91.9_{0.5}/89.3_{0.8}$	$\underline{\mathbf{95.6}}_{0.2}/\underline{\mathbf{94.3}}_{0.1}$
RP (Inside) Node: 0.1	$84.1_{1.4}/82.9_{2.7}$	$77.5_{1.2}/73.5_{2.3}$	$92.8_{1.6}/91.6_{2.3}$	$88.2_{1.1}/86.7_{0.9}$	$91.9_{0.7}/89.4_{1.1}$	$95.5_{0.2}/94.2_{0.1}$

Table 4.8: Covariance Term's Complexity Reduction Methods in Comparison with Uniform Sampling and Without Sampling Baselines.

Configuration	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics	Average
Dim = 7K Node: 1 Dim: 1	$81.9_{0.3}/81.2_{0.4}$	$79.0_{0.6}/75.8_{1.8}$	$93.2_{0.4}/92.2_{0.9}$	$88.0_{0.7}/85.9_{0.4}$	$89.7_{0.5}/85.1_{0.7}$	$93.1_{0.5}/90.9_{0.7}$	87.5/85.2
Dim = 4K Node: 1 Dim: 1	$77.6_{0.5}/73.8_{1.3}$	$78.5_{1.0}/75.5_{1.4}$	$92.5_{0.5}/91.3_{0.9}$	$88.5_{0.6}/86.0_{0.6}$	$90.0_{0.7}/86.6_{1.4}$	$93.7_{0.2}/91.7_{0.4}$	86.8/84.1
Dim = 1K Node: 1 Dim: 1	$76.6_{1.4}/75.3_{2.2}$	$78.5_{1.0}/75.3_{1.3}$	$91.5_{0.6}/90.2_{0.9}$	$87.5_{1.0}/85.4_{1.4}$	$90.8_{0.5}/87.1_{1.4}$	$94.7_{0.3}/92.8_{0.5}$	86.6/84.3
Without Expander Module	$76.1_{1.7}/74.9_{1.7}$	$76.1_{0.8}/70.8_{1.9}$	$91.1_{0.7}/89.4_{0.4}$	$86.9_{0.5}/85.7_{0.6}$	$90.6_{0.3}/86.6_{1.7}$	$94.3_{0.3}/92.4_{0.4}$	85.8/83.3
Node: 0.2 Dim: 0.14	$81.7_{1.0}/80.9_{0.8}$	$73.1_{1.9}/65.9_{3.5}$	$93.1_{1.2}/91.9_{1.8}$	$88.9_{0.5}/87.5_{0.4}$	$91.9_{0.4}/89.3_{0.8}$	$95.3_{0.0}/93.9_{0.1}$	87.3/84.9
Nystrom (Full) Dim: 0.1	$80.4_{3.2}/79.6_{3.2}$	$78.9_{1.0}/75.4_{1.5}$	$92.4_{1.2}/91.2_{1.6}$	$87.1_{1.0}/85.9_{1.4}$	$\underline{\mathbf{92.7}}_{0.3}/\underline{\mathbf{90.2}}_{0.6}$	$94.5_{0.3}/92.7_{0.3}$	87.7/85.8
LSH (LE, 0.25) Bits: 5	$83.9_{1.8}/82.8_{2.3}$	$\underline{\mathbf{79.6}}_{0.7}/\underline{\mathbf{76.8}}_{0.6}$	$93.0_{1.4}/91.9_{1.9}$	$88.5_{0.6}/86.7_{0.5}$	$91.8_{0.3}/89.0_{0.7}$	$95.3_{0.1}/93.9_{0.1}$	**88.7**/86.7
RP (Inside) Node: 0.01	$\underline{\mathbf{85.1}}_{0.9}/\underline{\mathbf{84.0}}_{1.5}$	$77.8_{1.5}/73.6_{2.1}$	$92.8_{1.5}/91.7_{2.1}$	$88.9_{1.1}/87.5_{1.0}$	$91.9_{0.5}/89.3_{0.8}$	$\underline{\mathbf{95.6}}_{0.2}/\underline{\mathbf{94.3}}_{0.1}$	**88.7/86.8**
Sampling Metric: Forman Ricci	$79.4_{0.2}/78.0_{0.5}$	$78.9_{0.4}/76.1_{1.4}$	$\underline{\mathbf{93.5}}_{0.3}/\underline{\mathbf{92.5}}_{0.6}$	$\underline{\mathbf{89.3}}_{0.3}/\underline{\mathbf{87.5}}_{0.4}$	$92.1_{0.6}/89.7_{0.7}$	$\underline{\mathbf{95.6}}_{0.2}/\underline{\mathbf{94.3}}_{0.2}$	88.1/86.3

Table 4.9: Best Performing Methods.

In the next section, these listed methods will be compared based on training time and average performance.

4.5.5 The Best Performing Methods and Efficiency and Training Time Comparison

The downstream performance of the best methods from the previous tables are listed in Table 4.9. Also, the training times compared to the best methods without sampling and complexity reduction can be seen in Table 4.10. Based on these two tables, we can draw this conclusion that the best method for reducing computational cost without degradation in performance is Random Projection only for covariance calculation (instead of random node combining). Also, uniform dimension sampling and node sampling based on Ricci flow provides the best save in calculation, but with less gain in downstream performance.

4.6 Chapter's Conclusion

This chapter introduced several methods aimed at enhancing the efficiency of VICReg when processing graph datasets. While the primary objective of these methods was to improve efficiency, they also delivered performance gains in certain scenarios, defying the

Configuration	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics	Average
Dim = 7K Node: 1 Dim: 1	1.00x	1.00x	1.00x	2.82x	15.75x	15.02x	-
Dim = 4K Node: 1 Dim: 1	0.34x	0.35x	0.34x	1.00x	6.57x	6.44x	-
Dim = 1K Node: 1 Dim: 1	0.12x	0.05x	0.05x	0.05x	1.00x	1.00x	-
Without Expander Module	0.03x	0.03x	0.05x	0.10x	0.56x	0.66x	-
Node: 0.2 Dim: 0.14	0.15x	0.21x	0.19x	0.27x	0.72x	0.73x	0.38x
Nystrom (Full) Dim: 0.1	1.14x	0.92x	0.77x	0.64x	0.85x	0.89x	0.88x
LSH (LE, 0.25) Bits: 5	0.89x	1.15x	1.41x	2.13x	3.54x	3.12x	2.04x
RP (Inside) Node: 0.01	0.58x	0.38x	0.42x	0.41x	0.71x	0.82x	0.55x
Sampling Metric: Forman Ricci	0.15x	0.38x	0.19x	0.27x	0.64x	0.83x	0.41x

Table 4.10: Training Time of the Best Performing Methods.

belief that the complexity of VICReg was necessary for optimal downstream performance. Based on the results and discussions presented in this chapter, we were able to improve the performance of VICReg on graph datasets while reducing processing time by a factor of 2. This was accomplished through the use of random projection to reduce the computational cost of calculating the covariance term, alongside uniform sampling of the nodes within the graph. Furthermore, we investigated the use of Ricci flow for node sampling in combination with uniform dimension sampling, which resulted in comparable time savings but with less performance improvement.

Chapter 5

Rank Degradation Prevention in Asymmetric Non-Contrastive Self-supervised Learning by Weight Regularization

5.1 Chapter's Overview

Non-contrastive Self-supervised Learning methods based on covariance/cross-covariance whitening rely on preventing collapse in the embedding space by maximizing the information content and the intrinsic rank of the learnt representation. However, these methods, usually have high computational complexity compared to the ones relying on asymmetry for preventing collapse in the representation (refer to Chapter 4). In this chapter, it will be shown that by adding an extra regularization term to the loss terms of Asymmetric Non-contrastive SSL methods, we can achieve on par and for some datasets better performance compared to the best performing covariance/cross-covariance whitening based methods, while having significantly less computational complexity. Bootstrapped Graph Latents (BGRL) has been chosen as the SSL method to add the proposed regularizer into its loss terms. It will be shown empirically that the proposed regularizer improves BGRL's performance by ≈ 2 percent on average with minimal extra computational complexity, $\approx$ 30 percent slower compared to Vanilla BGRL. Also, it is 9.2 and 3.8 times faster than GVICReg and GBT respectively. Mathematical justification for the proposed regularizer will be provided as well.

5.2 Introduction

In the previous chapters, it was justified that non-contrastive SSL methods are favourable compared to other methods. Between non-contrastive techniques, the ones based on asymmetry are computationally more efficient. However, for almost all datasets, asymmetric methods down-perform methods based on covariance/cross-covariance matrix whitening. The underlying reason behind this superior performance is that those methods prevent collapse by preventing rank degradation. Recently, there has been multiple observations indicating that preventing the rank degradation in the output of embedding is beneficial in improving downstream task performance [15, 12, 16]. In this chapter, a novel regularizer will be proposed which regularizes the weights of the network instead of the covariance matrix. It leads to computationally more efficient method compared to the complex loss terms proposed by covariance whitening based SSL methods [3, 4, 73].

