BOOTSTRAPPED REPRESENTATION LEARNING ON GRAPHS

Shantanu Thakoor
DeepMind
thakoor@google.com

Corentin Tallec
DeepMind
corentint@google.com

Mohammad Gheshlaghi Azar
DeepMind
mazar@google.com

Rémi Munos
DeepMind
munos@google.com

Petar Veličković
DeepMind
petarv@google.com

Michal Valko
DeepMind
valkom@google.com

ABSTRACT

Current state-of-the-art self-supervised learning methods for graph neural networks are based on contrastive learning. As such, they heavily depend on the construction of augmentations and negative examples. Increasing the number of negative pairs improves performance, thereby requiring quadratic computation and memory cost 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 benchmarks. Moreover, it enables the effective usage of graph attentional (GAT) encoders, allowing us to further improve the state of the art, in particular achieving 70.49% Micro-F1 on the PPI dataset using the linear evaluation protocol.

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 unrelated pairs. They have displayed performance that matches or improves over 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., 2020b). 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., 2020b) adapts the SimCLR (Chen et al., 2020a;b) method to graphs and achieves state-of-the-art performance. GRACE learns node representations by creating two augmented versions of a graph, pulling together representations of the same node in the two graphs, and pushing apart representations of every other node. Appendix A contains further discussion of self-supervised learning methods on graphs.

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 this requires a worst-case time and space complexity quadratic in the number of nodes. In general, relying on negative examples seems undesirable, particularly for graphs, where negative examples are difficult to define in a principled way.

Introduced in the vision domain, BYOL (Grill et al., 2020) is a method competitive with the best contrastive approaches while avoiding the need for negative examples (Richemond et al., 2020). We adapt BYOL to graphs, and propose *Bootstrapped Graph Latents* (BGRL). BGRL maintains two

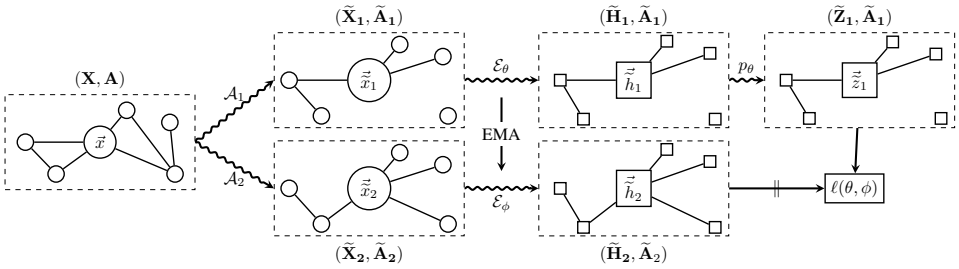

Figure 1: Overview of our proposed BGRL method.

distinct graph encoders, and learns a node representation by training an online encoder to predict the representation of a target one. By removing the need to contrast different node representations, BGRL relieves self-supervised learning in graphs from its reliance on negative examples.

2 BOOTSTRAPPED GRAPH LATENTS

2.1 BGRL COMPONENTS

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

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. We consider simple augmentations used previously (You et al., 2020; Zhu et al., 2020b), stochastic **node feature masking** and **edge masking**. These augmentations are graph-level: they do not operate on each node independently, and leverage graph topology information. Further details are available in Appendix B.

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 using graph information (ie. operating over $\widetilde{\mathbf{H}}_1$ only, and not $\widetilde{\mathbf{A}}_1$). Note that a contrastive approach would instead encourage $\widetilde{\mathbf{H}}_{(1,i)}$ and $\widetilde{\mathbf{H}}_{(2,j)}$ to be far apart for node pairs (i, j) that are dissimilar. However, 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), scales quadratically causing memory issues, and sampling negatives randomly worsens performance. BGRL's computation scales linearly and does not require arbitrary choices on sampling nodes.

2.2 BGRL UPDATE STEP

The online parameters θ (and not ϕ), are updated to make the predictions $\widetilde{\mathbf{Z}}_1$ closer to the true targets $\widetilde{\mathbf{H}}_2$ for each node, following the gradient of the cosine similarity w.r.t. θ, i.e.,

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

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

where η is the learning rate and the final updates are computed from the gradients of the objective with respect to θ *only*. In practice, we symmetrize the training, by also predicting the target representation of the first view using the online representation of the second. The target parameters ϕ are updated as an exponential moving average of the online parameters θ, i.e.

$$\phi \leftarrow \tau \phi + (1 - \tau) \theta$$

	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` (ours)*	**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 1: Performance measured as classification accuracy with standard deviations. Our experiments, marked as $\star$, are over 20 random dataset splits and model initializations. Other results are from previous reports. OOM indicates out-of-memory on a 16GB V100 GPU.

where τ is a decay rate controlling how close ϕ remains to θ.

