TASK-AGNOSTIC GRAPH NEURAL NETWORK EVALUATION VIA ADVERSARIAL COLLABORATION

Xiangyu Zhao[1], Hannes Stärk[2], Dominique Beaini[3], Yiren Zhao[1], Pietro Liò[4]
[1]Department of Electrical and Electronic Engineering, Imperial College London
[2]CSAIL, Massachusetts Institute of Technology
[3]Valence Discovery, Mila, Université de Montréal
[4]Department of Computer Science and Technology, University of Cambridge
[1]{x.zhao22,a.zhao}@imperial.ac.uk,
[2]hstark@mit.edu, [4]pietro.lio@cl.cam.ac.uk

ABSTRACT

It has been increasingly demanding to develop reliable methods to evaluate the progress of Graph Neural Network (GNN) research for molecular representation learning. Existing GNN benchmarking methods for molecular representation learning focus on comparing the GNNs' performances on some node/graph classification/regression tasks on certain datasets. However, there lacks a principled, task-agnostic method to directly compare two GNNs. Additionally, most of the existing self-supervised learning works incorporate handcrafted augmentations to the data, which has several severe difficulties to be applied on graphs due to their unique characteristics. To address the aforementioned issues, we propose GraphAC (Graph Adversarial Collaboration) – a conceptually novel, principled, task-agnostic, and stable framework for evaluating GNNs through contrastive self-supervision. We introduce a novel objective function: the Competitive Barlow Twins, that allow two GNNs to jointly update themselves from direct competitions against each other. GraphAC succeeds in distinguishing GNNs of different expressiveness across various aspects, and has demonstrated to be a principled and reliable GNN evaluation method, without necessitating any augmentations.

1 INTRODUCTION

Graph Neural Networks (GNNs) have gained immense research attention in recent years, leading to significant progress that has been successfully implemented across a broad range of fields, including chemistry (Gilmer et al., 2017) and biology (Stokes et al., 2020). This makes GNNs important tools in the molecular representation learning landscape, and improving their development is of great interest to the biomedical machine learning community. As this field of research grows rapidly, it becomes crucial to develop general and reliable evaluation methods to facilitate GNN research and quantify the performances of various GNN architectures in the context of molecular graph data.

Existing approaches on benchmarking GNNs focus on comparing the GNNs with respect to their performances on some node/graph classification/regression tasks in a collection of molecular/protein/DNA datasets (Dwivedi et al., 2020; Hu et al., 2020). However, these approaches can be limited in the following ways: 1) classification/regression tasks are naturally simple in terms of the combinatorial complexity, and cannot fully challenge the GNNs to learn from the graphs; 2) there exist many different molecular prediction datasets used by various works on GNNs, but there lacks a standardized way to compare those results on different datasets, and sometimes it is not feasible to evaluate on all of them; and 3) Deep Neural Networks (DNNs) generally rely on high-quality labeled data for high-performance training, but unfortunately, large datasets almost always contain examples with inaccurate, incorrect or even missing labels, known as label noises. It has been demonstrated that most DNNs, especially GNNs, are vulnerable to noisy labels, resulting in drastically lowered generalization performances (Dai et al., 2021; NT et al., 2019; Zhang et al., 2017; Zhang et al., 2020). Consequently, it is highly desirable to develop GNN evaluation methods that can effectively exploit the training data, without necessitating any labels.

There have been some attempts to evaluate the capacities of GNNs against theoretical tests such as the Weisfeiler-Lehman (1-WL) graph isomorphism test (Weisfeiler & Leman, 1968; Xu et al., 2019; Dwivedi et al., 2020), but the information they provide is normally limited. These methods are normally designed for small-scale benchmarks, and measure GNNs' ability to detect patterns, substructures and clusters (Dwivedi et al., 2020), or graph properties such as diameter, eccentricity, and spectral radius (Corso et al., 2020). However, these approaches cannot be used consistently, since certain GNN types can exploit this information as positional encodings to directly cheat the task (Kreuzer et al., 2021; Bodnar et al., 2021; Dwivedi et al., 2022). Therefore, there is a need for a task-agnostic evaluation of the expressiveness of GNNs.

Designing principled self-supervised learning (SSL) methods for graphs is also a challenging task. As labeled data can be expensive, limited or even unavailable in many real-word scenarios, it has become increasingly demanding to develop powerful SSL methods on graphs. A lot of successful SSL works on graphs (Veličković et al., 2019; Sun et al., 2020; You et al., 2020; 2021; Xu et al., 2021) rely on applying handcrafted augmentations to the graphs, which are further described in Section 2. However, there are several key difficulties in applying augmentations to graphs. Firstly, there exists no universal augmentation that works across all types of graphs. Secondly, graphs are not invariant to augmentations like images: applying filters or rotating an image still preserves its essential invariances, but even a tiny augmentation on a graph can significantly change its topological structure or intrinsic properties. Another class of SSL methods on graphs that do not require augmentations (Stärk et al., 2021) relies on exploiting the physical properties of small molecules, and cannot be generalized to other graph types, such as proteins or DNAs. Therefore, it is highly desirable to develop a principled and generalizable SSL framework that does not require handcrafted augmentations.

Our solution: Graph Adversarial Collaboration (GraphAC). We address both aforementioned questions by proposing a conceptually novel, principled, and task-agnostic framework for evaluating GNNs in the context of molecular data, via a self-supervised, adversarial collaboration manner, without the need of handcrafted augmentations. In the GraphAC framework, two GNNs directly compete against each other on the same unlabeled graphs. The more expressive GNN produces more complex and informative graph embeddings and is thereby able to win the game. We make the following contributions in this paper:

- We introduce a novel principle for evaluating GNNs, by having them directly compete against each other in a self-supervised manner, rather than comparing them using a scoreboard of training performances on some datasets;
- Inspired by the novel principle, we propose a new architecture and an original modification to the existing Barlow Twins loss (Zbontar et al., 2021) that enables the GNNs to stably compete against each other, while ensuring that more expressive GNNs can always win;
- We provide the very first framework for evaluating GNN expressiveness directly on the molecular graph data, without the need of a specific downstream task, or theoretical representations of these molecular graphs;
- We develop a principled contrastive learning framework without needing any handcrafted augmentations, which is also generalizable to various types of GNNs.

2 Related Work

Contrastive Self-Supervised Learning To the best of our knowledge, there has not been any published attempt to develop a method for evaluating deep learning models by directly competing two models in a contrastive self-supervised environment, no matter in the general machine learning or the graph representation learning communities. Current approaches center around competing with a set of baselines that evaluate a limited number of performance metrics on a fixed number of benchmark or datasets. However, the state-of-the-arts in contrastive SSL, both in the non-graph and the graph domains, is still relevant to this work. Their successes in building contrastive learning architectures can help us build a principled, task-agnostic graph model evaluation framework. It is worth noting that GraphAC is a task-agnostic evaluation of GNNs, and not an optimization for downstream tasks. Therefore, the performance of state-of-the-art contrastive SSL on downstream tasks is not relevant.

