Bootstrapped Representation Learning on Graphs

Shantanu Thakoor [1] **Corentin Tallec** [1] **Mohammad Gheshlaghi Azar** [1] **Rémi Munos** [1] **Petar Veličković** [1]
Michal Valko [1]

Abstract

Current state-of-the-art self-supervised learning methods for graph neural networks (GNNs) are based on contrastive learning. As such, they heavily depend on the construction of augmentations and negative examples. For example, on the standard PPI benchmark, increasing the number of negative pairs improves performance, thereby requiring computation and memory cost quadratic in the number of nodes to achieve peak performance. Inspired by `BYOL`, a recently introduced method for self-supervised learning that does not require negative pairs, we present Bootstrapped Graph Latents, `BGRL`, a self-supervised graph representation method that gets rid of this potentially quadratic bottleneck. `BGRL` outperforms or matches the previous unsupervised state-of-the-art results on several established benchmark datasets. Moreover, it enables the effective usage of graph attentional (GAT) encoders, allowing us to further improve the state of the art. In particular on the PPI dataset, using GAT as an encoder we achieve state-of-the-art 70.49% Micro-F1, using the linear evaluation protocol. On all other datasets under consideration, our model is competitive with the equivalent supervised GNN results, often exceeding them.

1. Introduction

Self-supervised learning is a promising path towards eliminating the need for costly label information in representation learning on many domains, including images, video, speech and text. This is especially relevant in the graph domain, where unsupervised data is abundant, but label information is scarce. Most of the best performing self-supervised learning methods are *contrastive* (Hjelm et al., 2019; Oord et al., 2018). Specifically, contrastive methods build representations by pulling together representations of related objects and pushing apart representations of the unrelated ones. They have displayed performance that matches or improves over the equivalent methods trained with labeled data (Tian et al., 2020; Bachman et al., 2019; Mitrovic et al., 2021; Caron et al., 2020; Xu et al., 2020).

Inspired by the success of contrastive methods in vision and elsewhere, contrastive learning methods were adapted to graphs (Veličković et al., 2019; Peng et al., 2020; Hassani & Khasahmadi, 2020; Zhu et al., 2020a). The first such method, `DGI` (Veličković et al., 2019), is closely aligned with `Deep InfoMax` (Hjelm et al., 2019). More recently, `GRACE` (Zhu et al., 2020a) adapts the `SimCLR` (Chen et al., 2020a;b) method to graphs and achieves state-of-the-art performance. In particular, `GRACE` learns node representations by creating two augmented versions of a graph, pulling together the representation of the same node in the two graphs, while pushing apart representations of every other node.

However, the practical efficiency of contrastive methods relies on the ability to compare each object to a large number of *negative* examples (He et al., 2020). This is especially prohibitive for large graphs as in the worst case this requires a time and space complexity quadratic in the number of nodes, as we exemplify on the PPI dataset in Figure 2. In general, relying on negative examples seems to be undesirable, particularly for graphs, where negative examples are difficult to define in a principled way.

`BYOL` (Grill et al., 2020) is a self-supervised method, introduced in the vision domain that provides results that are competitive with the best-performing contrastive methods while avoiding the need for negative examples (Richemond et al., 2020). In this paper, we adapt `BYOL` to graphs, and propose *Bootstrapped Graph Latents* (`BGRL`). `BGRL` learns a node representation by encoding two augmented versions of a graph using two distinct graph encoders, an online encoder, and a target one. The online encoder is trained through predicting the representation of the target encoder, while the target encoder is updated as an exponential moving average of the online network. By removing the need to contrast different node representations, `BGRL` relieves self-supervised learning in graphs from its reliance on negative examples.

[1]DeepMind. Correspondence to: Shantanu Thakoor <thakoor@google.com>.

Preprint.

Our main contributions are:

- We introduce BGRL, a self-supervised graph representation learning method that achieves state-of-the-art results on several large and challenging graph datasets with time and space complexity at most *linear* in the number of edges. We further show that BGRL can be used with attentional models, outperforming simpler convolutional models using self-supervised losses. With BGRL, we demonstrate that the BYOL method can be adapted to work *at scale*, on the graph domain.
- We show that, as was the case for other domains, contrastive methods for graphs require large amounts of negatives to work well, and may suffer from subsampling of negative examples. Reducing the number of negative examples in order to decrease the time and memory complexity also deteriorates the performance of some constrative methods, including the state-of-the-art, GRACE. This implies that contrastive methods may require time and space that is quadratic in the number of nodes for peak performance. BGRL avoids these pitfalls by eliminating the need for negative examples altogether.
- We additionally show that longer training time and normalization layers significantly improve self-supervised learning performance across the board, and provide the improved performance on a number of graph datasets. In particular, we are the first to report self-supervised GNN representation results on the ogbn-arXiv dataset.

2. Bootstrapped Graph Latents

To enable self-supervised graph representation learning without the use of a contrastive objective, we adapt BYOL (Grill et al., 2020) to the graph domain and propose **B**ootstrapped **Gr**aph **L**atents (BGRL). As in BYOL, BGRL learns representations by bootstrapping the output of a delayed version of its own encoder, *without having to define any negative examples*. Unlike BYOL, which deals with datasets of independent data points, we follow past graph representation learning methods and leverage *topological structure* inherent in the graph.

2.1. BGRL Components

Formally, BGRL maintains two graph encoders, an online encoder $\mathcal{E}_\theta$ and a target encoder $\mathcal{E}_\phi$, where θ and ϕ denote two distinct sets of parameters.

We consider a graph $\mathbf{G} = (\mathbf{X}, \mathbf{A})$, with *node features* $\mathbf{X} \in \mathbb{R}^{N \times F}$ and *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{N \times N}$. Here N represents the number of nodes in the graph and F the number of features.

BGRL first produces two alternate views of $\mathbf{G}$: $\mathbf{G}_1 = (\widetilde{\mathbf{X}}_1, \widetilde{\mathbf{A}}_1)$ and $\mathbf{G}_2 = (\widetilde{\mathbf{X}}_2, \widetilde{\mathbf{A}}_2)$, by applying stochastic graph augmentation functions $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. The online encoder produces an online representation from the first augmented graph, $\widetilde{\mathbf{H}}_1 \coloneqq \mathcal{E}_\theta(\widetilde{\mathbf{X}}_1, \widetilde{\mathbf{A}}_1)$; similarly the target encoder produces a target representation of the second augmented graph, $\widetilde{\mathbf{H}}_2 \coloneqq \mathcal{E}_\phi(\widetilde{\mathbf{X}}_2, \widetilde{\mathbf{A}}_2)$.