Note that although the objective $\ell(\theta, \phi)$ has undesirable or trivial solutions, the `BGRL` update as a whole does not optimize this loss. Only the online parameters θ are updated to reduce this loss, while the target parameters ϕ 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 F).

2.3 Discussion of computational costs

The `BGRL` step takes time and space only linear in the size of the input graph, as opposed to contrastive methods such as `GRACE`, which are quadratic. Consider a graph with N nodes and M edges, and simple graph models that compute embeddings in time and space $\mathcal{O}(N + M)$. `BGRL` does 4 encoder computations per update step (2 for target/online encoders, and 2 for each augmentation) plus a prediction step; `GRACE` does 2 (one for each augmentation), plus a projection step. Both methods have the same cost for computing the augmentations, which we ignore in this comparison. Both methods backpropagate the learning signal 2 times (once for each augmentation), and we assume the backward pass to be approximately as costly as a forward pass. Thus the total time complexity per update step for `BGRL` is $6C_{\text{encoder}}(M + N) + 4C_{\text{prediction}}N + C_{\text{BGRL}}N$, and for `GRACE` is $4C_{\text{encoder}}(M+N)+4C_{\text{projection}}N+C_{\text{GRACE}}N^2$, where $C_\cdot$ are constants depending on architecture of the different components. A similar analysis applies to the memory complexities.

We further provide an empirical analysis and comparison of runtime and memory requirements in Section 3.3.

3 Experiments

We evaluate representation learning methods following the standard linear evaluation protocol (Veličković et al., 2019): we first learn node representations in a fully unsupervised manner, and a linear model is then trained on top of these frozen embeddings without flowing any gradients back to the graph encoder network. Our training and evaluation process is implemented in JAX (Babuschkin et al., 2020) and Scikit-Learn (Pedregosa et al., 2011) . Details about the datasets used, our GNN models, and training and evaluation setup are provided in Appendix C, D, and E respectively.

3.1 Experiments on transductive tasks

WikiCS, Amazon Computers/Photos, Coauthor CS/Physics We first evaluate our method on a set of 5 recent real-world datasets (Mernyei & Cangea, 2020; Shchur et al., 2018) for evaluating node classification in the transductive setting. In our experiments, we primarily compare `BGRL` against `GRACE` (Zhu et al., 2020b), the current state-of-the-art *contrastive* representation learning approach that relies on negative samples. For fair comparison, we also report results of running `GRACE` using our training setup, improving over previously published results on 3 datasets. Where available, we also report performances of other methods from previously published results. 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 particular outperforming all previously reported baselines on the WikiCS dataset. Table 1 shows

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

	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 2: Performance on the ogbn-arXiv task as classification accuracy with standard deviations. Our experiments, marked as $\star$, are averaged over 20 random model initializations. Other results are from previous reports. OOM indicates out-of-memory on a 16GB V100 GPU.

ogbn-arXiv We consider a much larger dataset from the OGB benchmark (Hu et al., 2020a), and show in Table 2 that BGRL still performs well. We implement and compare against two representative contrastive learning approaches, DGI and GRACE. We also report results for node2vec (Grover & Leskovec, 2016) and a supervised learning baseline. For GRACE, since taking negative examples from the entire graph consumes $\mathcal{O}(N^2)$ memory and is impractical for a dataset of this size, we subsample k nodes as negative examples per gradient step, seeing that GRACE requires high k to be competitive.

3.2 Experiments on an inductive task with multiple graphs

Finally, we examine the PPI (Protein-Protein Interaction) inductive task, and show in Table 3 that our proposed changes to the training process improve the performance of both BGRL and GRACE to set a new state-of-the-art while maintaining training stability for thousands of epochs.

Effect of subsampling on GRACE The high performance of GRACE is due in part to providing a rich contrastive learning signal that considers every pair of nodes in a 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 to 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. GRACE requires high k to be competitive, making BGRL promising to apply to larger graphs with memory constraints.