Many works on contrastive SSL on graphs (Veličković et al., 2019; Sun et al., 2020; You et al., 2020; 2021; Xu et al., 2021) are inspired from the successes of the idea of mutual information maximization between two representations of the same data, with manually applied augmentations, in the non-graph domain (Gutmann & Hyvärinen, 2010; Oord et al., 2018; Hjelm et al., 2019; He et al., 2020; Chen et al., 2020). Those works vary in augmentation strategies and mutual information estimators. However, in order to prevent *information collapse* (models ignoring the input data and outputting identical and constant vectors), all those works require large batch sizes or memory banks, and extensive searches for augmentations and negative pairs, making them very costly. Besides, applying augmentations to graphs can be much harder than to images, since there exists no universal augmentation that works for all graph types. Furthermore, graphs are not noise-invariant – small changes to a graph can significantly alter its topological structure, especially for small graphs such as molecules. While existing research has developed fine-grained graph augmentations, it is still almost impossible to apply these augmentations while preserving the graph's intrinsic properties, such as the chemical properties of molecules. Moreover, graph augmentations can deviate the data from real-word distributions, since they introduce arbitrary human knowledge not provided by the training data. Stärk et al. (2021) propose a noiseless framework by maximizing the mutual information between the embedding of a 2D molecular graph and the embedding capturing its 3D structure, but it is specific to the physical properties of molecules, and cannot be generalized to other domains. Consequently, *there is a need for a principled contrastive SSL framework that can be applied across a diversity of graph types without requiring any augmentations.*

Barlow Twins Zbontar et al. (2021) introduce an alternative approach to prevent information collapse, by maximizing the information contents within the representations. In Barlow Twins, for a given input batch $\mathbf{X} \in \mathbb{R}^{N_b \times d_\mathbf{x}}$ of batch size N_b and dimension $d_\mathbf{x}$, two batches of distorted views $\tilde{\mathbf{X}}^A$ and $\tilde{\mathbf{X}}^B$ of $\mathbf{X}$ are generated using manual data augmentation. The two batches of distorted views $\tilde{\mathbf{X}}_A$ and $\tilde{\mathbf{X}}^B$ are fed into two separate models, which produces batches of d-dimensional embeddings $\mathbf{H}^A, \mathbf{H}^B \in \mathbb{R}^{N_b \times d}$. For simplicity, the features in both $\mathbf{H}^A$ and $\mathbf{H}^B$ are assumed to have a mean of zero across the batch. Barlow Twins then computes the cross-correlation $\mathbf{C} \in \mathbb{R}^{d \times d}$ matrix between $\mathbf{H}^A$ and $\mathbf{H}^B$ along the batch dimension. It then applies the following loss function on $\mathbf{C}$:

$$\mathcal{L}_{\text{BT}} = \underbrace{\sum_{i}^{d}(1-\mathbf{C}_{i,i})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_{i}^{d}\sum_{j \neq i}^{d}\mathbf{C}_{i,j}^2}_{\text{redundancy reduction term}} \tag{1}$$

The invariance term of the Barlow Twins loss enforces the two output embeddings to be similar by pushing the on-diagonal elements of the cross-correlation matrix towards one. Meanwhile, the redundancy reduction term attempts to ensure that the off-diagonal elements of the cross-correlation matrix closer to zero, thereby decorrelating the different features of the embeddings, so that the embeddings contain non-redundant information about the data. This process implicitly maximizes the amount of information contained within the embedding vectors.

Variance-Invariance-Covariance Regularization (VICReg) Bardes et al. (2022) build VICReg based on the principle of preserving the information content of the representations, similar to Barlow Twins. The architecture of VICReg is the same as Barlow Twins, except that it uses three regularization terms in its objective function: 1) invariance regularization $\mathcal{L}_{\text{Inv}}$: the mean square Euclidean distance between the output embeddings; 2) variance regularization $\mathcal{L}_{\text{Var}}$: a hinge loss to maintain the standard deviation of the embeddings along the batch dimension close to 1, which forces the output embeddings within a batch to be different; and 3) covariance regularization $\mathcal{L}_{\text{Cov}}$: the sum of the squared off-diagonal elements of the covariance matrix, with a factor $1/d$ to scale the term as a function of the feature dimension. This term attracts the covariances between every pair of features of the embeddings over a batch towards zero, decorrelating the different features of the embeddings, thus preventing them from encoding similar information. The overall loss function for VICReg is then a weighted sum of the invariance, variance and covariance regularization terms:

$$\mathcal{L}_{\text{VICReg}} = \lambda\mathcal{L}_{\text{Inv}} + \mu\mathcal{L}_{\text{Var}} + \nu\mathcal{L}_{\text{Cov}} \tag{2}$$

where $\lambda, \mu, \nu > 0$ are hyperparameters controlling the importance of each term in the loss.

3 METHOD

The intuition behind Graph Adversarial Collaboration (GraphAC) is to have different GNNs competing against each other on the same *unlabeled* graphs, and encouraging more expressive GNNs to produce more complex and informative graph embeddings. This can be measured by the ability to predict other GNNs' graph embeddings from a GNN's own graph embeddings: if a GNN can predict another GNN's graph embeddings from its own graph embeddings better than the other way round, then its graph embeddings can be deemed more complex and informative than the other GNN's graph embeddings, and therefore, more expressive. The two GNNs collaborate by predicting each other's output graph embeddings, and compete adversarially to prevent the other GNNs from predicting their own graph embeddings. To solve the challenge of maximizing the performance differences between different GNNs while ensuring stable training, we introduce the *Competitive Barlow Twins*, a novel pair of loss functions modified from the Barlow Twins described in Section 2.

3.1 COMPETITIVE BARLOW TWINS

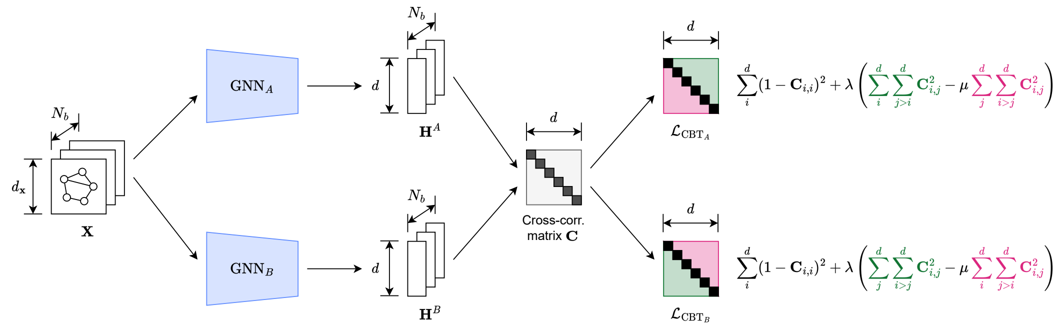
A deeper analysis of the Barlow Twins shows that, according to Zbontar et al. (2021)'s definition for the cross-correlation matrix $\mathbf{C}$ between the embeddings,

$$\mathbf{C}_{i,j} = \frac{\sum_b^{N_b} \mathbf{H}^A_{b,i}\mathbf{H}^B_{b,j}}{\sqrt{\sum_b^{N_b}\left(\mathbf{H}^A_{b,i}\right)^2}\sqrt{\sum_b^{N_b}\left(\mathbf{H}^B_{b,j}\right)^2}} \tag{3}$$

the (i,j)-th entry $\mathbf{C}_{i,j}$ of the cross-correlation matrix represents how much feature i of the first model's output embeddings $\mathbf{H}^A$ correlates to feature j of the second model's output embeddings $\mathbf{H}^B$. Therefore, for output embeddings of dimensionality d, the row $\mathbf{C}_{i,[i+1:d]}$ at the upper-triangle of the cross-correlation matrix represents how much feature i of $\mathbf{H}^A$ correlates to features $i+1$ to d of $\mathbf{H}^B$. For i close to one, the row $\mathbf{C}_{i,[i+1:d]}$ in the upper-triangle becomes much longer, and thus the i-th feature of $\mathbf{H}^A$ represented by that piece of the row correlates to the majority of the features of $\mathbf{H}^B$. For i close to d, the row at the upper-triangle becomes much shorter, and thus the i-th feature of $\mathbf{H}^A$ represented by that piece of the row correlates to very few features of $\mathbf{H}^B$. This means that the smaller-indexed features of the first model's output embeddings $\mathbf{H}^A$ become the more important features, if monitored by the upper-triangle of the cross-correlation matrix. Similarly, in the lower-triangle of the cross-correlation matrix, the column $\mathbf{C}_{[j+1:d],j}$ represents how much feature j of $\mathbf{H}^B$ correlates to features $j+1$ to d of $\mathbf{H}^A$, making the smaller-indexed features of the second model's output embeddings also becoming the more important features. It can therefore be hypothesized that under this upper-lower-triangle setting, the first few features of both models' output embeddings are targeted at capturing the low frequency signals as they are easier to predict, and the later features are set to capture the high frequency signals, which are harder to predict.