The online representation is fed into a predictor p_θ that outputs a prediction of the target representation, $\widetilde{\mathbf{Z}}_1 \coloneqq p_\theta(\widetilde{\mathbf{H}}_1, \widetilde{\mathbf{A}}_1)$. Unless otherwise specified, the predictor works at the node level, without taking into account graph information (ie. operating over $\widetilde{\mathbf{H}}_1$ only, and not $\widetilde{\mathbf{A}}_1$).

BGRL differs from BYOL (and other methods) in that it *does not use a projection network* to project the representation to a smaller space before predicting it. BYOL relies on this to simplify the task of the predictor p_θ, as it is challenging to predict directly very high-dimensional embeddings (such as are required for learning on large-scale classification tasks like ImageNet, Deng et al., 2009). However, we empirically found that at the scale of commonly available graph datasets, this is not required and in most cases actually slows down learning due to providing a more indirect learning signal.

2.2. BGRL update step

Updating θ The online parameters θ (and not ϕ), are updated to make the predicted target representations $\widetilde{\mathbf{Z}}_1$ closer to the true target representations $\widetilde{\mathbf{H}}_2$ for each node, by following the gradient of the cosine similarity

$$\ell(\theta, \phi) = -\frac{2}{N} \sum_{i=0}^{N-1} \frac{\widetilde{\mathbf{Z}}_{(1,i)} \widetilde{\mathbf{H}}_{(2,i)}^\top}{\|\widetilde{\mathbf{Z}}_{(1,i)}\| \|\widetilde{\mathbf{H}}_{(2,i)}\|} \tag{1}$$

w.r.t. θ, i.e.,

$$\theta \leftarrow \text{optimize}(\theta,\ \eta,\ \partial_\theta \ell(\theta, \phi)) \tag{2}$$

where η is the learning rate and the final updates are computed from the gradients of the objective with respect to θ *only*, using an optimization method such as SGD or Adam (Kingma & Ba, 2015). In practice, we symmetrize the training, by also predicting the target representation of the first view using the online representation of the second.

Updating ϕ The target parameters ϕ are updated as an exponential moving average of the online parameters θ, i.e.,

$$\phi \leftarrow \tau\phi + (1 - \tau)\theta, \tag{3}$$

where τ is a decay rate controlling how close ϕ remains to θ. Figure 1 visually summarizes BGRL's architecture.

Although the objective $\ell(\theta, \phi)$ has undesirable or trivial solutions, BGRL does not optimize this loss. Only the online

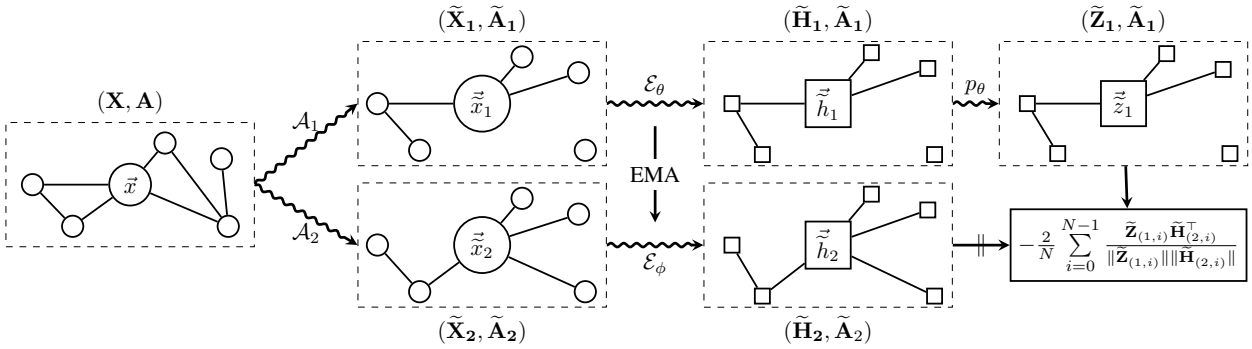

Figure 1. Overview of our proposed BGRL method. The original graph is first used to derive two different semantically similar views using augmentations $\mathcal{A}_{1,2}$. From these, we use encoders $\mathcal{E}_{\theta,\phi}$ to form online and target node embeddings. The predictor p_θ uses the online embedding $\widetilde{\mathbf{H}}_1$ to form a prediction $\widetilde{\mathbf{Z}}_1$ of the target embedding $\widetilde{\mathbf{H}}_2$. The final objective is then computed as the cosine similarity between $\widetilde{\mathbf{Z}}_1$ and $\widetilde{\mathbf{H}}_2$, flowing gradients only through $\widetilde{\mathbf{Z}}_1$. The target parameters ϕ are updated as an exponentially moving average of θ.

parameters are updated to reduce this loss, while the target parameters follow a different objective. Empirically, similarly to BYOL, BGRL does not collapse to trivial solutions, and $\ell(\theta, \phi)$ does not converge to 0 (see Appendix B).

2.3. Fully non-contrastive objective

Here we note that a contrastive approach would instead encourage $\widetilde{\mathbf{Z}}_{(1,i)}$ and $\widetilde{\mathbf{H}}_{(2,j)}$ to be far apart for node pairs (i, j) that are dissimilar. However, choosing such dissimilar node pairs requires domain knowledge and may not be easy to define. In the absence of a principled way of choosing negative examples, the naïve approach of simply contrasting all pairs $\{(i, j) \mid i \neq j\}$ (as done by GRACE and GCA), quickly runs into memory issues on realistically-sized graphs, and sampling negatives randomly worsens performance (see Section 3.3).

No reliance on negative examples As a result of being fully non-contrastive, BGRL does not require any negative examples. Representations are directly learned by predicting the representation of each node in one view of the graph, using the representation of the same node in another view. BGRL's computational and memory complexity scale linearly with the number of edges, as opposed to naive contrastive methods, which scale quadratically with the number of nodes. Further, contrary to hard-mining contrastive methods, BGRL does not require making arbitrary choices on which nodes to select as negative examples for each node.

2.4. Graph augmentation functions

In this work, we consider the standard graph augmentation pipeline that has been used in previous works on representation learning (You et al., 2020; Zhu et al., 2020a). We use the term "augmentation" as opposed to "corruption" as has been used before (Veličković et al., 2019), as our intention is to produce two views which are semantically similar. This differs from, e.g. DGI, where *semantically dissimilar* views are constructed and used to contrast against the original one.

We consider two simple graph augmentation functions — **node feature masking** and **edge masking**. These augmentations are graph-wise: they do not operate on each node independently, and leverage graph topology information through edge masking. This contrasts with transformations used in BYOL, which operate on each image independently.