5.3 Methodology

The proposed regularizer is used on the top of vanilla BGRL technique [58] shown in Figure 5.1. The regularization term is based on making the weight matrices of different layers orthonormal. It can get the following two forms shown in Equations 5.1 and 5.2:

$$\mathcal{L}_{reg} = \sum_{i=1}^{L} \alpha_i \left|\left| WW^T - I_{d_{in} \times d_{in}} \right|\right|_F \tag{5.1}$$

$$\mathcal{L}_{reg} = \sum_{i=1}^{L} \alpha_i \left|\left| W^T W - I_{d_{out} \times d_{out}} \right|\right|_F \tag{5.2}$$

where L is the number of layers, α_i controls the intensity of regularization for different layers, and d_{in} and d_{out} are the input and output dimensions of the specific layer respectively. To decide between these two terms, d_{in} and d_{out} must be considered. If d_{in} is smaller than d_{out}, Equation 5.1 should be used. Otherwise, Equation 5.2 is the better choice. The proposed method is called BGRLReg in the result tables. The regularizer is justified in Section 5.4 based on rank degradation prevention and in Section 5.5 based on Lipschitz constant controlling affect of the regularizer.

The total loss of the SSL method will be as follows:

$$\mathcal{L} = \mathcal{L}_{BGRL} + \mathcal{L}_{reg} \tag{5.3}$$

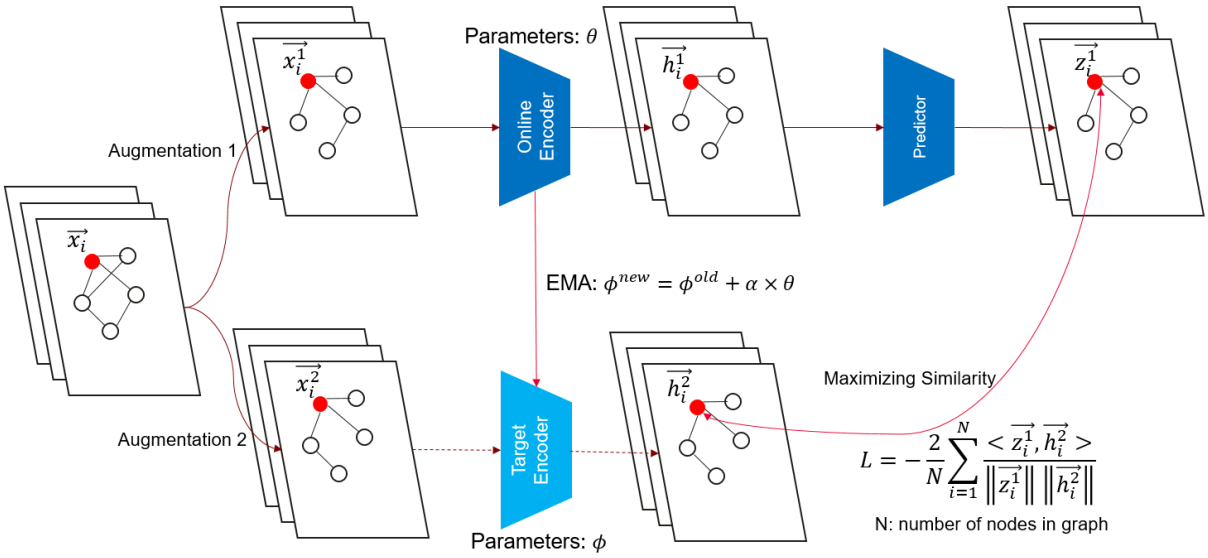

Figure 5.1: Overview of training procedure. Two views of the original graph obtained by augmentation. One of them is passed through online (student) encoder and the other one through the target (teacher) encoder. The weights of online encoder change by back-propagation, while the weights of target encoder updated by performing Exponential Moving Averaging(EMA). Finally, the embedding of the upper branch is passed through a predictor module which predicts the target embedding. The objective is maximizing the cosine similarity between $\vec{z_i^1}$ and $\vec{h_i^2}$. Dashed lines show stop-gradient.

5.4 Justification: Rank Degradation Prevention

For the sake of simplicity, suppose we have a Graph Neural Network with one Graph Convolution Layer [32]. The output of this layer can be written as follows:

$$X^{(1)} = D^{-0.5}AD^{-0.5}X^{(0)}W = HW \tag{5.4}$$

The non-contrastive methods based on making dimensions uncorrelated and whitening the covariance matrix over dimensions, such as Barlow Twins and VICReg, try to define loss terms which make the covariance matrix at the output of network close to identity. We want to show that controlling the singular values of the weight matrix by making W^TW or WW^T close to identity will decrease the Forbenious norm between covariance matrix of output and identity. To increase the information content of each dimensions, non-contrastive methods based on whitening covariance matrix tries to minimize the Frobenius

norm between covariance matrix and identity. The Frobenius norm can be written based on trace as follows:

$$||Cov - I_{(d_{out} \times d_{out})}||_F = ||W^T H^T HW - I||_F = \sqrt{tr\Big((W^T H^T HW - I)(W^T H^T HW - I)\Big)} \tag{5.5}$$

To simplify the formula more:

$$||Cov - I||_F = \sqrt{tr(Cov^2) - 2tr(Cov) + I} = \sqrt{\sum_{i=1}^{d_{out}} (\sigma_i^2 - 2\sigma_i + 1)} = \sqrt{\sum_{i=1}^{d_{out}} (\sigma_i - 1)^2} \tag{5.6}$$

To minimize the metric shown in Equation 5.6, we need to make all of the singular values of the covariance matrix as close as possible to one. Covariance matrix consists of two components which are input and weights. During training, the input to the network is not controllable and therefore, the only way for controlling the singular values of covariance is regularizing the weights of network. In the following equation, by help of singular value inequalities, we show how we can bound the singular values of covariance. The i_{th} singular value of covariance can be bounded as follows:

$$\sigma_i(W^T H^T H)\sigma_1(W) \leq \sigma_i(W^T H^T HW) \leq \sigma_i(W^T H^T H)\sigma_{d_{out}}(W) \tag{5.7}$$

By further decomposition of lower and upper bounds we get the following bounds:

$$\sigma_i(H)^2\sigma_1(W)^2 \leq \sigma_i(W^T H^T HW) \leq \sigma_i(H)^2\sigma_{d_{out}}(W)^2 \tag{5.8}$$

For a perfect whitening, we need to regularize $W^T W$ or WW^T to be as close as possible to a diagonal matrix in which the i_{th} diagonal element is $d_i = \frac{1}{\sigma_i(H)^2}$. However, calculating this regularization term requires performing Singular Value Decomposition (SVD) for each batch and each layer during all epochs which makes it computationally intractable. Also, if we perform the training in batch format, the optimization will be unstable as for each batch the singular values will be different from others. Based on all of these aforementioned facts, we used identity matrix as the target matrix for $W^T W$ or WW^T.

Our proposed regularization term shown in Equations 5.14 and 5.15, in perfect case, forces the weight matrices to be orthonormal. In other words, it forces singular values of W to be as close as possible to one. By doing that, the upper and lower bounds of Equation 5.8 becomes the same since singular values of W will be almost one. It makes all of the singular values of covariance matrix the same as eigenvalues of the input matrix. In other words, the GCN [32] layer will not cause any rank degradation on the input data. The rank of output has shown to be an important metric which guarantees a proper downstream performance [16, 15, 12].