Based on the above findings, if the two triangles of the cross-correlation matrix are summed, then the sum of each triangle is dominated by the first few rows/columns, as they contain the most entries. Therefore, the sum of the triangle provides a measure of how much a model's output features correlate to the other model's output features, weighted by importance, since there are more elements in the triangle corresponding to the more important features. Consequently, a larger sum implies a better correlation of a model's most important features in its output embeddings with the other model's output features, which implies a stronger ability to predict the other model's output embeddings from its own output embeddings. This naturally yields the definition of the Competitive Barlow Twins loss, which preserves the invariance term in the original Barlow Twins, but replaces the off-diagonal sum with the difference between the upper-triangle and the lower-triangle of the cross-correlation matrix:

$$\begin{aligned}\mathcal{L}_{\mathrm{CBT}_A} &= \sum_i^d (1-\mathbf{C}_{i,i})^2 + \lambda\left(\sum_i^d\sum_{j>i}^d \mathbf{C}_{i,j}^2 - \mu\sum_j^d\sum_{i>j}^d \mathbf{C}_{i,j}^2\right)\\ \mathcal{L}_{\mathrm{CBT}_B} &= \sum_i^d (1-\mathbf{C}_{i,i})^2 + \lambda\left(\sum_j^d\sum_{i>j}^d \mathbf{C}_{i,j}^2 - \mu\sum_i^d\sum_{j>i}^d \mathbf{C}_{i,j}^2\right)\end{aligned} \tag{4}$$



N_b
$d_{\mathbf{x}}$
$\mathbf{X}$
GNN$_A$
GNN$_B$
d
N_b
$\mathbf{H}^A$
$\mathbf{H}^B$
d
Cross-corr. matrix $\mathbf{C}$
$\mathcal{L}_{\text{CBT}_A}$
$\mathcal{L}_{\text{CBT}_B}$
$\sum_i^d (1-\mathbf{C}_{i,i})^2 + \lambda \left(\sum_i^d \sum_{j>i}^d \mathbf{C}_{i,j}^2 - \mu \sum_j^d \sum_{i>j}^d \mathbf{C}_{i,j}^2 \right)$
$\sum_i^d (1-\mathbf{C}_{i,i})^2 + \lambda \left(\sum_j^d \sum_{i>j}^d \mathbf{C}_{i,j}^2 - \mu \sum_i^d \sum_{j>i}^d \mathbf{C}_{i,j}^2 \right)$

Figure 1: Architecture of GraphAC's framework. Batched unlabeled graph data $\mathbf{X} \in \mathbb{R}^{N_b \times d_{\mathbf{x}}}$ are fed into two different GNNs, obtaining two different batched embeddings $\mathbf{H}^A, \mathbf{H}^B \in \mathbb{R}^{N_b \times d}$. Then, the cross-correlation matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$ between $\mathbf{H}^A$ and $\mathbf{H}^B$ along the batch dimension is calculated. The Competitive Barlow Twins losses, $\mathcal{L}_{\text{CBT}_A}$ and $\mathcal{L}_{\text{CBT}_B}$, are then computed by pushing the on-diagonal elements of $\mathbf{C}$ towards one, while adding the difference between the upper/lower-triangles of $\mathbf{C}$. The pair of Competitive Barlow Twins losses are then used to update the two GNNs respectively. Details about the GraphAC framework are described in Section 3.

where $\lambda, \mu > 0$ are weighting coefficients, with λ inherited from the original Barlow Twins, and μ trading off the importance of correlating the opponent GNN's output features (collaboration) and preventing the opponent GNN from correlating the GNN's own output features (competition). Although the above reasoning discourages the use of different weights on the sums of triangles, which is also confirmed by the hyperparameter tuning results described in Appendix D.1, we still include μ in the definition of the Competitive Barlow Twins for the purpose of hyperparameter tuning.

Another important enhancement by the Competitive Barlow Twins is that, since the triangles make the smaller-indexed features of both models' output embeddings the more important features, both models' output embeddings are ordered by feature importance. This ordering prevents the models from simply permuting the entries of their output embeddings to avoid being predicted by their opponent models, and makes the training much more stable.

3.2 Proposed Framework

The architecture of the GraphAC framework is illustrated in Figure 1. We also include the covariance regularization term from VICReg in GraphAC's loss functions because it decorrelates different feature dimensions within each output graph embeddings, and forces the graph embeddings to be fully used to capture graph information. Therefore, we obtain the following definitions for GraphAC's final loss functions:

$$\begin{aligned} \mathcal{L}_{\text{GNN}_A} &= \alpha \mathcal{L}_{\text{CBT}_A} + \beta \mathcal{L}_{\text{Cov}} \\ \mathcal{L}_{\text{GNN}_B} &= \alpha \mathcal{L}_{\text{CBT}_B} + \beta \mathcal{L}_{\text{Cov}} \end{aligned} \tag{5}$$

where $\alpha, \beta > 0$ are weighting coefficients, and $\mathcal{L}_{\text{Cov}}$ is the VICReg covariance regularization term defined in Section 2. The invariance regularization term from VICReg is not included in the loss functions, because the effect of the invariance regularization term has already been achieved by the invariance term from the Competitive Barlow Twins. We also do not include the variance regularization term from VICReg in GraphAC's loss functions, because it forces the variance of the embeddings over a batch to be above a given threshold, which can potentially cause the training to be unstable. Although the Competitive Barlow Twins losses can enable stable training of the models, and can counter the instability caused by the VICReg variance regularization term, we still do not include that term in the loss functions, because it is used to prevent the models from producing the same embedding vectors for samples within a batch, which did not occur in this framework.

4 Evaluation

4.1 Data Preparation

In order for GraphAC to provide the most realistic evaluations of the GNNs, the datasets used for evaluating it should be application-oriented with real-word implications. To ensure GraphAC can discriminate between GNNs with statistical significance, the datasets should be large-scale of high quality. Moreoever, since GraphAC is designed for graph-level prediction, the datasets should also be constructed for graph-level prediction, which means that they should contain a large number of relatively small graphs. Finally, in order for GraphAC to support GNNs both with and without edge features, and allowing it to study the effect for a GNN with edge features, the datasets should provide both node and edge features for the graphs. Based on the above requirements, GraphAC is best suited to drug-like small molecular datasets for the following reasons:

- Molecules can naturally be represented as graphs;
- Molecular property prediction is a fundamental task within many important applications in chemistry, biology and medicine;
- There is a vast variety of molecules in the world, and the drug-like small molecular graphs can be trained efficiently without requiring extensive GPU resources.

Therefore, we use the largest molecular property prediction dataset from OGB (Hu et al., 2020), namely the `ogbg-molpcba` dataset. It contains 437,929 drug-like molecules, with on average 26.0 atoms (nodes) and 28.1 bonds (edges) per molecule (graph). Despite the fact that the dataset comprises multiple classification tasks and its class balance is highly skewed, these factors do not affect GraphAC's evaluation, as only the molecular graphs are used while their labels are discarded. In order to confirm that GraphAC is indeed task-agnostic, we also evaluate GraphAC on the `ogbg-code2` dataset, which contains 452,741 abstract syntax trees obtained from Python method definitions, with on average 125.2 nodes and 124.2 edges per tree.

4.2 Experimental Setup

Training and experiments were conducted on an NVIDIA A100 SXM GPU with 80GB graphics memory. All experiments were trained for 50 epochs. The pseudocode for the core training algorithm of GraphAC can be found in Appendix A, and details of the hyperparameter tuning experiments are described in Appendix D.1. The source code for GraphAC is publicly available at `https://github.com/VictorZXY/GraphAC`.