First, we generate a single random binary mask of size F, each element of which follows a Bernoulli distribution $\mathcal{B}(1 - p_f)$, and use it to mask the features of all nodes in the graph (i.e., all nodes have the same features masked).

In addition to this node-level attribute transformation, we then also compute a binary mask of size E (where E is the number of edges in the original graph), each element of which follows a Bernoulli distribution $\mathcal{B}(1 - p_e)$, and use it to mask edges in the augmented graph.

To compute our final augmented graphs, we make use of both augmentation functions with different hyperparameters for each graph, i.e. p_{f_1} and p_{e_1} for the first view, and p_{f_2} and p_{e_2} for the second view.

Some prior works have also investigated *adaptive* augmentations (Zhu et al., 2020b; Che et al., 2020), using heuristics such as node centrality or PageRank centrality (Page et al., 1999) to mask different edges with different probabilities. This improves the quality of the augmented graphs by helping these transformations preserve semantic similarity (e.g. by making it less likely to mask critical edges that connect otherwise disjoint parts of the graph). We consider only simple, standard augmentations in order to isolate and study the effect of BGRL as a representation learning method, as it is known that stronger augmentations can have a large impact on the quality of representations learned (Grill et al., 2020). However, as we show in Section 3.1.3, our method is competitive with baselines that use adaptive augmentations.

3. Experiments

We demonstrate the effectiveness of the BGRL technique on a set of 7 transductive and inductive tasks, using BGRL to generate node embeddings and evaluating their quality based on the performance of a linear model trained on top.

Unsupervised training In all our experiments, we use the AdamW optimizer (Kingma & Ba, 2015; Gugger & Howard, 2018) with weight decay set to 10^{-5}, and all models initialized using Glorot initialization (Glorot & Bengio, 2010). The BGRL predictor p_θ used to predict the embedding of nodes across views is fixed to be a Multilayer Perceptron (MLP) with a single hidden layer, unless otherwise specified. The decay rate τ controlling the rate of updates to the BGRL target parameters ϕ is initialized to 0.99 and gradually increased to 1.0 over the course of training following a cosine schedule. Our unsupervised training models and process are implemented in JAX (Babuschkin et al., 2020). Other model architecture and training details vary per dataset and are described further below. Information about the dataset are summarized in Table 1. The augmentation hyperparameters $p_{f_{1,2}}$ and $p_{e_{1,2}}$ closely follow those previously used in prior works and are reported in Appendix A.

Evaluation of embeddings We follow the standard linear evaluation protocol introduced by Veličković et al. (2019). We first train each model in a fully unsupervised manner and compute embeddings for each node. A simple linear model is then trained on top of these frozen embeddings through a logistic regression loss with ℓ_2 regularization, without flowing any gradients back to the graph encoder network. We implement this linear model using the Scikit-Learn (Pedregosa et al., 2011) logistic regression implementation, with a small hyperparameter search over the regularization strength described in Appendix A. On larger datasets where this does not converge in reasonable time, we learn the linear model through 100 steps of optimization using AdamW.

3.1. Experiments on standard transductive tasks

We first evaluate our method on a set of 5 recent real-world datasets (Mernyei & Cangea, 2020; Shchur et al., 2018) proposed for evaluating semi-supervised node classification — WikiCS,[1] Amazon-Computers, Amazon-Photos, Coauthor-CS, Coauthor-Physics[2] — in the transductive setting and show that our proposed method is competitive with or exceeds prior unpervised baselines as well as supervised learning methods.

Note that on transductive tasks, the training process has access to *all* nodes, but only the training set labels. Thus, unsupervised methods that provide a training signal at every node can often provide a much richer signal than a fully supervised setup which only provides training signals to nodes in the train set. In datasets where the number of training nodes is a small fraction of all nodes, unsupervised methods can make better use of all the data available and often outperform fully supervised baselines.

3.1.1. Datasets

WikiCS This graph is constructed from Wikipedia references, with nodes representing articles about Computer Science and edges representing links between them. Articles are classified into 10 classes based on their subfield, and node features are the average of GloVE (Pennington et al., 2014) embeddings of all words in the article. This dataset comes with 20 canonical train/valid/test splits, which we use directly.

Amazon Computers, Amazon Photos These graphs are from the Amazon co-purchase graph (McAuley et al., 2015) with nodes representing products and edges being between pairs of goods frequently purchased together. Products are classified into 10 (for Computers) and 8 (for Photos) classes based on product category, and node features are a bag-of-words representation of a product's reviews. We use a random split of the nodes into (10/10/80%) train/validation/test nodes respectively as these datasets do not come with a standard dataset split.

Coauthor CS, Coauthor Physics These graphs are from the Microsoft Academic Graph (Sinha et al., 2015), with nodes representing authors and edges between authors who have co-authored a paper. Authors are classified into 15 (for CS) and 5 (for Physics) classes based on the author's research field, and node features are a bag-of-words representation of the keywords of an author's papers. We again use a random (10/10/80%) split for these datasets.

3.1.2. Models

We use a GCN model (Kipf & Welling, 2017) as our graph encoder $\mathcal{E}$ in these experiments. Formally, the GCN propagation rule for a single layer is as follows,

$$\text{GCN}_i(\mathbf{X}, \mathbf{A}) = \sigma\left(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}_i\right), \tag{4}$$

where $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops, $\widehat{\mathbf{D}}$ is the degree matrix, σ is a non-linearity such as ReLU, and $\mathbf{W}_i$ is a learned weight matrix for the i'th layer. We use a 2-layer GCN model in our experiments. The exact model architectures and graph augmentations used for each dataset are described in Appendix A.

[1] https://github.com/pmernyei/wiki-cs-dataset/raw/master/dataset

[2] https://github.com/shchur/gnn-benchmark/tree/master/data/npz

	Task	Nodes	Edges	Features	Classes
WikiCS	Transductive	11,701	216,123	300	10
Amazon Computers	Transductive	13,752	245,861	767	10
Amazon Photos	Transductive	7,650	119,081	745	8
Coauthor CS	Transductive	18,333	81,894	6,805	15
Coauthor Physics	Transductive	34,493	247,962	8,415	5
ogbn-arxiv	Transductive	169,343	1,166,243	128	40
PPI (24 graphs)	Inductive	56,944	818,716	50	121 (multilabel)

Table 1. Statistics of datasets used in our experiments.

In our experiments, we closely follow models and architectures as used in prior works (Zhu et al., 2020b; Veličković et al., 2019; Zhu et al., 2020a) for fair comparisons. For all BGRL experiments, we anneal the learning rate using a cosine decay (Loshchilov & Hutter, 2017), and use batch normalization (Ioffe & Szegedy, 2015) between layers. We also experimented with these additions on the DGI and GRACE baselines, but they did not make a significant difference .