5.5 Justification: Controlling Lipschitz Constant

Inspired by [76] which shows that the orthogonality constraint in transformers [61] can control the Lipschitz constant, this section shows that the same effect is imposed by the introduced regularizer in GNN's with GCN layers [32]. The Lipschitz constant of the composition of functions is less than or equal the multiplication of Lipschitz constants of each function. The Graph Neural Network which is the backbone encoder of our method is the composition of multiple functions which are the different layers of the network. In our implementation, each layer is Graph Convolution Layer. Therefore, to control Lipschitz constant of the whole encoder, the Lipschitz constant of each GCN [32] layer of GNN needs to be controlled. The goal is to control this constant to be less than one in order to prevent Lipschitz constant explosion. Therefore, one proper regularizer could be the one which can force the Lipschitz constant of each layer to be less than one. First, it will be shown that Lipschitz constant of each layer is larger than or equal the largest singular value of the weights of that layer. Then, it will be argued that a regularizer which makes the matrix WW^T close to identity can control the largest singular value of W. Without loss of generality, we focus on the first layer of GNN which in our case is a GCN module. The output of this layer for node v can be written as follows:

$$X^{(l)} = D^{-0.5}AD^{-0.5}X^{(l-1)}W \tag{5.9}$$

the superscript is the index of layer. For each node i, we can write this formula as follows:

$$x_i^l = \Big[\sum_{j\in N(i)\cup i} \frac{1}{\sqrt{d_i d_j}} x_j^{(l-1)}\Big] W = h_i^{(l-1)}W \tag{5.10}$$

The Lipschitz constant controls the upper bound on the amount of change in the output after changing the input of the function. Mathematically it can be defined as follows:

$$\Big|\Big|h_i^{(l-1)}W - h_i^{(l-1)'}W\Big|\Big| \leq L\Big|\Big|h_i^{(l-1)} - h_i^{(l-1)'}\Big|\Big| \tag{5.11}$$

$$\frac{\Big|\Big|(h_i^{(l-1)} - h_i^{(l-1)'})W\Big|\Big|}{\Big|\Big|h_i^{(l-1)} - h_i^{(l-1)'}\Big|\Big|} \leq L \tag{5.12}$$

therefore, Lipschitz constant (L) is greater than or equal:

$$\sup_{h_i^{(l-1)}\neq h_i^{(l-1)'}} \frac{\Big|\Big|(h_i^{(l-1)} - h_i^{(l-1)'})W\Big|\Big|}{\Big|\Big|h_i^{(l-1)} - h_i^{(l-1)'}\Big|\Big|} = \sup_{h_i^{(l-1)}\neq h_i^{(l-1)'}} \Big|\Big|\frac{h_i^{(l-1)} - h_i^{(l-1)'}}{||h_i^{(l-1)} - h_i^{(l-1)'}||}W\Big|\Big| \tag{5.13}$$

which can be simplified as $\sup_{h_i^{(l-1)} \neq h_i^{(l-1)'}} ||vW||$, where $||v|| = 1$. This is the norm definition of the largest singular value of W. We want to limit the Lipschitz constant to be no grater than one. Therefore, we need to control the largest singular value of W to be not greater than one. We can try to satisfy this condition by minimizing any of the following constraints:

$$\left|\left|WW^T - I_{d_{in} \times d_{in}}\right|\right|_F \tag{5.14}$$

$$\left|\left|W^TW - I_{d_{out} \times d_{out}}\right|\right|_F \tag{5.15}$$

where d_{in} is the input dimension of the layer and d_{out} is the output dimension. Minimizing any of these equations can control the maximum singular value of W. However, to have smaller complexity, we need to use Equation 5.14 when $d_{in} < d_{out}$ and otherwise, we need to use Equation 5.15.

5.6 Results

Method	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics	Average Rank
GVICReg	$81.9_{0.3}/81.2_{0.4}$	$79.0_{0.6}/75.8_{1.8}$	$93.2_{0.4}/92.2_{0.9}$	$88.5_{0.6}/86.0_{0.6}$	$90.8_{0.5}/87.1_{1.4}$	$94.7_{0.3}/92.8_{0.5}$	1.5
GBT ([4])	$75.8_{0.8}/74.9_{1.1}$	$77.6_{1.0}/74.9_{1.5}$	$91.7_{1.2}/90.1_{1.7}$	$87.0_{0.5}/85.4_{0.8}$	$91.7_{0.4}/88.2_{1.0}$	$94.3_{0.2}/92.5_{0.1}$	2.7
GRACE ([82])	$75.4_{2.4}/72.3_{2.3}$	$26.9_{1.9}/10.2_{1.9}$	$88.5_{1.5}/85.1_{2.0}$	$74.4_{3.7}/65.2_{8.7}$	OOM	OOM	NA
BGRL (Vanilla, [58])	$74.8_{1.8}/72.2_{1.9}$	$77.2_{1.5}/74.4_{1.5}$	$91.2_{0.8}/89.3_{1.1}$	$85.7_{0.7}/83.4_{1.4}$	$90.6_{0.9}/87.3_{1.4}$	$94.4_{0.4}/92.6_{0.5}$	3.8
BGRLReg (BGRL With Regularizer)	$80.4_{3.7}/78.5_{3.9}$	$77.4_{1.4}/74.5_{1.5}$	$92.4_{1.1}/90.9_{1.5}$	$86.6_{1.5}/84.9_{2.3}$	$92.4_{0.7}/90.5_{1.2}$	$95.1_{0.2}/93.6_{0.2}$	2

Table 5.1: The format of results is F1 Micro/Macro and values have been averaged over 5 different runs with 5 seed values. F1 is calculated for a Linear Logistic Regression node classifier trained on the embedding when the encoder is frozen to the weights at the end of the SSL training.

Before talking about results, in the tables 5.1 and 5.2 BGRL (With Regularizer) is our proposed method and GVICReg is the implementation of VICReg loss on top of GBT framework (the same as Chapter 4). The downstream performance (Node Classification) of different methods can be seen in Table 5.1. For all of the datasets, the proposed regularizer

Execution Time Compared to BGRL (With Regularizer)	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics	Average
GVICReg	14.9x	15.0x	19.2x	4.9x	0.7x	0.5x	9.2x
GBT ([4])	6.7x	5.8x	7.1x	1.5x	1.4x	0.6x	3.8x
GRACE ([82])	0.7x	1.5x	1.5x	1.4x	OOM	OOM	N/A
BGRL (Vanilla, [58])	0.6x	0.8x	0.8x	0.8x	0.7x	0.8x	0.7x
BGRLReg (BGRL With Regularizer)	1.0x	1.0x	1.0x	1.0x	1.0x	1.0x	1.0x

Table 5.2: Comparison Between the Execution of Other Methods and Our Method

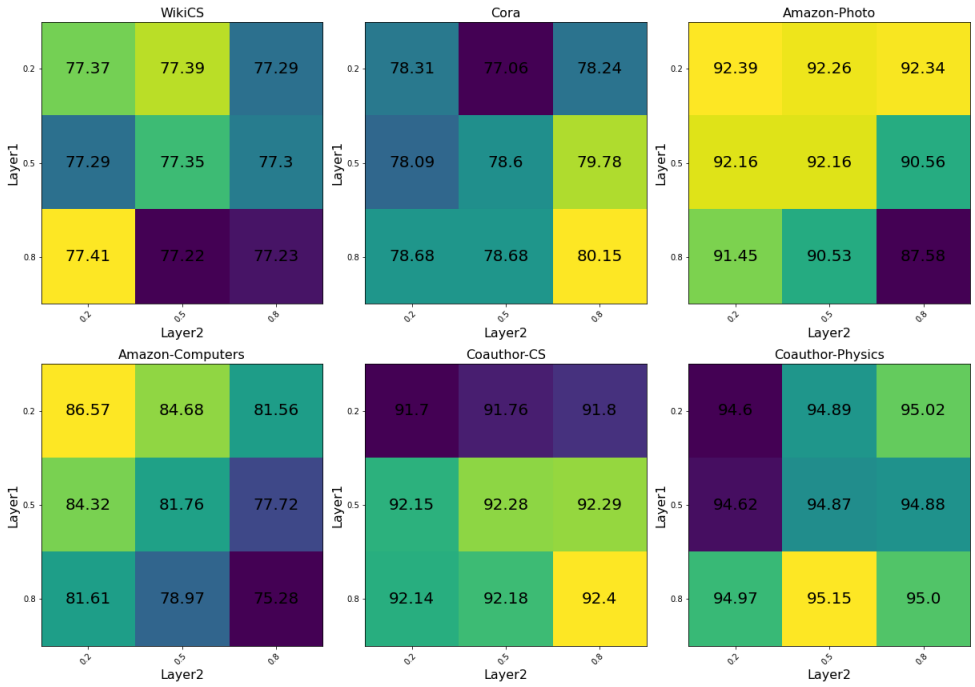

Figure 5.2: F1 Micro Based on Different Regularization Factor for Different Layers

improves the performance of Vanilla BGRL. Also, on average, the rank of BGRL with the regularizer is higher than GBT and slightly worse than GVICReg. This rank is the average rank of methods based on their performance for different datasets. It also outperforms all other baselines for Coauthor-CS/Physics datasets. In the sense of complexity, this novel method is on average 30 percent slower than Vanilla BGRL, while 9.2 times and 3.8 times faster than GBT and GVICReg respectively.