In order to fairly compare the GNNs as well as to evaluate GraphAC's ability in distinguishing different components of a GNN, the experiments were split into five groups of controlled experiments. In each group, one component of the GNN is varied, while all other components are fixed. The five aspects of a GNN evaluated by GraphAC are:

- **Number of GNN layers:** in this group of experiments, PNAs (Corso et al., 2020) with 2, 4, 6, 8, and 10 layers compete in the GraphAC framework on a double round-robin basis, with one extra experiment performed for each model to compete against itself. All PNAs have a fixed hidden dimension of 256, and use the combination of [max, mean, sum] as their aggregators. All PNAs use [identity, amplification, attenuation] as their scalers, and their message passing functions are parametrized by 2-layer MLPs. We choose PNAs for evaluation due to their flexibility and state-of-the-art performance on molecular tasks.
- **Hidden dimensions:** in this group of experiments, 4-layer PNAs with hidden dimenions of 16, 32, 64, 128, and 256 competed in the GraphAC framework on a double round-robin basis, with an additional experiment performed for each model to compete against itself. All PNAs utilize [max, mean, sum] as their aggregators.
- **Aggregators:** in this group of experiments, four 4-layer PNAs with 64 hidden dimensions, and [max], [mean], [sum], [max, mean, sum] as their aggregators respectively, are set to compete in the GraphAC framework on a double round-robin basis, again with one extra experiment for each model to compete against itself.

Table 1: Loss differences of the conducted experiments. Negative value means GNN$_A$ wins the game, and positive value means GNN$_B$ wins the game. Greater absolute value indicates larger gap in expressiveness determined by GraphAC. The consistent gradient from bottom left to upper right clearly indicates that GraphAC genuinely favors more expressive GNNs. Since the `ogbg-code2` dataset does not contain any edge features, no experiments regarding the inclusion of edge features for the `ogbg-code2` dataset are conducted.

		#Layers in GNN$_B$ (256 hidden dims, aggregators: [max, mean, sum])									
		`ogbg-molpcba`					`ogbg-code2`				
		2	4	6	8	10	2	4	6	8	10
#Layers in GNN$_A$ (256 hidden dims, [max, mean, sum])	2	-0.04	1.29	1.46	1.88	1.97	-0.01	0.31	0.36	0.63	0.65
	4	-1.31	0.03	0.80	1.40	1.75	-0.25	0.01	0.34	0.40	0.52
	6	-1.48	-0.84	-0.01	0.67	0.81	-0.36	-0.31	-0.00	0.24	0.34
	8	-1.69	-1.28	-0.41	0.03	0.56	-0.59	-0.40	-0.27	0.00	0.19
	10	-2.01	-1.75	-1.08	-0.67	-0.00	-0.67	-0.46	-0.35	-0.18	-0.08

		Hidden dims in GNN$_B$ (4 layers, aggregators: [max, mean, sum])									
		`ogbg-molpcba`					`ogbg-code2`				
		16	32	64	128	256	16	32	64	128	256
Hidden dims in GNN$_A$ (4 layers, [max, mean, sum])	16	0.02	1.39	1.88	2.36	2.55	0.01	0.61	0.81	0.86	0.93
	32	-1.24	0.00	0.93	1.61	2.09	-0.66	-0.04	0.53	0.74	0.88
	64	-2.29	-0.89	0.02	1.21	1.81	-0.73	-0.66	-0.00	0.39	0.62
	128	-2.49	-1.61	-1.00	-0.01	1.49	-0.81	-0.72	-0.41	-0.03	0.42
	256	-2.54	-2.04	-1.70	-1.33	-0.01	-0.91	-0.78	-0.64	-0.40	-0.03

		Aggregators in GNN$_B$ (4 layers, 64 hidden dims)							
		`ogbg-molpcba`				`ogbg-code2`			
		[max]	[mean]	[sum]	Comb.	[max]	[mean]	[sum]	Comb.
Aggregators in GNN$_A$ (4 layers, 64 hidden dims)	[max]	-0.03	0.18	0.31	0.34	-0.02	0.14	0.30	0.34
	[mean]	-0.17	-0.01	0.24	0.33	-0.15	-0.00	0.17	0.20
	[sum]	-0.30	-0.25	0.05	0.23	-0.30	-0.12	-0.01	0.15
	Comb.	-0.33	-0.29	-0.24	0.02	-0.41	-0.28	-0.15	-0.00

`ogbg-molpcba`		Edge feat. − No edge feat. Hidden dims (PNA)				GNN architecture (4 layers, 64 hidden dims)					
						`ogbg-molpcba`			`ogbg-code2`		
		64	128	256		GCN	GIN	PNA	GCN	GIN	PNA
#Layers (PNA)	4	-0.42	-0.34	-0.20	GCN	-0.07	0.34	0.47	-0.02	1.06	1.66
	6	-0.45	-0.28	-0.21	GIN	-0.34	-0.03	0.44	-1.19	-0.01	1.40
	8	-0.42	-0.26	-0.20	PNA	-0.59	-0.42	0.02	-1.58	-1.43	-0.00

- **GNN architectures:** in this group of architectures, PNA, GIN (Xu et al., 2019) and GCN (Kipf & Welling, 2017), all with 4 layers and 64 hidden dimensions, and PNA with [max, mean, sum] as aggregators, are set to compete in the GraphAC framework on a double round-robin basis, again with one extra experiment for each model to compete against itself.
- **Edge features:** in this group of experiments, PNAs with 4, 6, and 8 layers, and hidden dimensions ranging from 64, 128, and 256 are used. All PNAs use [max, mean, sum] as their aggregators. In each experiment, PNAs with the same structure, but one with the edge features and the other without the edge features, are inserted into GraphAC for competitions.

4.3 Results

Training took from 1.7 hours (2-layer PNA vs. 2-layer PNA) to 4.2 hours (10-layer PNA vs. 10-layer PNA). The details of the training process are described in Appendix D.2. The training outcomes indicate that the more expressive GNN can continuously achieve a lower loss in our framework, and that GraphAC can successfully avoid information collapse.

The results of the experiments, recorded as $\mathcal{L}_{\text{GNN (edge features)}} - \mathcal{L}_{\text{GNN (no edge features)}}$ for the edge features group and $\mathcal{L}_{\text{GNN}_A} - \mathcal{L}_{\text{GNN}_B}$ for all other groups, are reported in Table 1. Another two tables containing detailed results of the experiments can be found in Appendix B. These results demonstrate that *GraphAC can successfully distinguish GNNs of different expressiveness across various aspects, and consistently favors the more expressive GNNs*: 1) deeper GNNs; 2) GNNs with larger hidden dimensions; 3) combining multiple aggregators $>$ sum $>$ mean $>$ max as aggregators (Xu et al., 2019; Corso et al., 2020); 4) PNA $>$ GIN $>$ GCN (Xu et al., 2019; Corso et al., 2020); and 5) GNNs

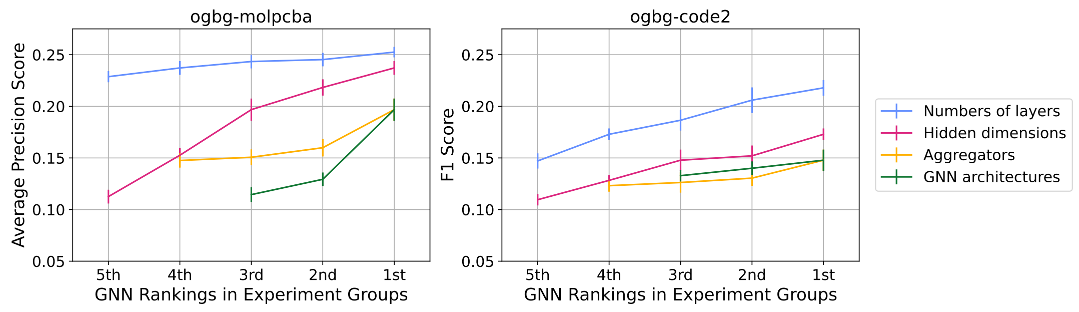


Figure 2: Correlation plots of GraphAC's GNN rankings with the GNNs' performances. The positive correlations show that GraphAC's expressiveness rankings align well with the GNNs' performances.

that include edge features. Furthermore, regardless of the ordering of the GNNs in the framework, there is no distinction between GNN_A or GNN_B; this means that for all pairs of experiments, even if the order is switched, the absolute loss differences still remain roughly equal but only with the sign flipped. These observations suggest that *GraphAC can genuinely distinguish different GNNs, regardless of their ordering in the framework*. Additionally, for the experiments with the same GNNs competing, GraphAC produced loss differences close to zero, demonstrating its ability to allow GNNs with the same expressiveness to tie, rather than falsely deciding a winner. Moreover, for every three GNNs: GNN_A, GNN_B and GNN_C in the experiments, if their expressiveness can be ordered as $\text{GNN}_A > \text{GNN}_B > \text{GNN}_C$, then there is also $|\mathcal{L}_{\text{GNN}_A} - \mathcal{L}_{\text{GNN}_C}| > |\mathcal{L}_{\text{GNN}_A} - \mathcal{L}_{\text{GNN}_B}|$ produced by GraphAC. This phenomenon shows that *GraphAC is able to produce a total ordering of all GNNs*, which further demonstrates its credibility in GNN evaluation.