3.1.3. Results

In our experiments, we primarily compare BGRL against GRACE (Zhu et al., 2020a), the current state-of-the-art *contrastive* representation learning approach that relies on negative samples. Where available, we also report performances from previously published results (Zhu et al., 2020b) for DeepWalk (Perozzi et al., 2014), DGI (Veličković et al., 2019), GMI (Peng et al., 2020), MVGRL (Hassani & Khasahmadi, 2020), GCA (Zhu et al., 2020b) (noting that GRACE is equivalent to the GCA-T-A ablation studied in Zhu et al. (2020b)), and a supervised-learning baseline. In all our experiments, we use identical encoder architectures for BGRL and all baselines. We also use identical graph augmentation hyperparameters as Zhu et al. (2020b) for fair comparison.

We run BGRL for 10,000 epochs on all datasets. For fair comparison, we also report results of running GRACE for 10,000 epochs. Finally, we report the performance of Random-Init (Veličković et al., 2019), a randomly initialized encoder with an identical architecture, showing that this simple baseline can lead to very strong embeddings.

In Table 2, we report results of our experiments on 5 transductive node classification tasks. Notice that the Random-Init model is very strong, in particular outperforming all previously reported baselines on the WikiCS dataset. Further, we find that the performance of GRACE improves over previously published results with longer training on 3 of our considered datasets.

BGRL performs competitively both with our unsupervised and fully supervised baselines using identical architectures.

3.2. Experiments on a larger transductive task

We further run experiments on a node classification task on a much larger dataset from the OGB benchmark (Hu et al., 2020a), and show that our method is still competitive with supervised learning methods using similar architectures.

3.2.1. Dataset

ogbn-arXiv This is another citation network, where nodes represent CS papers on arXiv indexed by the Microsoft Academic Graph (Sinha et al., 2015). In our experiments, we symmetrize this graph and thus there is an edge between any pair of nodes if one paper has cited the other. Papers are classified into 40 classes based on arXiv subject area. The node features are computed as the average word-embedding of all words in the paper, where the embeddings are computed using a skip-gram model (Mikolov et al., 2013) over the entire corpus.

3.2.2. Models

We use the same GCN model described in Section 3.1.2. To account for the increased complexity of this task, we slightly expand our model to use 3 GCN layers of dimension 256, following the baseline model provided in Hu et al. (2020a). For this task, we found the use of layer normalization (Ba et al., 2016) and weight standardization (Qiao et al., 2019) to be more effective than batch normalization at stabilizing training, following an alternative stabilizing technique used in Richemond et al. (2020). We have applied these to our baseline models in these experiments as well for fair comparison. Further details on model structure and augmentation hyperparameters can be found in Appendix A.

3.2.3. Results

As there has not been prior work done on applying GNN-based unsupervised approaches to the ogbn-arXiv task, we implement and compare against two representative contrastive learning approaches, DGI and GRACE. In addition, we report results from Hu et al. (2020a) for node2vec (Grover & Leskovec, 2016) and a supervised learning baseline. For GRACE, since the standard approach

	WikiCS	Am. Computers	Am. Photos	CoauthorCS	CoauthorPhy
Raw features	71.98 ± 0.00	73.81 ± 0.00	78.53 ± 0.00	90.37 ± 0.00	93.58 ± 0.00
DeepWalk	74.35 ± 0.06	85.68 ± 0.06	89.44 ± 0.11	84.61 ± 0.22	91.77 ± 0.15
DeepWalk + features	77.21 ± 0.03	86.28 ± 0.07	90.05 ± 0.08	87.70 ± 0.04	94.90 ± 0.09
DGI	75.35 ± 0.14	83.95 ± 0.47	91.61 ± 0.22	92.15 ± 0.63	94.51 ± 0.52
GMI	74.85 ± 0.08	82.21 ± 0.31	90.68 ± 0.17	OOM	OOM
MVGRL	77.52 ± 0.08	87.52 ± 0.11	91.74 ± 0.07	92.11 ± 0.12	95.33 ± 0.03
GRACE	78.19 ± 0.01	87.46 ± 0.22	92.15 ± 0.24	92.93 ± 0.01	95.26 ± 0.02
Random-Init*	78.95 ± 0.58	86.46 ± 0.38	92.08 ± 0.48	91.64 ± 0.29	93.71 ± 0.29
GRACE (10,000 epochs)*	**80.14 ± 0.48**	**89.53 ± 0.35**	**92.78 ± 0.45**	91.12 ± 0.20	OOM
BGRL*	**79.36 ± 0.53**	**89.68 ± 0.31**	**92.87 ± 0.27**	**93.21 ± 0.18**	**95.56 ± 0.12**
GCA (stronger augmentations)	*78.35 ± 0.05*	*88.94 ± 0.15*	*92.53 ± 0.16*	*93.10 ± 0.01*	*95.73 ± 0.03*
Supervised GCN	*77.19 ± 0.12*	*86.51 ± 0.54*	*92.42 ± 0.22*	*93.03 ± 0.31*	*95.65 ± 0.16*

Table 2. Result on transductive tasks. Performance measured in terms of classification accuracy along with standard deviations. Our experiments, marked as $\star$, are over 20 random dataset splits and model initializations. The other results are taken from previously published reports. OOM indicates running out of memory on a 16GB V100 GPU. We report the best result for GCA out of the proposed GCA-DE, GCA-PR, and GCA-EV models.

of taking negative examples from the entire graph consumes $\mathcal{O}(N^2)$ memory and is impractical for a dataset of this size, we provide results on a version where we subsample k nodes to use as negative examples for each node per gradient step. Note that we experimented with different numbers of negative examples and found monotonic increase in validation accuracy, stopping at 2048 due to memory constraints. Thus, GRACE may be negatively impacted by the need to sample negative examples instead of using the full graph. Section 3.3 has a more detailed study of this effect on a dataset where full-graph sampling is possible for comparison.

We report results on both validation and test sets, since the dataset is split based on a chronological ordering, and thus the validation and test sets also measure progressively harder challenges for out-of-distribution generalization. Our results, summarized in Table 3, show that BGRL is competitive with the supervised learning baseline and state-of-the-art contrastive unsupervised baselines.