To analyze the importance of regularization for different layer, the optimum strength of regularization terms for different layers, α_i's in Equations 5.14 and 5.15, are not the same. In Figures 5.2 and 5.3 we can see F1 micro and macro performance of different regularization strength for different layers. In two out of six datasets, the strength of the first layer should be higher than the second one. Investigating the reason behind this observation could be considered as an interesting future work. A hypothesize could be that without using the regularizer, the singular values of the first layer is significantly larger than the second layer and that is the reason behind requiring stronger regularization.

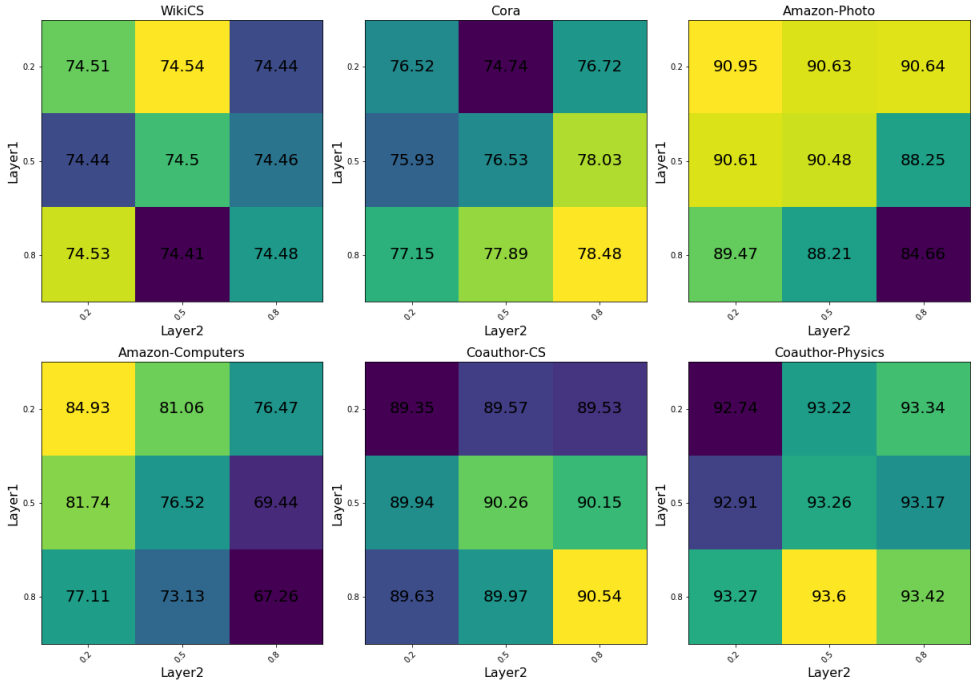

Figure 5.3: F1 Macro Based on Different Regularization Factor for Different Layers

5.7 Chapter's Conclusion

In this chapter, a weight regularizer based on making the weights of layers of GNN orthonormal was proposed and justified mathematically. This regularizer was combined with one recent and successful asymmetry-based NCSSL method called BGRL. The proposed regularizer improved vanilla BGRL by around 2 percent while being only 30 percent slower than that.

As future work, it would be beneficial to analyze the underlying reasons behind the different layer importances in the regularization process. Furthermore, while covariance-based methods already incorporate a regularizer to prevent rank degradation by regularizing the covariance matrix, it is possible that the proposed regularizer in this chapter could also improve their performance. Therefore, exploring the application of our regularizer to covariance-based methods could be a promising direction for future research.

Chapter 6

Improving Asymmetric Non-Contrastive Self-Supervised Learning Methods by Early Stopping the Update of the Target Network

6.1 Chapter's Overview

Asymmetric non-contrastive self-supervised learning (SSL) methods can be formulated as student-teacher paradigm. Recent observations made in Knowledge Distillation (KD) literature have shown that there is an optimum epoch at which the training of the teacher network should be stopped to achieve the best performance. In this chapter, a similar phenomenon will be shown to happen in asymmetric non-contrastive SSL methods as well. It means that their performance can be improved by stopping the procedure of updating the target (teacher) network at early epochs. It will be argued that this improvement can be explained by the change in the mutual information between the output of layers and the input data. This hypothesis will be verified and based on that, a novel approach for self-supervised learning will be proposed. This method will be added to Bootstrapped Graph Latents (BGRL) technique for graph representation tasks and improves its performance by ≈ 1 percent on average.

6.2 Introduction

In the previous chapters, using non-contrastive SSL over Contrastive was justified. Also, it was mentioned that the current asymmetric methods in Graph Learning Literature do not work as well as methods based on covariance whitening. However, the ones based on asymmetry are computationally more efficient. Therefore, it would be beneficial to enhance their performance to a level better or comparable to more complex methods. In this chapter, the goal is using the insight from Knowledge Distillation literature in order to improve asymmetry-based NCSSL methods which can be formulated as teacher-student paradigm as well.

Asymmetric non-contrastive SSL methods can be looked from the angle of knowledge distillation. Similar to KD, there is a student (online) encoder which tries to mimic the teacher (target) encoder. Inspired by the recent progress in Knowledge Distillation showing that a fully-trained teacher is not necessary the best teacher [49, 66, 11, 8], it will be investigated if the same phenomenon happens in asymmetric SSL. In other words, if switching off the EMA module and stopping the training of the target (teacher) encoder improves the performance. Bootstrapped Graph Latents (BGRL) is chosen as a the method representing the class of asymmetry-based NCSSL techniques. However, the application of the proposed method is not limited to BGRL and it is applicable to any asymmetric non-contrastive SSL framework which is based on updating one of the encoders by help of the other one.

In summary, the contributions of this chapter are as follows. First, it shows that we need to stop the updating procedure of the target (teacher) encoder at early epochs. Then, it will be shown empirically that this epoch is the same as the one at which the mutual information between the output of different layers and the input start decreasing. Finally, a methodology will be proposed by which we can detect the optimum epoch for switching EMA module off during training.

6.3 Methodology and Results

BGRL solves the problem of "Collapse" by making the dual-encoder structure asymmetric. Since BGRL can be formulated as student-teacher framework, it needs to be validated that if similar to KD, early stopping of the teacher's training improves the performance on the downstream task. Therefore, a switch is added to the Exponential Moving Average (EMA) module of BGRL. This switch controls if the target encoder will be updated or

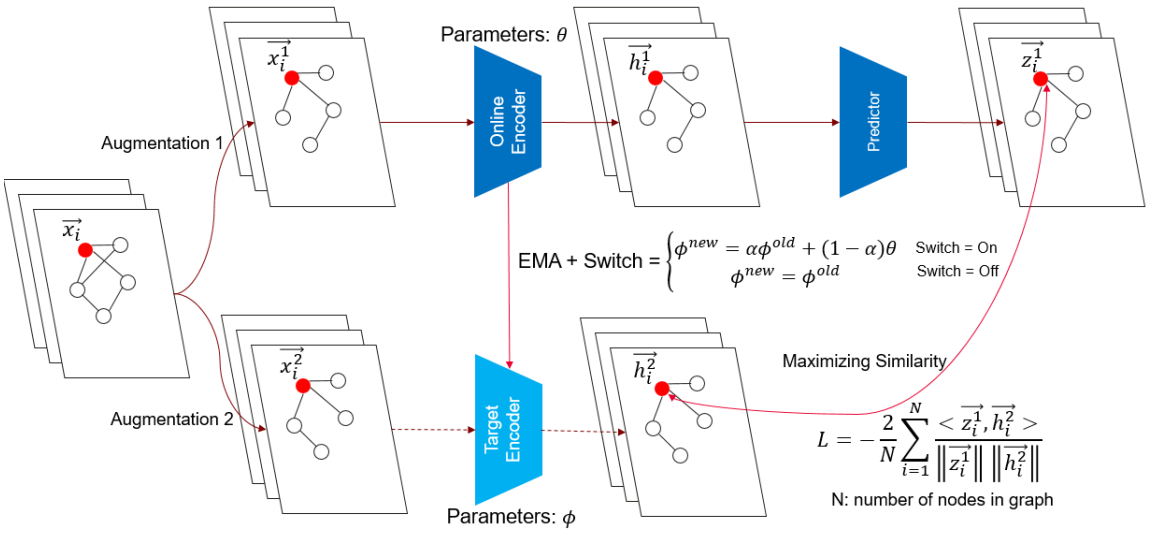

Figure 6.1: Overview of proposed method. We get two views of the original graph by augmentation. One of them is passed through online (student) encoder and the other one through the target (teacher) encoder. The weights of online encoder change by back-propagation, while the weights of target encoder updated by performing Exponential Moving Averaging(EMA). Finally, the embedding of the upper branch is passed through a predictor module which predicts the target embedding. The objective is maximizing the cosine similarity between $\vec{z_i^1}$ and $\vec{h_i^2}$. Dashed lines show stop-gradient.

not. The overall architecture can be seen in Figure 6.1. The preliminary empirical observations summarized in Table 6.1 indicates that there is an epoch at which the EMA update procedure on the target network should be stopped in order to achieve the optimum performance. The hypothesize for the underlying reason behind this observation is the decrease in mutual information between the output of layers of the online network and the input data during compression stage which starts after a specific epoch [59, 50]. In other words, after this epoch, the online network, and as a result the target network, starts forgetting some information obtained from the input which seems to be important for SSL. We need to prevent this forgetting in the target network since the target network is the one which provides the anchor representations for the online network, and these anchors must be maximally informative about the input samples. In Section 6.4, more detailed justification using the recent progress in Knowledge Distillation literature is provided.