Some notable group-specific observations are as follows:

Different aggregators It is observed that that the loss differences in the aggregators group are less significant than in the numbers of GNN layers and hidden dimensions groups. This is possibly because the effect on expressiveness by the aggregators is less significant than the number of parameters (i.e., number of layers and hidden dimensions).

Different GNN architectures It is also observed that the differences between architectures have a greater impact than simply changing the aggregators. This can be due to other differences, such as message passing framework in PNA compared to convolutions in GCN and GIN, and the added ϵ term in GIN compared to GCN.

Inclusion of edge features It is also observed that when the number of layers and hidden dimensions are larger, the magnitude of the loss difference becomes smaller. This is possibly because when a GNN is more complex, it can capture enough information from graphs even when they contain no edge information, thus the gain in performance by including edge features becomes relatively smaller.

4.4 Correlation with Task Performance

In order to fully validate that the GNNs favored by GraphAC are indeed more expressive, we further evaluate all the GNNs used in the aforementioned experiments against the supervised learning tasks under the `ogbg-molpcba` and `ogbg-code2` datasets (Hu et al., 2020). Each of the supervised training experiments takes 50 epochs. We then perform a correlation study on the GNNs' task performances with their expressiveness rankings produced by GraphAC.

Figure 2 shows the GNNs' task performance on both datasets with respect to their expressiveness rankings produced by GraphAC, measured using the average precision of multi-class classification for the tasks on the `ogbg-molpcba` dataset, and F1 score of sub-token prediction for the tasks on the `ogbg-molpcba` dataset. The plots are divided into the experiment groups as presented in the previous section, in order to accurately demonstrate the correlation. The consistent, monotonic upward trend shows a strong correlation between the GNNs' task performance and GraphAC's expressiveness rankings on them, suggesting that *GraphAC can genuinely distinguish GNNs of different expressiveness across various aspects, favoring the more expressive GNNs*. Separated plots of Figure 2, with one diagram per experiment group and detailed descriptions of the GNN architectures and parameters, can be found in Appendix C.

5 Conclusion

We propose GraphAC (Graph Adversarial Collaboration), a novel, principled, and task-agnostic framework for evaluating GNNs through contrastive self-supervision, without the need of handcrafted augmentations. Inspired by the Barlow Twins loss (Zbontar et al., 2021), we introduce a novel objective function: the Competitive Barlow Twins, which replaces its redundancy reduction term with a difference between the upper-triangle and lower-triangle of the cross-correlation matrix of the two GNN's output embeddings. GraphAC successfully distinguishes GNNs of different expressiveness in all experiments within graphs of two distinct contexts: molecular graphs and abstract syntax trees, across various aspects including the number of layers, hidden dimensionality, aggregators, GNN architecture and edge features, and ensures that more expressive GNNs can always win with a statistically significant difference. GraphAC is also able to estimate the degree of expressiveness of different GNNs, and produce a total ordering of all GNNs with its measurements. GraphAC provides a novel principle of evaluating GNNs and an effective contrastive SSL framework without requiring any augmentations, making a notable contribution to the graph SSL and molecular representation learning community, which can be applied to many important tasks in drug discovery.

We believe that the success of GraphAC opens up a new, principled way of thinking when developing contrastive SSL methods, by considering the more expressive GNN as an encoder that learns more complex but less general information from the graphs, and the less expressive GNN as one that captures more basic but general information. Consequently, combining the two GNNs creates a better overall understanding of the graphs and can be used to perform SSL on graphs without manually applying augmentations, which may have introduced arbitrary human knowledge that were not originally provided by the training data.

Acknowledgements

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service, provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

References

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *10th International Conference on Learning Representations (ICLR 2022)*. OpenReview.net, 2022.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido F Montufar, and Michael Bronstein. Weisfeiler and Lehman go cellular: CW networks. In *Advances in Neural Information Processing Systems (NeurIPS 2021)*, volume 34, pp. 2625–2640. Curran Associates, Inc., 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119, pp. 1597–1607. PMLR, 2020.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pp. 13260–13271. Curran Associates, Inc., 2020.

Enyan Dai, Charu Aggarwal, and Suhang Wang. NRGNN: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 227–236. Association for Computing Machinery, 2021.

Vijay P. Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982v3*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *10th International Conference on Learning Representations (ICLR 2022)*. OpenReview.net, 2022.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70, pp. 1263–1272. PMLR, 2017.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pp. 297–304. PMLR, 2010.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 2020.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pp. 22118–22133. Curran Associates, Inc., 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net, 2017.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems (NeurIPS 2021)*, volume 34, pp. 21618–21629. Curran Associates, Inc., 2021.

Hoang NT, Jun Jin Choong, and Tsuyoshi Murata. Learning graph neural networks with noisy labels. In *ICLR 2019 Workshop on Learning from Limited Labeled Data*, 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 3D Infomax improves GNNs for molecular property prediction. *arXiv preprint arXiv:2110.04126*, 2021.

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net, 2020.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019.

Boris Weisfeiler and Andrei Leman. A reduction of a graph to canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968. English translation available at `https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf`.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019.

Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, volume 139, pp. 11548–11558. PMLR, 2021.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pp. 5812–5823. Curran Associates, Inc., 2020.

Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, volume 139, pp. 12121–12132. PMLR, 2021.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, volume 139, pp. 12310–12320. PMLR, 2021.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net, 2017.

Mengmei Zhang, Linmei Hu, Chuan Shi, and Xiao Wang. Adversarial label-flipping attack and defense for graph neural networks. In *2020 IEEE International Conference on Data Mining (ICDM 2020)*, pp. 791–800, 2020.

A Algorithm

Algorithm 1 PyTorch-style pseudocode for GraphAC's framework

```
# gnn_a, gnn_b: GNN encoder networks
# alpha, beta, lambd, mu: coefficients of the loss terms
# N: batch size
# d: dimensionality of the embeddings
#
# diagonal: on-diagonal elements of a matrix
# off_diagonal: off_diagonal elements of a matrix
# triu: upper-triangle elements of a matrix
# tril: lower-triangle elements of a matrix

for x in dataloader: # load a batch with N samples
    # compute embeddings
    h_a_out = gnn_a(x)   # N x d
    h_b_out = gnn_b(x)   # N x d

    # normalize embeddings along the batch dimension (VICReg)
    h_a_out_norm = (h_a_out - h_a_out.mean(dim=0))  # N x d
    h_b_out_norm = (h_b_out - h_b_out.mean(dim=0))  # N x d

    # covariance matrices
    cov_a = (h_a_out_norm.T @ h_a_out_norm) / (N - 1)  # d x d
    cov_b = (h_b_out_norm.T @ h_b_out_norm) / (N - 1)  # d x d

    # covariance regularisation loss
    cov_loss = off_diagonal(cov_a).pow(2).sum() / d \
             + off_diagonal(cov_b).pow(2).sum() / d

    # normalize embeddings along the batch dimension (Barlow Twins)
    h_a_out_norm = h_a_out_norm / h_a_out.std(dim=0)  # N x d
    h_b_out_norm = h_a_out_norm / h_a_out.std(dim=0)  # N x d

    # cross-correlation matrix
    corr = (h_a_out_norm.T @ h_b_out_norm) / N  # d x d

    # Competitive Barlow Twins loss components
    on_diag = (diagonal(corr) - 1).pow(2).sum()
    upper_tri = triu(corr, diagonal=1).pow(2).sum()
    upper_tri = tril(corr, diagonal=-1).pow(2).sum()

    # Competitive Barlow Twins losses
    loss_a = alpha * (on_diag + lambd * (upper_tri - mu * lower_tri)) \
           + beta * cov_loss
    loss_b = alpha * (on_diag + lambd * (lower_tri - mu * upper_tri)) \
           + beta * cov_loss

    # optimisation steps
    loss_a.backward()
    loss_b.backward()
    optimiser_a.step()
    optimiser_b.step()
    optimiser_a.zero_grad()
    optimiser_b.zero_grad()
```