3.3. Experiments on inductive task with multiple graphs

Finally, we examine the PPI[3] inductive task, and show that our proposed changes to the training process improve the performance of both BGRL and GRACE to set a new state-of-the-art. Further, as there is still a significant gap with supervised learning on this task and it is known that Graph Attention Networks (GAT) (Veličković et al., 2018) perform much better than non-attentional models on this dataset, we show that BGRL can be used to stably train GAT models and further improve performance from the previous state-of-the-art of 66.2% micro-F1 to 70.49% micro-F1.

[3] https://s3.us-east-2.amazonaws.com/dgl.ai/dataset/ppi.zip

3.3.1. Datasets

PPI is a protein-protein interaction network (Zitnik & Leskovec, 2017; Hamilton et al., 2017), comprised of multiple (24) graphs each corresponding to different human tissues. We use the standard dataset split as 20 graphs for training, 2 for validation, and 2 for testing. Each node has 50 features computed from various biological properties. This is a multilabel classification task, where each node can possess up to 121 labels.

3.3.2. Models

Here we use a simple mean-pooling propagation rule from the GraphSAGE-GCN model (Hamilton et al., 2017):

$$\mathrm{MP}_i(\mathbf{X}, \mathbf{A}) = \sigma(\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}\mathbf{X}\mathbf{W}_i) \tag{5}$$

As proposed by Veličković et al. (2019), our exact encoder $\mathcal{E}$ is a 3-layer mean-pooling network with skip connections. We use a layer size of 512 and PReLU (He et al., 2015) activation.

We also consider GAT models where each node aggregates features from its neighbors non-uniformly using a learned attention weight. The GAT layer consists of a learned matrix $\mathbf{W}$ that transforms each node features. We then use self-attention to compute attention coefficient for a pair of nodes i and j as $e_{ij} = a(\mathbf{h}_i, \mathbf{h}_j)$. The attention function a is computed as $\mathrm{LeakyReLU}(\mathbf{a}[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j])$, where $\mathbf{a}$ is a learned matrix transforming a pair of concatenated attention queries into a single scalar attention logit. The weight of the edge between nodes i and j is computed as $\alpha_{ij} = \mathrm{softmax}_j(e_{ij})$.

We follow the architecture proposed by Veličković et al. (2018), including a 3-layer GAT model (with the first 2

	Validation	Test
MLP	57.65± 0.12	55.50 ± 0.23
node2vec	71.29 ± 0.13	70.07 ± 0.13
Random-Init⋆	69.90 ± 0.11	68.94 ± 0.15
DGI⋆	71.26 ± 0.11	70.34 ± 0.16
GRACE full-graph⋆	OOM	OOM
GRACE ($k = 2$)⋆	60.49 ± 3.72	60.24 ± 4.06
GRACE ($k = 8$)⋆	71.30 ± 0.17	70.33 ± 0.18
GRACE ($k = 32$)⋆	72.18 ± 0.16	71.18 ± 0.16
GRACE ($k = 2048$)⋆	**72.61 ± 0.15**	**71.51 ± 0.11**
BGRL⋆	**72.53 ± 0.09**	**71.64 ± 0.12**
Supervised GCN	*73.00 ± 0.17*	*71.74 ± 0.29*

Table 3. Performance on the ogbn-arXiv task measured in terms of classification accuracy along with standard deviations. Our experiments, marked as ⋆, are averaged over 20 random model initializations. Other results are taken from previously published reports. OOM indicates running out of memory on a 16GB V100 GPU.

layers consisting of 4 heads of size 256 each and the final layer size 512 with 6 output heads), ELU activation (Clevert et al., 2016), and skip-connections in intermediate layers.

As in Section 3.2, we train BYOL for $10,000$ epochs and use layer normalization and learning rate annealing with a cosine schedule to increase stability. We also find that the performance of GRACE can also be greatly improved with these stabilizing changes when run for $10,000$ epochs. When using the larger GAT encoder, due to memory constraints we train with a batch size of 1 graph and thus ran experiments longer, for $20,000$ update steps.

3.3.3. Results

We report our results in Table 4. BGRL and GRACE both significantly beat the previously published state-of-the-art, while maintaining stable training for thousands of epochs.

Effect of subsampling on GRACE The high performance of GRACE is due in part to its ability to provide a rich learning signal through its contrastive loss that considers every pair of nodes in the graph. In Figure 2 we more closely analyze the effect of subsampling fewer negative examples per gradient step, to study how this contrastive loss behaves in the absence of being able to provide a rich enough signal. In our experiments, we sample k random nodes to use as negative examples, and measure how performance is affected by varying k from $\{1, 8, 32, 128, 1024, 4096\}$. Note that since the average number of nodes in each graph is 2372, the higher values of k we consider are very close to a quadratic computation. We see that performance has a direct relation with the number of negative examples sampled. BGRL is competitive with GRACE in these tasks, making it a promising method to apply to larger graphs where subsampling is necessary to fit memory constraints.

	PPI
Raw features	42.20
DGI	63.80 ± 0.20
GMI	65.00 ± 0.02
GRACE	66.20 ± 0.10
Random-Init	62.60 ± 0.20
GRACE (10,000 epochs)⋆	69.66 ± 0.15
BGRL MLP predictor⋆	68.90 ± 0.21
BGRL GCN predictor⋆	69.55 ± 0.21
GRACE GAT encoder⋆	69.71 ± 0.17
BGRL ConstGAT encoder⋆	67.67 ± 0.22
BGRL GAT encoder⋆	**70.49 ± 0.05**
Supervised MeanPooling	*96.90 ± 0.20*
Supervised GAT	*97.30 ± 0.20*

Table 4. Performance on the PPI task measured in terms of Micro-F_1 across the 121 labels along with standard deviations. Our experiments, marked as ⋆, are averaged over 20 random model initializations. Other results are taken from previously published reports.

GNN-based BGRL predictor Next, we examine ways to improve the performance of BGRL by strengthening the predictor p_θ used in the BGRL loss. Unlike the case of applying BYOL to images, where the predictor must learn to predict the projection of each image in the minibatch independently, here our graph augmentations are applied to all the nodes in the graph in a coherent way. Thus, it is intuitive that a model that makes use of all the node embeddings in one view, would be better able to predict the embeddings of nodes in the second view. Thus, we perform an experiment comparing either an MLP to predict the embedding of each node independently, or a GNN model that can make use of all the node embeddings at once. Our results are summarized in Table 4, where we see that the use of this more powerful graph-based predictor further increases BGRL's performance to more or less match that of GRACE. In order to ensure the improved performance with a GNN encoder is due to better use of graph information in the BGRL predictor and not simply due to increase in model parameters, we also experimented with wider and deeper MLP predictors but found that this did not affect performance significantly.