To test this hypothesis, we need a metric for calculating the mutual information be-

tween the output of different layers and the input. Then it can be shown that the epoch at which mutual information starts a decreasing trend is the optimum epoch for switching off EMA. Mutual information shows the dependency between two random variables. Therefore, estimating dependency is similar to estimating mutual information. One stable and proper metric for estimating dependency is Hilbert-Schmidt Independence Criterion (HSIC) [18]. An empirical approximation for HSIC between two random variables x and y can be calculated as follows [18]:

$$HSIC(x, y) = \frac{tr(KHLH)}{(m-1)^2} \tag{6.1}$$

where $H, K, L \in R^{m \times m}$. $H := I_{m \times m} - ee^T/m$ and $e \in R^m$ is a column of all ones. H is a centring matrix which makes the mean of rows and columns of K and L equal to zero. K and L are the kernels on x and y defined as follows: $K_{i,j} = k(x_i, x_j)$, $L_{i,j} = l(y_i, y_j)$. As we want to aggregate the mutual information over all layers of the network, the HSIC metric is the summation of HSIC between output of each layer and the input. Suppose the output of layer l is named as $y^{(l)}$ and the input as x. Total HSIC will be:

$$HSIC_{Total} = \sum_{i=1}^{N} HSIC(x, y^{(i)}) \tag{6.2}$$

Where N is the total number of layers.

Epoch	WikiCS	Cora	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics
20	$77.2_{1.4}/74.4_{1.5}$	$71.4_{1.7}/69.2_{1.3}$	$88.2_{1.0}/84.9_{1.5}$	$79.5_{1.4}/75.2_{1.2}$	$\mathbf{91.6}_{1.0}/\mathbf{88.8}_{1.0}$	$94.6_{0.3}/92.7_{0.3}$
40	$77.2_{1.5}/74.4_{1.5}$	$\mathbf{76.9}_{1.6}/\mathbf{74.9}_{2.0}$	$88.1_{1.6}/85.5_{2.2}$	$77.0_{2.5}/70.5_{4.8}$	$91.0_{0.6}/87.8_{0.5}$	$94.7_{0.2}/92.8_{0.4}$
80	$\mathbf{77.6}_{1.0}/\mathbf{74.8}_{1.1}$	$76.6_{1.0}/74.8_{1.3}$	$88.0_{0.8}/85.0_{1.3}$	$77.9_{1.7}/71.9_{3.1}$	$90.9_{0.5}/87.7_{0.5}$	$\mathbf{94.8}_{0.4}/\mathbf{93.0}_{0.6}$
160	$77.2_{1.4}/74.3_{1.6}$	$76.6_{2.0}/74.5_{2.8}$	$88.4_{1.1}/85.2_{1.4}$	$76.7_{2.2}/71.1_{3.7}$	$90.9_{0.5}/87.7_{0.5}$	$94.7_{0.2}/92.9_{0.5}$
320	$77.2_{1.5}/74.4_{1.6}$	$76.7_{1.7}/74.5_{2.1}$	$89.3_{0.9}/86.7_{1.2}$	$79.6_{2.1}/74.3_{3.7}$	$90.9_{0.8}/87.6_{1.0}$	$94.6_{0.3}/92.9_{0.3}$
640	$77.3_{1.4}/74.5_{1.5}$	$75.5_{1.7}/73.6_{1.9}$	$\mathbf{91.6}_{1.0}/\mathbf{89.8}_{1.48}$	$83.0_{1.8}/80.5_{2.7}$	$90.8_{0.9}/87.6_{1.3}$	$94.4_{0.2}/92.6_{0.2}$
No Stop	$77.2_{1.5}/74.4_{1.5}$	$74.8_{1.8}/72.2_{1.9}$	$91.2_{0.8}/89.3_{1.1}$	$\mathbf{85.7}_{0.7}/\mathbf{83.4}_{1.4}$	$90.6_{0.9}/87.3_{1.4}$	$94.4_{0.4}/92.6_{0.5}$

Table 6.1: Exhaustive search to find the optimum epoch for stopping target model update procedure. The format of results is F1 Micro/Macro and values have been averaged over 5 different runs with 5 seed values. F1 is calculated for a Linear Logistic Regression node classifier trained on the embeddings when the encoder is frozen to the weights at the end of the SSL training.

In Figures 6.2, 6.3, 6.4, and 6.5, the change of HSIC metric during SSL is plotted for different datasets. These plots show high correlation between the optimum epoch for stopping the update procedure found by exhaustive search and the epoch at which HSIC

Method	WikiCS	Cora	Coauthor-CS	Coauthor-Physics
GBT	$77.6_{1.0}$/$\mathbf{74.9}_{1.5}$	$75.8_{0.8}$/$\mathbf{74.9}_{1.1}$	$91.7_{0.4}$/$88.2_{1.0}$	$94.3_{0.2}$/$92.5_{0.1}$
GRACE [82]	$26.9_{1.9}$/$10.2_{1.9}$	$75.4_{2.4}$/$72.3_{2.3}$	OOM	OOM
BGRL (Vanilla)	$77.2_{1.5}$/$74.4_{1.5}$	$74.8_{1.8}$/$72.2_{1.9}$	$90.6_{0.9}$/$87.3_{1.4}$	$94.4_{0.4}$/$92.6_{0.5}$
BGRL (Early Stop-Exhaustive Search)	$\mathbf{77.6}_{1.0}$/$74.8_{1.1}$	$\mathbf{76.9}_{1.6}$/$\mathbf{74.9}_{2.0}$	$\mathbf{91.6}_{1.0}$/$\mathbf{88.8}_{1.0}$	$\mathbf{94.8}_{0.4}$/$\mathbf{93.0}_{0.6}$
BGRL (Early Stop-HSIC)	$\mathbf{77.6}_{1.4}$/$74.8_{1.5}$	$76.5_{1.9}$/$74.6_{2.3}$	$\mathbf{91.6}_{1.1}$/$88.7_{1.1}$	$94.7_{0.6}$/$\mathbf{93.0}_{0.6}$

Table 6.2: The format of results is F1 Micro/Macro and values have been averaged over 5 different runs with 5 seed values. F1 is calculated for a Linear Logistic Regression node classifier trained on the embeddings when the encoder is frozen to the weights at the end of the SSL training.

reaches its peak. This high correlation gives us a clue for an adaptive method of changing the state of EMA's switch based on calculated HSIC metric during SSL.

As finding the optimum epoch by exhaustive search is not practical, HSIC can be used as the metric to find the epoch at which the update of the target model must be stopped. The recipe for early stopping the target network's update is as follows:

1. Calculate HSIC during training. To make this step efficient, HSIC is calculated in 5-epoch intervals. Also, a portion of data are sampled to calculate kernel. This sampling is repeated during each epoch. Also, instead of nonlinear kernels, inner product as a linear kernel can be used to approximate HSIC. The results summarized in Table 6.2 show that these approximations will not lead to significant degradation of performance.

2. Stop the update procedure when HSIC metric reaches its peak. It must be noted that HSIC must be smoothed to remove high frequency changes in values from one epoch to another.

The results of this method for stopping the update procedure can be seen in Table 6.2. Switching based on HSIC performs almost the same as the best case in exhaustive search scenario, while it is practical in contrast with exhaustive search. Although the proposed recipe leads to better results for four benchmark datasets and it performs on par exhaustive search, it could not perform on par with exhaustive search for two datasets: Amazon-Photo and Amazon-Computers. The reason behind this observation for these two datasets left to future work. The hypothesize is that these two datasets need optimizing parameters and the type of the kernel used in HSIC calculation.