B Detailed Results

Table 2: Loss differences of the conducted experiments on the `ogbg-molpcba` dataset, with the uncertainty included and with a higher precision.

		#Layers in GNN$_B$ (256 hidden dims, aggregators: [max, mean, sum])				
		2	4	6	8	10
#Layers in GNN$_A$ (256 hidden dims, [max, mean, sum])	2	-0.042 $\pm$ 0.095	1.290 $\pm$ 0.127	1.458 $\pm$ 0.101	1.882 $\pm$ 0.154	1.966 $\pm$ 0.175
	4	-1.315 $\pm$ 0.106	0.027 $\pm$ 0.097	0.799 $\pm$ 0.190	1.399 $\pm$ 0.264	1.748 $\pm$ 0.344
	6	-1.478 $\pm$ 0.218	-0.845 $\pm$ 0.320	-0.014 $\pm$ 0.069	0.674 $\pm$ 0.211	0.808 $\pm$ 0.152
	8	-1.687 $\pm$ 0.284	-1.276 $\pm$ 0.269	-0.406 $\pm$ 0.087	0.030 $\pm$ 0.098	0.555 $\pm$ 0.176
	10	-2.008 $\pm$ 0.245	-1.751 $\pm$ 0.235	-1.076 $\pm$ 0.298	-0.668 $\pm$ 0.160	-0.004 $\pm$ 0.065
		Hidden dims in GNN$_B$ (4 layers, aggregators: [max, mean, sum])				
		16	32	64	128	256
Hidden dims in GNN$_A$ (4 layers, [max, mean, sum])	16	0.022 $\pm$ 0.078	1.390 $\pm$ 0.301	1.883 $\pm$ 0.174	2.356 $\pm$ 0.238	2.554 $\pm$ 0.293
	32	-1.240 $\pm$ 0.215	0.001 $\pm$ 0.083	0.932 $\pm$ 0.238	1.614 $\pm$ 0.287	2.093 $\pm$ 0.301
	64	-2.290 $\pm$ 0.155	-0.895 $\pm$ 0.228	0.018 $\pm$ 0.047	1.206 $\pm$ 0.229	1.813 $\pm$ 0.246
	128	-2.493 $\pm$ 0.262	-1.609 $\pm$ 0.340	-0.997 $\pm$ 0.166	-0.010 $\pm$ 0.084	1.490 $\pm$ 0.143
	256	-2.541 $\pm$ 0.235	-2.041 $\pm$ 0.348	-1.704 $\pm$ 0.234	-1.330 $\pm$ 0.212	-0.006 $\pm$ 0.096

		Aggregators in GNN$_B$ (4 layers, 64 hidden dims)			
		[max]	[mean]	[sum]	[max, mean, sum]
Aggregators in GNN$_A$ (4 layers, 64 hidden dims)	[max]	-0.026 $\pm$ 0.060	0.182 $\pm$ 0.091	0.305 $\pm$ 0.059	0.342 $\pm$ 0.051
	[mean]	-0.174 $\pm$ 0.041	-0.007 $\pm$ 0.040	0.241 $\pm$ 0.068	0.328 $\pm$ 0.069
	[sum]	-0.304 $\pm$ 0.032	-0.248 $\pm$ 0.038	0.047 $\pm$ 0.028	0.228 $\pm$ 0.049
	[max,mean,sum]	-0.329 $\pm$ 0.068	-0.295 $\pm$ 0.047	-0.239 $\pm$ 0.045	0.018 $\pm$ 0.047

		GNN$_B$ architecture (4 layers, 64 hidden dims)					Hidden dims (PNA, edge feat. $-$ no edge feat.)		
		GCN	GIN	PNA			64	128	256
GNN$_A$	GCN	-0.069 $\pm$ 0.073	0.338 $\pm$ 0.074	0.467 $\pm$ 0.117	#Layers	4	-0.418 $\pm$ 0.096	-0.340 $\pm$ 0.118	-0.198 $\pm$ 0.060
	GIN	-0.337 $\pm$ 0.056	-0.026 $\pm$ 0.018	0.441 $\pm$ 0.142		6	-0.453 $\pm$ 0.119	-0.285 $\pm$ 0.085	-0.214 $\pm$ 0.089
	PNA	-0.594 $\pm$ 0.200	-0.419 $\pm$ 0.160	0.018 $\pm$ 0.047		8	-0.425 $\pm$ 0.100	-0.261 $\pm$ 0.099	-0.205 $\pm$ 0.061

Table 3: Loss differences of the conducted experiments on the `ogbg-code2` dataset, with the uncertainty included and with a higher precision. Since this dataset does not contain any edge features, no experiments regarding the inclusion of edge features for this dataset are conducted.

		#Layers in GNN$_B$ (256 hidden dims, aggregators: [max, mean, sum])				
		2	4	6	8	10
#Layers in GNN$_A$ (256 hidden dims, [max, mean, sum])	2	-0.006 $\pm$ 0.024	0.312 $\pm$ 0.189	0.363 $\pm$ 0.115	0.629 $\pm$ 0.285	0.655 $\pm$ 0.152
	4	-0.249 $\pm$ 0.135	0.005 $\pm$ 0.027	0.338 $\pm$ 0.091	0.404 $\pm$ 0.147	0.524 $\pm$ 0.115
	6	-0.357 $\pm$ 0.177	-0.308 $\pm$ 0.137	-0.002 $\pm$ 0.018	0.239 $\pm$ 0.142	0.342 $\pm$ 0.116
	8	-0.594 $\pm$ 0.143	-0.395 $\pm$ 0.111	-0.267 $\pm$ 0.162	0.001 $\pm$ 0.024	0.185 $\pm$ 0.114
	10	-0.670 $\pm$ 0.385	-0.463 $\pm$ 0.133	-0.346 $\pm$ 0.124	-0.183 $\pm$ 0.118	-0.077 $\pm$ 0.023
		Hidden dims in GNN$_B$ (4 layers, aggregators: [max, mean, sum])				
		16	32	64	128	256
Hidden dims in GNN$_A$ (4 layers, [max, mean, sum])	16	0.008 $\pm$ 0.051	0.612 $\pm$ 0.134	0.805 $\pm$ 0.185	0.856 $\pm$ 0.156	0.926 $\pm$ 0.168
	32	-0.660 $\pm$ 0.207	-0.036 $\pm$ 0.046	0.532 $\pm$ 0.195	0.742 $\pm$ 0.195	0.875 $\pm$ 0.290
	64	-0.728 $\pm$ 0.128	-0.659 $\pm$ 0.233	-0.002 $\pm$ 0.043	0.389 $\pm$ 0.085	0.620 $\pm$ 0.155
	128	-0.810 $\pm$ 0.166	-0.716 $\pm$ 0.163	-0.410 $\pm$ 0.115	-0.027 $\pm$ 0.056	0.417 $\pm$ 0.156
	256	-0.914 $\pm$ 0.129	-0.784 $\pm$ 0.141	-0.636 $\pm$ 0.129	-0.396 $\pm$ 0.147	-0.028 $\pm$ 0.068