GAT encoder models In a separate attempt to improve BGRL, we consider using the more powerful attention based model described in Section 3.3.2 as our encoder $\mathcal{E}$. GAT models are known to achieve better results on the PPI tasks than GCN encoders in supervised learning, but have thus far not been able to be trained to a higher performance than GCN models through unsupervised techniques.

Table 4 also reports the results of using a GAT encoder with either GRACE or BGRL, showing that BGRL can more effectively train the more complex GAT model to achieve

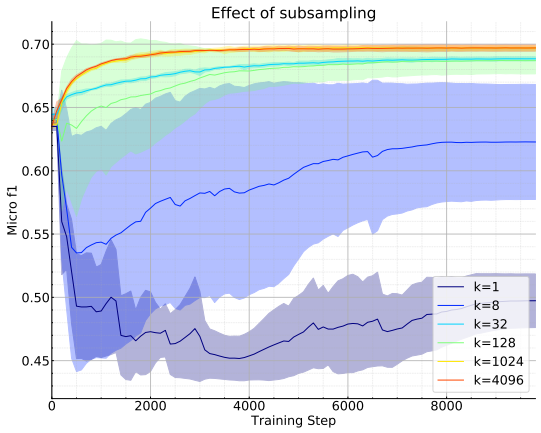

Figure 2. Studying the effect of subsampling k nodes as negative examples for GRACE loss on PPI dataset, averaged over 5 random model initializations.

a new state-of-the-art on this dataset. To ensure that this higher performance is not only due to the bigger architecture we used, we also report the performance of a ConstGAT model (i.e., a GAT model using constant attention weights) trained with BGRL, showing that the non-contrastive loss is able to provide enough signal to allow nodes to aggregate over their neighbors in a non-uniform way. Interestingly, the more complex GRACE contrastive loss is unable to improve performance of a GAT model over the standard, smaller MeanPooling encoder. The *all-vs-all* loss provided by GRACE is less guided than the *targeted* bootstrapping objective of BGRL, meaning that it is also less suited to guiding the (often brittle) attentional coefficients. This aligns with recent observations that carefully chosen auxiliary losses are often paramount for stability of GAT models (Kim & Oh, 2021; Wang et al., 2019). We provide further analyses of unsupervised training of GAT models in Appendix C.

4. Related Work

Prior to using graph neural networks (GNNs) as graph encoders, dominant methods in the area relied on *random-walk objectives* such as DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016). Even though GNNs have an inductive bias that aligns with these objectives, composing GNNs and random-walks does not work very well – randomly-initialised GNNs (Veličković et al., 2019; Kipf & Welling, 2017) and nonparametric GNNs (Wu et al., 2019) are both competitive with DeepWalk, with no training necessary. Moreover, training GNN encoders with such loss functions (as done by Hamilton et al., 2017) can even *degrade* performance relatively to an untrained encoder.

Earlier combinations of GNNs and self-supervised learning involve Embedding Propagation (García-Durán & Niepert, 2017), Variational Graph Autoencoders (Kipf & Welling, 2016) and Graph2Gauss (Bojchevski & Günnemann, 2018). While all of these are effectively variations of a random-walk style objective, they are enhanced by relevant additions that are still applicable for current methods, such as edge-wise corruption or explicitly incorporating embedding uncertainty. Yet another but tangential direction for training encoders for representation is re-using inspiration from BERT-style (Devlin et al., 2019) losses in graph-structured inputs, as leveraged by Hu et al. (2020b). In particular, the strategies of Hu et al. (2020b) assume that the input graph is attributed in a way that would make feature masking objectives viable.

Recently, contrastive methods effective on images have also been adapted to graphs using GNNs. This includes DGI (Veličković et al., 2019), inspired by Deep InfoMax (Hjelm et al., 2019), which contrasts node-local patches against global graph representations. Next, InfoGraph (Sun et al., 2020) provided modifications to DGI's pipeline to make the global embedding useful for graph classification tasks. DGI was also generalized to spatiotemporal graphs by ST-DGI (Opolka et al., 2019), and multiplex networks by DMGI (Park et al., 2020). GMI (Peng et al., 2020) directly maximizes a notion of *graphical* mutual information inspired by MINE (Belghazi et al., 2018), allowing for a more fine-grained contrastive loss than DGI's. Furthermore, the SimCLR method of Chen et al. (2020a) has been specialized for graphs by GRACE and variants such as GCA (Zhu et al., 2020a;b). GraphCL (You et al., 2020) adapts GRACE to learn graph-level embeddings using a contrastive objective. Additionally, MVGRL method (Hassani & Khasahmadi, 2020) generalizes CMC (Tian et al., 2020) to graphs. GRACE and MVGRL have emerged as state-of-the-art methods, contrasting nodes across various graph views. Finally, concurrently to our work, Che et al. (2020) have explored the possibility of using bootstrapping for self-supervised learning in graphs. However, they only consider small citations datasets with fixed train/test splits, known to be saturated and unreliable to evaluate GNN methods (Shchur et al., 2018).

5. Conclusion

We have introduced BGRL, a new method for self-supervised graph representation learning. Through a wide range of experiments, we have shown that our method is competitive with state-of-the-art approaches, in spite of not requiring negative examples and substantially reducing storage requirements due to not relying on a projection network or quadratic node comparisons. Moreover, our approach can be naturally extended to learn graph-level embeddings, where defining negative examples is challenging, and an all-vs-all objective does not scale.

References

Ba, J., Kiros, J., and Hinton, G. E. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Fantacci, C., Godwin, J., Jones, C., Hennigan, T., Hessel, M., Kapturowski, S., Keck, T., Kemaev, I., King, M., Martens, L., Mikulik, V., Norman, T., Quan, J., Papamakarios, G., Ring, R., Ruiz, F., Sanchez, A., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stokowiec, W., and Viola, F. The DeepMind JAX Ecosystem, 2020.

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *Neural Information Processing Systems*, 2019.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. Mutual information neural estimation. In *International Conference on Machine Learning*, 2018.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. In *Neural Information Processing Systems*, 2020.

Che, F., Yang, G., Zhang, D., Tao, J., Shao, P., and Liu, T. Self-supervised graph representation learning via bootstrapping. *arXiv preprint arXiv:2011.05126*, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020a.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. Big self-supervised models are strong semi-supervised learners. In *Neural Information Processing Systems*, 2020b.

Clevert, D., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

García-Durán, A. and Niepert, M. Learning graph representations with embedding propagation. In *Neural Information Processing Systems*, 2017.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Conference on Artificial Intelligence and Statistics*, 2010.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. In *Neural Information Processing Systems*, 2020.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, 2016.