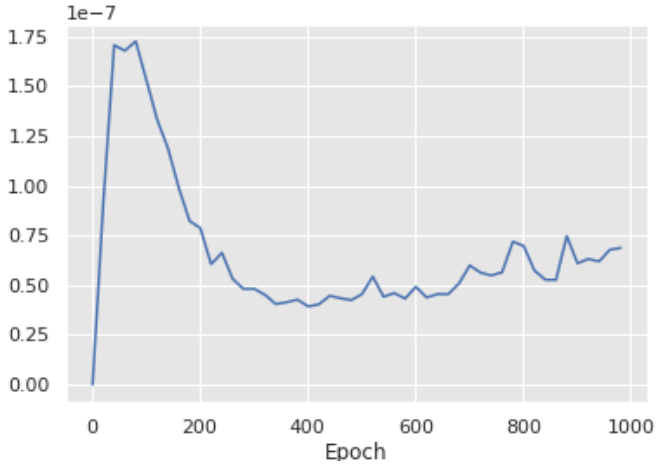

Figure 6.2: Change in HSIC during training for WikiCS Dataset. HSIC has been averaged over 5 runs with different seed values. Kernel Used in calculation is RBF Kernel. Maximum at Epoch = 60

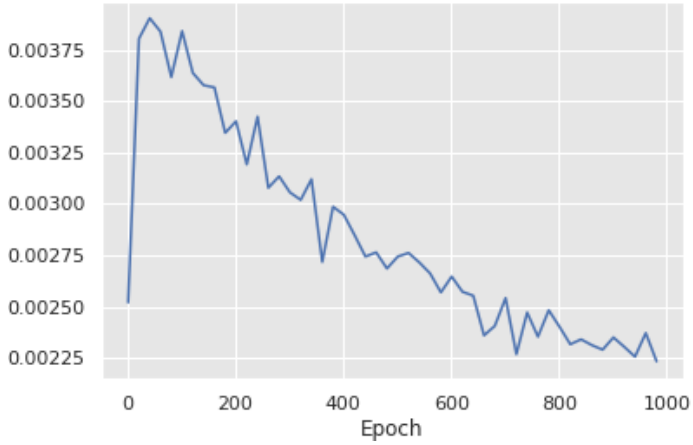

Figure 6.3: Change in HSIC during training for Cora Dataset. HSIC has been averaged over 5 runs with different seed values. Kernel Used in calculation is RBF Kernel. Maximum at Epoch = 40

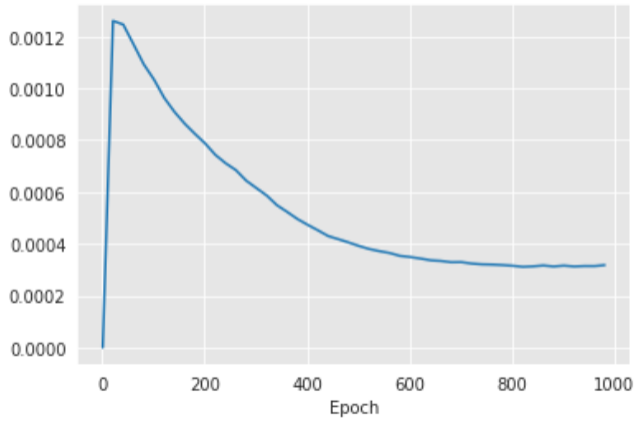

Figure 6.4: Change in HSIC during training for Coauthor-CS Dataset. HSIC has been averaged over 5 runs with different seed values. Kernel Used in calculation is RBF Kernel. Maximum at Epoch = 20

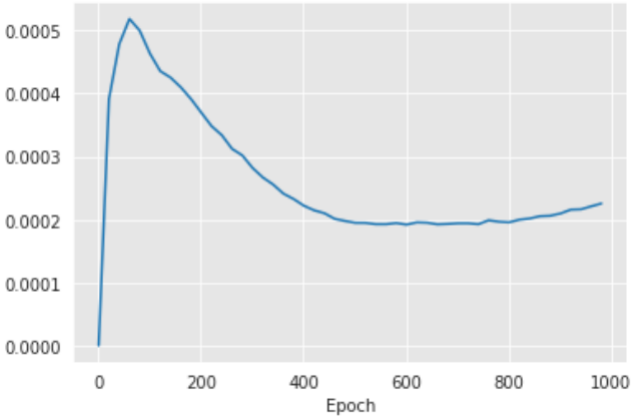

Figure 6.5: Change in HSIC during training for Coauthor-Physics Dataset. HSIC has been averaged over 5 runs with different seed values. Kernel Used in calculation is RBF Kernel. Maximum at Epoch = 80

6.4 Justification

Asymmetry-based non-contrastive dual-encoder SSL methods can be formulated as teacher-student paradigm. The goal of procedure is training a network which can mimic the target network's embedding, prediction, class assignments. In knowledge Distillation literature, there have been multiple observations indicating that early stopping the teacher's training improves the performance [49, 66, 11, 8]. The proposed method in this thesis corresponds to stopping the training of teacher at early epochs. This means that after a specific epoch, the target (teacher) network will not be updated anymore. To explain why this early stopping helps, authors of [49], by help of information bottleneck concept from papers[59, 50], argue that the reason is the fact the after some point during training, deep networks start forgetting information about input data. This stage is called compression stage [59, 50]. Compression stage is the stage during which network forgets input and tries to achieve better generalization [59, 50]. This information content, which is called Dark Knowledge in Knowledge Distillation literature [11], is crucial in KD and as a result, asymmetric Self-supervised Learning methods which can be seen as a special form of KD. Recently, Wang et. al. showed empirically that the best checkpoint for teacher is the checkpoint at which the compression stage starts [66].

Based on the empirical results of this chapter, it has been seen that the optimal epoch to stop updating the target network is approximately when the HSIC metric, which provides an estimate of mutual information and dependency, begins to decrease. This indicates that the online network is starting to forget valuable information about the input. It is essential not to allow the target (teacher) network to forget this information since it contains Dark Knowledge which is the inter-sample relationships. By stopping the updating procedure at this optimal epoch, it can be ensured that the target network retains all the relevant information needed to transfer its knowledge effectively to the student network.

6.5 Chapter's Conclusion

It was demonstrated in this chapter that there exists an optimal epoch to halt the weight updating process of the EMA target network in asymmetric NCSSL methods. A correlation was established between this phenomenon and recent findings in Knowledge Distillation which recommend early termination of teacher training. The peak epoch of HSIC metric was identified the optimal epoch for stopping the update procedure. For certain datasets, the proposed method demonstrated slightly inferior performance compared to an exhaustive search for the optimal epoch. It must be noted that this method is not limited to any

BGRL technique and can be utilized to enhance any existing or future dual-encoder SSL methods that rely on updating one encoder's weights based on the other encoder's weights.

As future work, hyperparameter optimization is required for the parameters used in smoothing the HSIC values and also the intervals of HSIC calculation. Also, the feasibility and performance of stopping the update of different parts of network at different epochs should be investigated. It is highly likely that the optimum epoch is different for different layers of GNN. Additionally, the reason behind not successful performance of the proposed methods for two Amazon datasets needs to be understood. Most likely, the reason is behind the type of kernel and paramaters of that.

Chapter 7

Conclusion

In previous chapters, various ideas were proposed in order to improve covariance-based and asymmetry-based non-contrastive SSL methods. To review, in dual-encoder SSL methods, as the optimization problem tries to map semantically similar points into points close to each other in the embedding space, mapping all points into the same point or vicinity of the same point is the trivial solution and encoder learns that. This phenomenon is called representation collapse and it degrades the quality of embedding space significantly. Covariance-based methods are the ones which prevent collapse by performing negative sampling in embedding variable space instead of sample space and those try to make the covariance/cross-covariance matrix identity. Asymmetry-based methods, try to prevent collapse by other mechanisms such as stopping gradient flow in one branch and so on. In Chapter 4, some novel ideas proposed in order to reduce the complexity of one of the well-known and recent covariance-based methods called VICReg [3]. In Chapters 5 and 6 two novel methods proposed for one of the successful asymmetry-based methods called BGRL [58]. In Table 7.1 the best results from previous chapters with supervised results and some baselines are shown.

As it can be concluded form Table 7.1, the improvements proposed in Chapter 4 can push the performance of VICReg on average to the level slightly higher than supervised training. This observation shows the surprising effectiveness of proposed methods for VICReg. Also, these methods have two times less complexity on average compared to vanilla VICReg.