		Aggregators in GNN$_B$ (4 layers, 64 hidden dims)			
		[max]	[mean]	[sum]	[max, mean, sum]
Aggregators in GNN$_A$ (4 layers, 64 hidden dims)	[max]	-0.016 $\pm$ 0.052	0.144 $\pm$ 0.067	0.300 $\pm$ 0.087	0.335 $\pm$ 0.148
	[mean]	-0.146 $\pm$ 0.071	-0.001 $\pm$ 0.045	0.172 $\pm$ 0.089	0.202 $\pm$ 0.074
	[sum]	-0.296 $\pm$ 0.199	-0.124 $\pm$ 0.048	-0.009 $\pm$ 0.049	0.152 $\pm$ 0.059
	[max,mean,sum]	-0.409 $\pm$ 0.267	-0.283 $\pm$ 0.230	-0.151 $\pm$ 0.110	-0.002 $\pm$ 0.043
		GNN$_B$ architecture (4 layers, 64 hidden dims)			
		GCN	GIN	PNA	
GNN$_A$ architecture (4 layers, 64 hidden dims)	GCN	-0.016 $\pm$ 0.056	1.061 $\pm$ 0.416	1.662 $\pm$ 0.402	
	GIN	-1.192 $\pm$ 0.493	-0.013 $\pm$ 0.062	1.403 $\pm$ 0.221	
	PNA	-1.582 $\pm$ 0.457	-1.427 $\pm$ 0.360	-0.002 $\pm$ 0.043	

C Detailed correlation plots

C.1 ogbg-molpcba Dataset

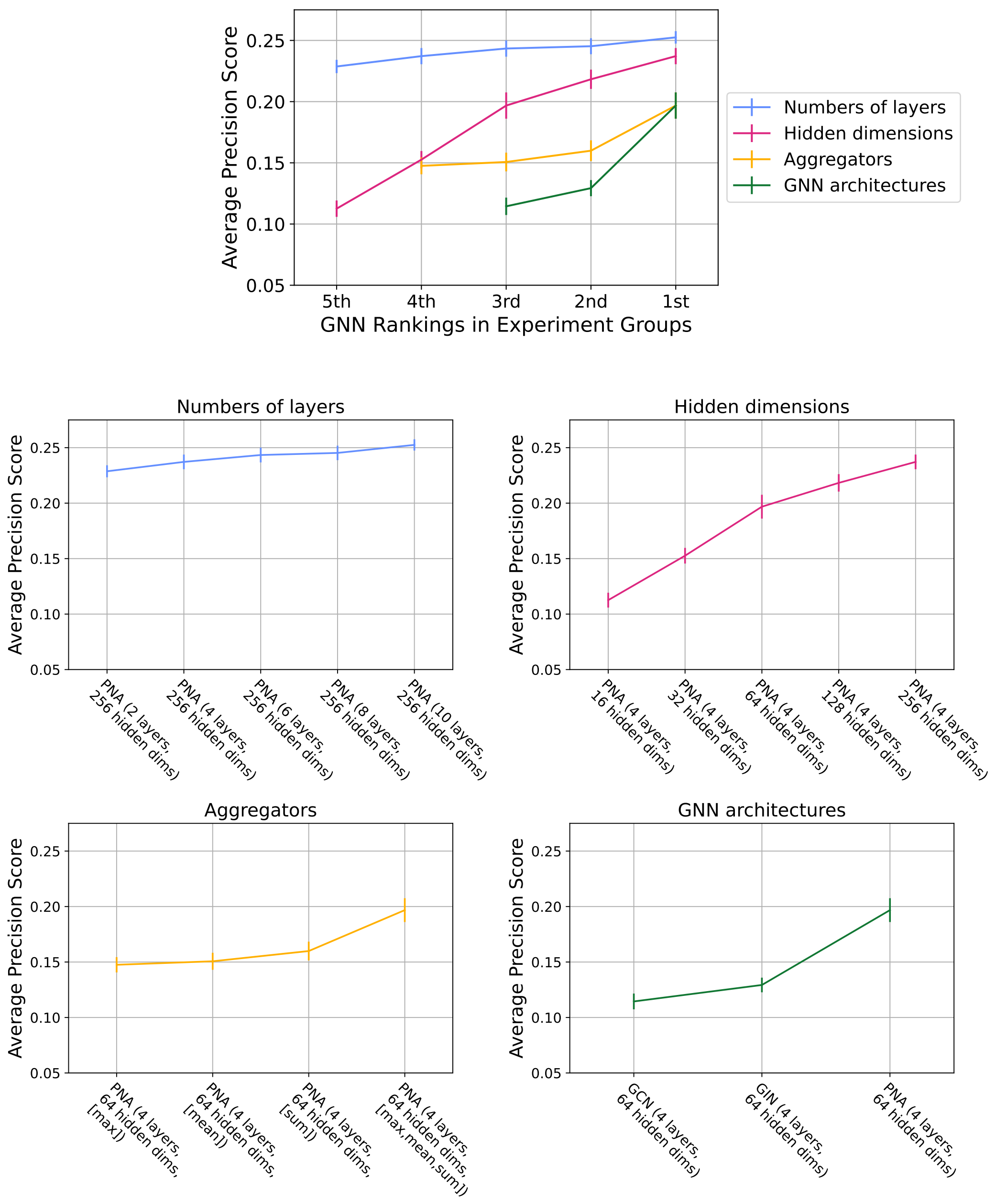


Figure 3: Correlation plots of GraphAC's GNN rankings with the GNNs' task performances on the `ogbg-molpcba` dataset, with detailed descriptions of the GNN architectures and parameters.