Gugger, S. and Howard, J. Adamw and super-convergence is now the fastest way to train neural nets. https://www.fast.ai/2018/07/02/adam-weight-decay/, 2018.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, 2017.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision*, USA, 2015.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Conference on Computer Vision and Pattern Recognition*, 2020.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *Neural Information Processing Systems*, 2020a.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Kim, D. and Oh, A. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. Image-based recommendations on styles and substitutes. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 2015.

Mernyei, P. and Cangea, C. Wiki-cs: A wikipedia-based benchmark for graph neural networks, 2020.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

Mitrovic, J., McWilliams, B., Walker, J. C., Buesing, L. H., and Blundell, C. Representation learning via invariant causal mechanisms. In *International Conference on Learning Representations*, 2021.

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

Opolka, F. L., Solomon, A., Cangea, C., Veličković, P., Liò, P., and Hjelm, R. D. Spatio-temporal deep graph infomax. In *RLGM Workshop, ICLR*, 2019.

Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

Park, C., Kim, D., Han, J., and Yu, H. Unsupervised attributed multiplex network embedding. In *Conference on Artificial Intelligence, AAAI*, 2020.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., and Huang, J. *Graph Representation Learning via Graphical Mutual Information Maximization*, pp. 259–270. Association for Computing Machinery, New York, NY, USA, 2020.

Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk. *ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, 2014.

Qiao, S., Wang, H., Liu, C., Shen, W., and Yuille, A. L. Weight standardization. *CoRR*, abs/1903.10520, 2019.

Richemond, P. H., Grill, J.-B., Altché, F., Tallec, C., Strub, F., Brock, A., Smith, S., De, S., Pascanu, R., Piot, B., and Valko, M. BYOL works even without batch statistics. In *NeurIPS 2020 Workshop on Self-Supervised Learning: Theory and Practice*, 2020.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.-J. P., and Wang, K. An overview of microsoft academic service (mas) and applications. In *International Conference on World Wide Web*, 2015.

Sun, F., Hoffmann, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. In *ECCV*, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *International Conference on Learning Representations*, 2019.

Wang, G., Ying, R., Huang, J., and Leskovec, J. Improving graph attention networks with large margin-based constraints. *ArXiv*, abs/1910.11945, 2019.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, 2019.

Xu, H., Zhang, X., Li, H., Xie, L., Xiong, H., and Tian, Q. Hierarchical semantic aggregation for contrastive representation learning. *arXiv preprint arXiv:2012.02733*, 2020.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. In *Neural Information Processing Systems*, 2020.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *ArXiv*, abs/2006.04131, 2020a.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Graph contrastive learning with adaptive augmentation. *ArXiv*, abs/2010.14945, 2020b.

Zitnik, M. and Leskovec, J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33 (14):i190–i198, Jul 2017.

A. Implementation Details

Model architectures As described in Section 3, we use GCN (Kipf & Welling, 2017) encoders in our experiments on the transductive tasks, while on the inductive task of PPI we use MeanPooling encoders with residual connections. The BGRL predictor p_θ is implemented as a mutilayer perceptron (MLP). We also used stabilization techniques like batch normalization (Ioffe & Szegedy, 2015), layer normalization (Ba et al., 2016), and weight standardization (Qiao et al., 2019). The decay rate use for statistics in the batch normalization is fixed to 0.99. We use PReLU activation (He et al., 2015) in all experiments except those using a GAT encoder, where we use the ELU activation (Clevert et al., 2016). In all our models, at each layer including the final layer, we apply first the batch/layer normalization as applicable, and then the activation function. Table 5 describes hyperparameter and architectural details for most of our experimental setups with BGRL.

In addition to these standard settings, we perform two additional experiments on the PPI dataset—using a GNN-based predictor p_θ, and using a GAT (Veličković et al., 2018) model as the encoder.

When experimenting on PPI with graph-based predictors, we define p_θ to be an MLP with one hidden layer of size 512, plus a stacked MeanPooling model with the same structure as the encoder $\mathcal{E}$. We found this combination of the node-based MLP and graph-based GNN to provide the best results.

When using the GAT encoder on PPI, we use 3 attention layers — the first two with 4 attention heads of size 256 each, and the final with 6 attention heads of size 512, following a very similar model proposed by Veličković et al. (2018). We concatenate the attention head outputs for the first 2 layers, and use the mean for the final output. We also use the ELU activation (Clevert et al., 2016), and skip connections in the intermediate attention layers, as suggested by Veličković et al. (2018).

	WikiCS	Am. Computers	Am. Photos	Co. CS	Co. Physics	ogbn-arXiv	PPI
$p_{f,1}$	0.2	0.2	0.1	0.3	0.1	0.0	0.25
$p_{f,2}$	0.1	0.1	0.2	0.4	0.4	0.0	0.00
$p_{e,1}$	0.2	0.5	0.4	0.3	0.4	0.6	0.30
$p_{e,2}$	0.3	0.4	0.1	0.2	0.1	0.6	0.25
η_{base}	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	10^{-4}	10^{-5}	10^{-5}	10^{-2}	$5 \cdot 10^{-3}$
embedding size	256	128	256	256	128	256	512
$\mathcal{E}$ hidden sizes	512	256	512	512	256	256, 256	512, 512
p_θ hidden sizes	512	512	512	512	512	256	512
batch norm	Y	Y	Y	Y	Y	N	N
layer norm	N	N	N	N	N	Y	Y
weight standard.	N	N	N	N	N	Y	N

Table 5. Hyperparameter settings for unsupervised BGRL learning.

Augmentation parameters The hyperparameter settings for graph augmentations, as well as the sizes of the embeddings and hidden layers, very closely follow previous work (Zhu et al., 2020a;b) on all datasets with the exception of ogbn-arXiv. On this dataset, since there has not been prior work on applying self-supervised graph learning methods, we provide the hyperparameters we found through a small grid search.

Optimization settings We perform full-graph training at each gradient step on all experiments, with the exception of experiments using GAT encoders on the PPI dataset. Here, due to memory constraints, we perform training with a batch size of 1 graph. Since the PPI dataset consists of multiple smaller, disjoint subgraphs, we do not have to perform any node subsampling at training time.