This table also shows the improvements achieved by the methods proposed in Chapters 5 and 6 for BGRL. It shows that these techniques can improve BGRL's performance by ≈ 2 and ≈ 0.8 percent respectively. In sense of adding computational complexity to the vanilla

Method	Cora	WikiCS	Amazon-Photo	Amazon-Computers	Coauthor-CS	Coauthor-Physics	Average
Supervised	$\mathbf{85.7}_{0.3}/\mathbf{84.5}_{0.4}$	$78.5_{0.5}/75.2_{0.6}$	$\mathbf{94.3}_{0.4}/\mathbf{92.7}_{0.4}$	$87.1_{0.6}/85.2_{0.5}$	$92.6_{0.5}/\mathbf{90.6}_{0.5}$	$95.3_{0.2}/94.2_{0.2}$	88.9/87.0
GVICReg	$81.9_{0.3}/81.2_{0.4}$	$79.0_{0.6}/75.8_{1.8}$	$93.2_{0.4}/92.2_{0.9}$	$88.5_{0.6}/86.0_{0.6}$	$90.8_{0.5}/87.1_{1.4}$	$94.7_{0.3}/92.8_{0.5}$	88.0/85.8
GVICReg (Best Method from Chapter 4)	$85.1_{0.9}/84.0_{1.5}$	$\mathbf{79.6}_{0.7}/\mathbf{76.8}_{0.6}$	$93.5_{0.3}/92.5_{0.6}$	$\mathbf{89.3}_{0.3}/\mathbf{87.5}_{0.4}$	$\mathbf{92.7}_{0.3}/90.2_{0.6}$	$\mathbf{95.6}_{0.2}/\mathbf{94.3}_{0.1}$	**89.3/87.5**
BGRL [58]	$74.8_{1.8}/72.2_{1.9}$	$77.2_{1.5}/74.4_{1.5}$	$91.2_{0.8}/89.3_{1.1}$	$85.7_{0.7}/83.4_{1.4}$	$90.6_{0.9}/87.3_{1.4}$	$94.4_{0.4}/92.6_{0.5}$	85.6/83.2
BGRL (Best Method from Chapter 5)	$80.4_{3.7}/78.5_{3.5}$	$77.4_{1.4}/74.5_{1.5}$	$92.4_{1.1}/90.9_{1.5}$	$86.6_{1.5}/84.9_{2.3}$	$92.4_{0.7}/90.5_{1.2}$	$95.1_{0.2}/93.6_{0.2}$	87.4/85.5
BGRL (Best Method from Chapter 6)	$76.5_{1.9}/74.6_{2.3}$	$77.6_{1.0}/74.8_{1.1}$	$91.2_{0.8}/89.3_{1.1}$	$85.7_{0.7}/83.4_{1.4}$	$91.6_{1.0}/88.7_{1.1}$	$94.7_{0.6}/93.0_{0.6}$	86.2/84.0
GBT [4]	$75.8_{0.8}/74.9_{1.1}$	$77.6_{1.0}/74.9_{1.5}$	$91.7_{1.2}/90.1_{1.7}$	$87.0_{0.5}/85.4_{0.8}$	$91.7_{0.4}/88.2_{1.0}$	$94.3_{0.2}/92.5_{0.1}$	86.3/84.3
DGI [63]	$80.4_{4.2}/78.1_{3.8}$	$34.9_{2.6}/12.7_{2.6}$	$88.9_{1.6}/83.5_{3.1}$	$75.6_{1.5}/61.2_{3.9}$	$90.6_{0.7}/86.7_{2.0}$	$93.8_{0.7}/91.6_{1.0}$	77.4/67.0
GRACE [82]	$75.4_{2.4}/72.3_{2.3}$	$26.9_{1.9}/10.2_{1.9}$	$88.5_{1.5}/85.1_{2.0}$	$74.4_{3.7}/65.2_{8.7}$	OOM	OOM	N/A

Table 7.1: F1 Micro/Macro for different Contrastive/Non-contrastive SSL methods and the best results from Chapters 4, 5, and 6 alongside supervised results. Values are averaged over 5 different runs with 5 seed values. F1 is calculated for a Linear Logistic Regression node classifier trained on the embedding when the encoder is frozen to the weights at the end of the SSL training.

BGRL, the method proposed in Chapter 5 makes BGRL 20 percent slower on average and the method proposed in Chapter 6 does not introduce any extra complexity.

In summary, the proposed methods for enhancing VICReg appear to be the most promising as they can elevate the performance of vanilla VICReg to higher levels than all baselines and even the supervised scenario. Additionally, these methods can significantly reduce the complexity of vanilla VICReg. The proposed improvements for BGRL, on the other hand, can improve its performance to be better than all other baselines except for vanilla VICReg and its variations. However, both BGRL and its improved versions require significantly less computational complexity compared to VICReg. It is worth noting that the improvement in performance is more important than efficiency as SSL is of interest to large companies that can afford high computational costs if it leads to better performance. Based on results and dicussions, the methods proposed for enhancing VICReg are more practical and significant in practice.

Chapter 8

Future Work

This thesis has provided insights into the effectiveness of non-contrastive self-supervised learning in learning representations for different graph datasets and proposed methods for improving different methods under this category. However, there are still several paths for future research that can build on the findings of this thesis and extend its contributions.

One potential direction for future research is to investigate how we can decide between contrastive and non-contrastive methods for different parts of the graph. It has recently been shown that non-contrastive methods work better for non-linear manifolds, while contrastive ones work better for linear manifolds [2]. One continuation, inspired by the method based on Ricci flow proposed in Chapter 4, could be based on calculating Ricci flow for different nodes and using contrastive loss for nodes with Ricci close to zero, flat regions, and non-contrastive loss for nodes with large negative or positive Ricci flows and see if that leads to improvement.

Another promising direction for future research is to explore the effect of regularizer proposed in Chapter 5 on covariance-based methods to see the effect of that in facilitating the covariance whitening, how fast the loss term drops. Also, further analysis is needed to find out why the effect of regularization should be stronger for the first layer compared to the second one and how this changes by increasing number of layers.

Finally, future research can also focus on testing the early stopping target network update for different types of networks and architectures to see if the turning point moves. Also, this thesis could not provide proper results for two datasets. The reason behind this lack of improvement for those datasets must be analyzed in detail and a new receipe should be drawn for those datasets. The hypothesis of author is that the reason is the HSIC metric in those datasets have multiple peaks.

In summary, this thesis has proposed effective methods to improve different types of non-contrastive self-supervised learning, and has identified several directions for future work and further improvements. By continuing to explore the potential of proposed methods and continuation on them, researchers can contribute to the development of more effective and efficient SSL algorithms.

References

[1] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. *arXiv preprint arXiv:2202.03555*, 2022.

[2] Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *arXiv preprint arXiv:2205.11508*, 2022.

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[4] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *Knowledge-Based Systems*, 256:109631, 2022.

[5] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. Bipartite graph embedding via mutual information maximization. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 635–643, 2021.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[8] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4794–4802, 2019.

[9] PyGCL Contributors. Pygcl: A pytorch library for graph contrastive learning. https://github.com/PyGCL/PyGCL, 2022. Accessed: November, 2022.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Bin Dong, Jikai Hou, Yiping Lu, and Zhihua Zhang. Distillation ≈ early stopping? harvesting dark knowledge utilizing anisotropic information retrieval for overparameterized neural network. *arXiv preprint arXiv:1910.01255*, 2019.

[12] Yann Dubois, Tatsunori Hashimoto, Stefano Ermon, and Percy Liang. Improving self-supervised learning by characterizing idealized representations. *arXiv preprint arXiv:2209.06235*, 2022.

[13] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pages 3015–3024. PMLR, 2021.

[14] Robin Forman. Bochner's method for cell complexes and combinatorial ricci curvature. *Discrete and Computational Geometry*, 29(3):323–374, 2003.

[15] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. *arXiv preprint arXiv:2210.02885*, 2022.

[16] Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann Lecun. On the duality between contrastive and non-contrastive self-supervised learning. *arXiv preprint arXiv:2206.02574*, 2022.

[17] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

[18] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic Learning Theory: 16th International Conference, ALT 2005, Singapore, October 8-11, 2005. Proceedings 16*, pages 63–77. Springer, 2005.

[19] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[20] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[21] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pages 4116–4126. PMLR, 2020.

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[25] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[26] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867, 2020.

[27] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining (ICDM)*, pages 222–231. IEEE, 2020.

[28] Ming Jin, Yizhen Zheng, Yuan-Fang Li, Chen Gong, Chuan Zhou, and Shirui Pan. Multi-scale contrastive siamese networks for self-supervised graph representation learning. *arXiv preprint arXiv:2105.05682*, 2021.

[29] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

[30] William B Johnson. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[31] JongMok Kim, JooYoung Jang, Seunghyeon Seo, Jisoo Jeong, Jongkeun Na, and Nojun Kwak. Mum: Mix image tiles and unmix feature tiles for semi-supervised object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14512–14521, 2022.

[32] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[33] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[35] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[36] Namkyeong Lee, Junseok Lee, and Chanyoung Park. Augmentation-free self-supervised learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7372–7380, 2022.