C.2 OGBG-CODE2 DATASET

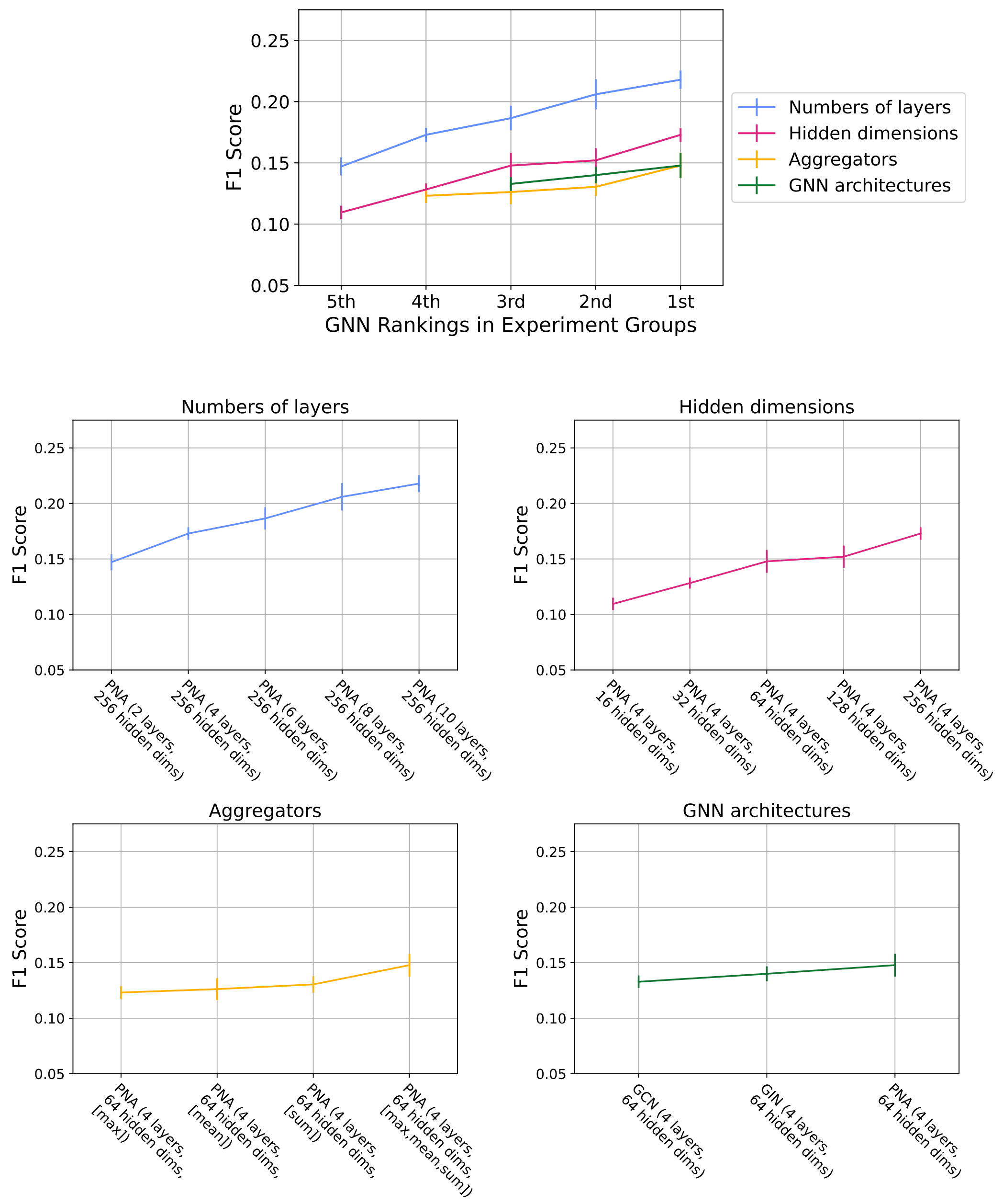


Figure 4: Correlation plots of GraphAC's GNN rankings with the GNNs' task performances on the `ogbg-code2` dataset, with detailed descriptions of the GNN architectures and parameters.

D Training Details

D.1 Hyperparameter Tuning

For all hyperparameter tuning experiments, we use a 10-layer PNA with 256 hidden dimensions, and a 10-layer PNA with 128 hidden dimensions, as the pair of competing GNNs. Both PNAs use [max, mean, sum] as their aggregators, [identity, amplification, attenuation] as their scalers, and their message passing functions are parametrized by 2-layer MLPs. The output dimensionality is set to 256 for all experiments. These settings on the GNNs are only for standardizing hyperparameter tuning, and can be changed in the actual evaluation of GraphAC. Adam Kingma & Ba (2015) was used as the optimizer for all experiments.

Hyperparameter tuning of GraphAC was focused on the batch size of the training data, weighting coefficients of the sums of the two triangles, and the learning rates. Since the trial experiments have shown that the Competitive Barlow Twins and VICReg covariance regularisation terms are in the same order or magnitude, and they share a similar importance in stabilizing training, the Competitive Barlow Twins terms $\mathcal{L}_{\text{CBT}_A}$, $\mathcal{L}_{\text{CBT}_B}$ and VICReg covariance regularisation terms $\mathcal{L}_{\text{Cov}}$ are set to share the same weights (i.e., $\alpha = \beta = 1$ in Equation (5)), and the value of λ in Equation (4) adopts the hyperparameter tuning results in the original Barlow Twins paper (Zbontar et al., 2021), which is 5×10^{-3}. In order to test whether trading-off is required between collaboration (i.e., predicting the other GNNs' graph embeddings) and competition (i.e., preventing the other GNNs from predicting the GNNs' own graph embeddings), the μ in Equation (4) is set to take values from 0.1, 0.2, 0.5, 1, 2, 5 and 10. In order to thoroughly validate the proposed framework for GraphAC and find the optimal hyperparameter settings for it, we conducted hyperparameter tuning experiments in a grid-search manner on each framework. The hyperparameter search space and the final values selected for GraphAC are specified in Table 4:

Table 4: Hyperparameters searched for the competitive Barlow Twins framework. **Bold** values indicate the final selections.

Dataset	Hyperparameter	Search space
`ogbg-molpcba`	Weighting coefficient of the triangle (μ)	[0.1, 0.2, 0.5, **1**, 2, 5, 10]
	Batch size of the training data	[64, 128, 256, **512**, 1024]
	Learning rate	[1×10^{-5}, $\mathbf{5 \times 10^{-5}}$, 2×10^{-4}]
`ogbg-code2`	Batch size of the training data	[64, **128**, 256, 512, 1024]
	Learning rate	[$\mathbf{1 \times 10^{-5}}$, 5×10^{-5}, 2×10^{-4}]

The results show that the GraphAC framework can indeed distinguish the two GNNs with stable training, and can ensure that the more expressive GNN always has a lower loss. The result that $\mu = 1$ is the most and only suitable weighting coefficient of the triangle also agrees to our derivation of Competitive Barlow Twins in Section 3.1. Apart from that, we noted that the training of GraphAC was generally stable and is not sensitive towards the choice of hyperparameters.

D.2 TRAINING OUTCOMES

Figure 5 presents an example of learning curves of the loss difference between two GNNs with the GraphAC framework. This also indicates that the more expressive GNN can continuously achieve a lower loss in our framework. Moreover, the PCA explained variance plot in Figure 6 suggests that GraphAC successfully avoids information collapse. This further confirms the claim made in Section 3.1 that the GNNs' output embeddings are ordered by feature importance, thereby provides a more stable training process.

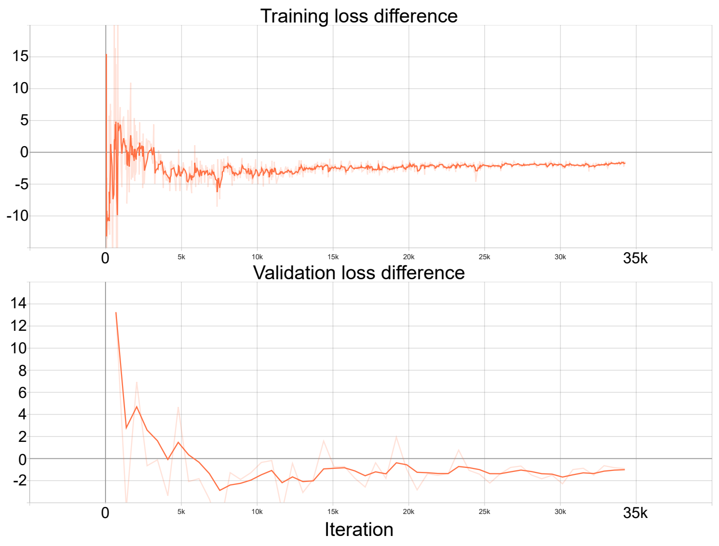


Figure 5: Example learning curves of the loss differences between the stronger and weaker GNNs under GraphAC's framework.

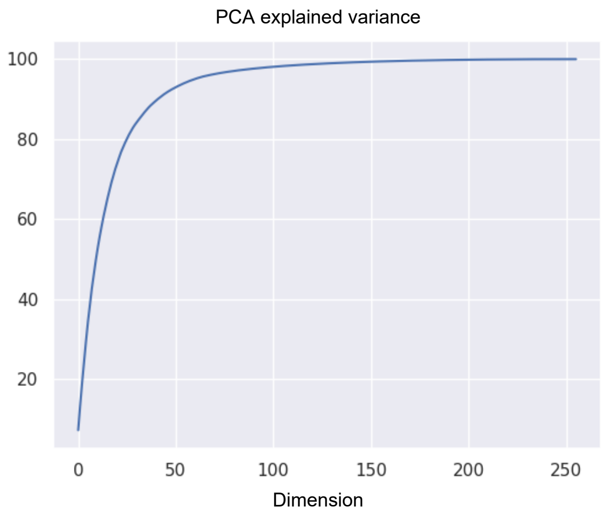


Figure 6: Example PCA explained variance of the stronger GNN's output embeddings under GraphAC's framework.