All experiments use Glorot initialization (Glorot & Bengio, 2010) the AdamW optimizer (Kingma & Ba, 2015; Gugger & Howard, 2018) with a base learning rate η_{base} and weight decay set to 10^{-5}. The learning rate is annealed using a cosine schedule over the course of learning of n_{total} total steps with an initial warmup period of n_{warmup} steps. Hence, the learning rate at step i is computed as

$$\eta_i \triangleq \begin{cases} \frac{i \times \eta_{\text{base}}}{n_{\text{warmup}}} & \text{if } i \leq n_{\text{warmup}}, \\ \eta_{\text{base}} \times \left(1 + \cos \frac{(i - n_{\text{warmup}}) \times \pi}{n_{\text{total}} - n_{\text{warmup}}}\right) \times 0.5 & \text{if } n_{\text{warmup}} \leq i \leq n_{\text{total}}. \end{cases}$$

We fix n_{total} to be 10,000 total steps and n_{warmup} to 1,000 warmup steps, with the exception of experiments on the GAT encoder that requires using a batch size of 1 graph on the PPI dataset. In this case, we increase the number of total steps to 20,000 and warmup to 2,000 steps.

The target network parameters ϕ are initialized randomly from the same distribution of the online parameters θ but with a different random seed. The decay parameter τ is also updated using a cosine schedule starting from an initial value of $\tau_{\text{base}} = 0.99$ and is computed as

$$\tau_i \triangleq 1 - \frac{(1 - \tau_{\text{base}})}{2} \times \left(\cos\left(\frac{i \times \pi}{n_{\text{total}}}\right) + 1\right).$$

These annealing schedules for both η and τ follow the procedure used by Grill et al. (2020).

Evaluation of embeddings The final evaluation is done by fitting a linear classifier on top of the frozen learned embeddings without flowing any gradients back to the encoder. For the smaller datasets of WikiCS, Amazon Computers/Photos, and Coauthor CS/Physics, we use an ℓ_2-regularized LogisticRegression classifier from Scikit-Learn (Pedregosa et al., 2011) using the 'liblinear' solver. We do a hyperparameter search over the regularization strength to be between $\{2^{-10}, 2^{-9}, \ldots 2^{9}, 2^{10}\}$.

For larger PPI and ogbn-arXiv datasets, where the liblinear solver takes too long to converge, we instead perform 100 steps of gradient descent using AdamW with learning rate 0.01, with a smaller hyperparameter search on the weight decay between $\{2^{-10}, 2^{-8}, 2^{-6}, \ldots 2^{6}, 2^{8}, 2^{10}\}$.

B. BGRL does not converge to trivial solutions

In Figure 3 we show the BGRL loss curve throughout training for all the datasets considered. As we see, the loss does not converge to zero, indicating that the training does not result in a trivial solution.

In Figure 4 we plot the spread of the node embeddings, i.e., the standard deviation of the representations learned across all nodes, divided by the average norm. As we see, the embeddings learned across all datasets have a standard deviation that is a similar order of magnitude as the norms of the embeddings themselves, further indicating that the training dynamics do not converge to a constant solution.

Further, Figure 5 shows that the embeddings do not collapse to zero or blow up as training progresses.

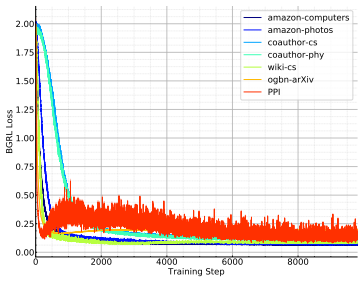

Figure 3. BGRL Loss

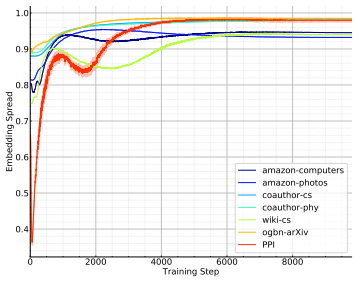

Figure 4. Embedding spread

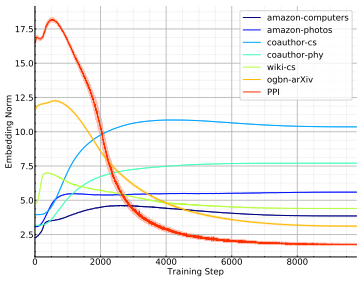

Figure 5. Average embedding norm

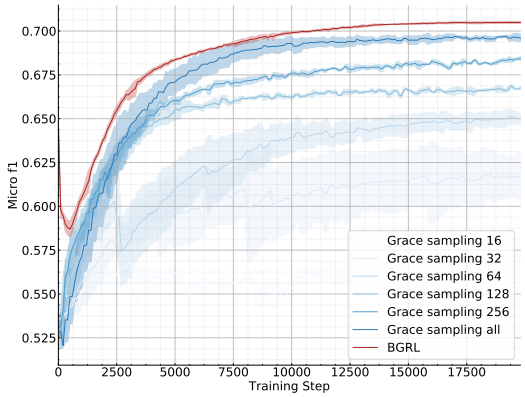

Figure 6. GAT model performance

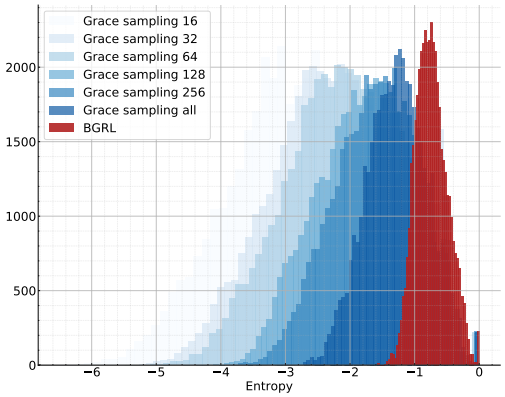

Figure 7. Histogram of GAT attention entropies

C. Analyzing learned graph attention models

In this section, we analyze the training of GAT encoders on the PPI dataset more closely. For this purpose, we additionally evaluate the effect of subsampling negative examples when training with GRACE. All other experimental settings are unchanged.

Figure 6 shows that, similar to the MeanPooling models, GAT models are negatively affected by subsampling of negative examples in the GRACE contrastive loss. We see a steady increase in performance when more negative examples are used, with the peak reached by the version that does not do any subsampling.

Next, we study the entropy of the attention weights learned. Specifically, for each node in the PPI training graphs, we compute the average entropy of its attention weights across all GAT layers and attention heads. Since this entropy depends on the number of neighbors this node has, we subtract as a baseline the entropy of a uniform distribution at the node (i.e., taking the ConstGAT model as a reference point for studying the GAT model).

In Figure 7, we observe that there is a significant correlation between model performance and GAT attention entropies. We see that GAT models learned using GRACE, particularly when subsampling few negative examples, tend to have very low attention entropies. This provides further evidence that training GAT models can be difficult. On the other hand, BGRL is able to train the model to have meaningful attention weights, striking a balance between the low-entropy models learned through GRACE and the maximum-entropy ConstGAT model, thus achieving the best performance.