[37] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic level protein structure with a language model. *bioRxiv*, 2022.

[38] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[39] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[40] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[41] Costas Mavromatis and George Karypis. Graph infoclust: Maximizing coarse-grain mutual information in graphs. In *Advances in Knowledge Discovery and Data Mining: 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11–14, 2021, Proceedings, Part I*, pages 541–553. Springer, 2021.

[42] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[43] Felix L Opolka, Aaron Solomon, Cătălina Cangea, Petar Veličković, Pietro Liò, and R Devon Hjelm. Spatio-temporal deep graph infomax. *arXiv preprint arXiv:1904.06316*, 2019.

[44] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[45] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[46] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[47] Yuxiang Ren, Bo Liu, Chao Huang, Peng Dai, Liefeng Bo, and Jiawei Zhang. Heterogeneous deep graph infomax. *arXiv preprint arXiv:1911.08538*, 2019.

[48] Facebook Research. VICReg: Code for Variance-Invariant Conditional Regression. https://github.com/facebookresearch/vicreg, 2021. Accessed: November, 2022.

[49] Mehdi Rezagholizadeh, Aref Jafari, Puneeth Salad, Pranav Sharma, Ali Saheb Pasand, and Ali Ghodsi. Pro-kd: Progressive distillation by following the footsteps of the teacher. *arXiv preprint arXiv:2110.08532*, 2021.

[50] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[51] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[52] Qinfeng Shi, Chunhua Shen, Rhys Hill, and Anton van den Hengel. Is margin preserved after random projection? *arXiv preprint arXiv:1206.4651*, 2012.

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[54] strumke. hsic_python. https://github.com/strumke/hsic_python, 2020. Accessed: November, 2022.

[55] Arjun Subramonian. Motif-driven contrastive learning of graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15980–15981, 2021.

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[57] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

[58] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021.

[59] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[60] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[62] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[63] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

[64] Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. Towards domain-agnostic contrastive learning. In *International Conference on Machine Learning*, pages 10530–10541. PMLR, 2021.

[65] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 391–408, 2018.

[66] Chaofei Wang, Qisen Yang, Rui Huang, Shiji Song, and Gao Huang. Efficient knowledge distillation from model checkpoints. *arXiv preprint arXiv:2210.06458*, 2022.

[67] Chenguang Wang and Ziwen Liu. Learning graph representation by aggregating subgraphs via mutual information maximization. *arXiv preprint arXiv:2103.13125*, 2021.

[68] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[69] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

[70] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021.

[71] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[72] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[73] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[74] Jiaqi Zeng and Pengtao Xie. Contrastive self-supervised learning for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10824–10832, 2021.

[75] Shuangfei Zhai, Navdeep Jaitly, Jason Ramapuram, Dan Busbridge, Tatiana Likhomanenko, Joseph Yitan Cheng, Walter Talbott, Chen Huang, Hanlin Goh, and Joshua Susskind. Position prediction as an effective pretraining strategy. *arXiv preprint arXiv:2207.07611*, 2022.

[76] Aston Zhang, Alvin Chan, Yi Tay, Jie Fu, Shuohang Wang, Shuai Zhang, Huajie Shao, Shuochao Yao, and Roy Ka-Wei Lee. On orthogonality constraints for transformers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, volume 2, pages 375–382. Association for Computational Linguistics, 2021.

[77] Hanlin Zhang, Shuai Lin, Weiyang Liu, Pan Zhou, Jian Tang, Xiaodan Liang, and Eric P Xing. Iterative graph self-distillation. *arXiv preprint arXiv:2010.12609*, 2020.

[78] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[79] Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Curriculum learning by optimizing learning dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 433–441. PMLR, 2021.

[80] Qikui Zhu, Bo Du, and Pingkun Yan. Self-supervised training of graph convolutional networks. *arXiv preprint arXiv:2006.02380*, 2020.

[81] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116*, 2021.

[82] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

[83] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.

APPENDICES

.1 Experimental Detail of Results in Chapter 4

The base code for GVICReg is GBT combined with VICReg loss function taken from PyGCL library [9]. However, expander module added from the official repository of VICReg [48] as it was missing in the PyGCL library.

.1.1 Model Architecture

The model architecture consists of a graph convolutional neural network with a single hidden layer. The input dimension of the graph convolutional neural network is equal to the number of features in the dataset, and the output dimension is 256. The output of the graph convolutional neural network is passed through an expander module, which consists of two fully connected layers with ReLU activation. The final dimension of the expander is a hyper-parameter taking values 1000, 4000, or 7000 in various experiments. In the beginning, the static graph data is passed through two different augmentations, edge removal with probability 0.5, and feature masking with probability 0.1 before being passed to the encoder.

.1.2 Model Training

The optimizer used is the Adam optimizer with a learning rate of 5e-4. The learning rate is annealed using cosine learning rate scheduler with linear warm-up [40]. For all of the datasets, number of warm-up epochs is 100. The training is performed for a maximum of 1000 epochs. The factor for different loss terms are as follows; sim_weight: 25.0, var_weight: 25.0, cov_weight: 1.0, eps: 1e-4.

.1.3 Evaluation

The performance of the model is evaluated using the linear probing with a freezed encoder an fitting a logistic regression classifier for node classification downstream task into the embedding. The evaluation is performed on a test set, which consists of 20 percent of the nodes in graph. The evaluation metric used is the F1 micro and macro in predicting the node classification task. The training and evaluation are performed on a machine with Tesla V100 GPUs provided by Huawei Noah's Ark Lab.

.2 Experimental Detail of Results in Chapter 5

The base code is from PyGCL library [9], Local to Local code example for BGRL.

.2.1 Model Architecture

The model architecture consists of a graph convolutional neural network with a single hidden layer. The input dimension of the graph convolutional neural network is equal to the number of features in the dataset, and the output dimension is 256. The output of the online encoder is passed through a predictor module, which consists of one fully connected layer with Parametric ReLU activation. The final dimension of the predictor is the same as the output dimension of the online encoder. In the beginning, the static graph data is passed through two different augmentations, edge removal with probability 0.5, and feature masking with probability 0.1 before being passed to the encoder.

.2.2 Model Training

The optimizer used is the Adam optimizer with a learning rate of 5e-4 for WikiCS and 1e-4 for other datasets. The learning rate is annealed using cosine learning rate scheduler with linear warm-up [40]. For all of the datasets, number of warm-up epochs is 100. The training is performed for a maximum of 1000 epochs. The regularization factor for layers one and two of the graph neural network take values 0.2, 0.5, or 0.8 in different experiments. The momentum in Exponential Moving Average (EMA) is 0.99.

.2.3 Evaluation

The performance of the model is evaluated using the linear probing with a freezed encoder an fitting a logistic regression classifier for node classification downstream task into the embedding. The evaluation is performed on a test set, which consists of 20 percent of the nodes in graph. The evaluation metric used is the F1 micro and macro in predicting the node classification task. The training and evaluation are performed on a machine with Tesla V100 GPUs provided by Huawei Noah's Ark Lab.

.3 Experimental Detail of results in Chapter 6

The base code is from PyGCL library [9], Local to Local code example for BGRL.

.3.1 Model Architecture

The model architecture consists of a graph convolutional neural network with a single hidden layer. The input dimension of the graph convolutional neural network is equal to the number of features in the dataset, and the output dimension is 256. The output of the online encoder is passed through a predictor module, which consists of one fully connected layer with Parametric ReLU activation. The final dimension of the predictor is the same as the output dimension of the online encoder. In the beginning, the static graph data is passed through two different augmentations, edge removal with probability 0.5, and feature masking with probability 0.1 before being passed to the encoder.

.3.2 Model Training

The optimizer used is the Adam optimizer with a learning rate of 5e-4. The learning rate is annealed using cosine learning rate scheduler with linear warm-up [40]. For all of the datasets, number of warm-up epochs is 100. The training is performed for a maximum of 1000 epochs. The momentum in Exponential Moving Average (EMA) is 0.99. The intervals of recalculating HSIC is 5. In order to plot HSIC values, non-linear gaussain kernel with default parameters from github repository [54] used. In general methodology, vector inner products, linear kernel, used. During training, The ratio of nodes sampled for calculating HSIC is 10 percent. The smoothing factor of HSIC is 0.99

.3.3 Evaluation

The performance of the model is evaluated using the linear probing with a freezed encoder an fitting a logistic regression classifier for node classification downstream task into the embedding. The evaluation is performed on a test set, which consists of 20 percent of the nodes in graph. The evaluation metric used is the F1 micro and macro in predicting the node classification task. The training and evaluation are performed on a machine with Tesla V100 GPUs provided by Huawei Noah's Ark Lab.