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 compare using an MLP to predict each target independently, or a GNN that uses all the node embeddings at once. Table 3 shows this more powerful graph-based predictor further increases BGRL's performance.

GAT encoder models GAT models are known to outperform MeanPooling encoders on PPI task in supervised learning, but have thus far not been trained to a similarly higher performance through unsupervised techniques. Table 3 shows that BGRL can more effectively train the more complex GAT model to achieve a new state-of-the-art on this dataset. We also report the performance of a ConstGAT model (i.e. 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

	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 (ours)⋆	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⋆	$\mathbf{70.49 \pm 0.05}$
Supervised MeanPooling	$\mathit{96.90 \pm 0.20}$
Supervised GAT	$\mathit{97.30 \pm 0.20}$

Table 3: 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.

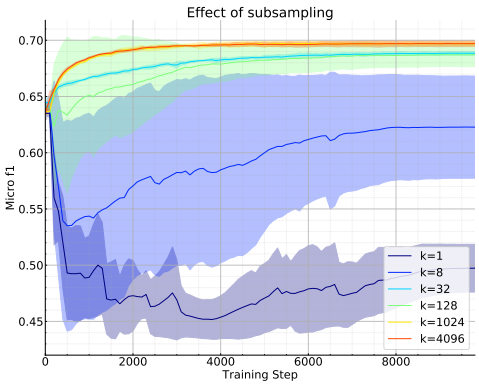

Figure 2: Studying the effect of subsampling k nodes as negative examples for GRACE loss on PPI dataset, averaged over 5 random model initializations.

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 G.

3.3 Empirical Computation Cost Analysis

We provide empirical comparisons of BGRL and GRACE on the datasets from Section 3.1 in Table 4.

As noted in Section 2.3, theoretical analysis of the BGRL update step shows a linear dependence on graph size, whereas GRACE is quadratic. In practice, runtime is affected by factors such as different operations parallelizing, or implementations trading off memory requirements for higher runtime costs. We show that an advantage of BGRL is reducing memory requirements sufficiently to allow scaling to larger datasets easily. This is particularly significant for graph-based applications in practice, where the bottleneck is often memory rather than speed.

	Amazon Photos	WikiCS	Amazon Computers	Coauthor CS	Coauthor Physics
BGRL steps/second	3.04	3.24	3.64	1.47	0.73
GRACE steps/second	3.24	3.28	3.86	1.07	OOM on 16GB GPU
BGRL Memory	**0.47 GB**	**0.63 GB**	**0.58 GB**	**2.86 GB**	**5.50 GB**
GRACE Memory	1.81 GB	3.82 GB	5.14 GB	11.78 GB	OOM on 16GB GPU

Table 4: Comparison of computational requirements.

4 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 computational requirements. 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

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

Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Tom Hennigan, Matteo Hessel, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Lena Martens, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/ddf354219aac374f1d40b7e760ee5bb7-Abstract.html.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, 2018. URL http://proceedings.mlr.press/v80/belghazi18a.html.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1ZdKJ-0W.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Neural Information Processing Systems*, 2020.

Feihu Che, Guohua Yang, Dawei Zhang, Jianhua Tao, Pengpeng Shao, and Tong Liu. Self-supervised graph representation learning via bootstrapping. *arXiv preprint arXiv:2011.05126*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020a. URL http://proceedings.mlr.press/v119/chen20j.html.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *Neural Information Processing Systems*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/fcbc95ccdd551da181207c0c1400c655-Abstract.html.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1511.07289.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Alberto García-Durán and Mathias Niepert. Learning graph representations with embedding propagation. In *Neural Information Processing Systems*, 2017. ISBN 9781510860964.

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

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Neural Information Processing Systems*, 2020.

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

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

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, 2020. URL http://proceedings.mlr.press/v119/hassani20a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision*, USA, 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Conference on Computer Vision and Pattern Recognition*, 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 *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bklr3j0cKX.

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 *Neural Information Processing Systems*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=HJlWWJSFDH.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Wi5KUNlqWty.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6980.

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

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 2015. ISBN 9781450336215. doi: 10.1145/2766462.2767755. URL https://doi.org/10.1145/2766462.2767755.

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

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013. URL http://arxiv.org/abs/1301.3781.

Jovana Mitrovic, Brian McWilliams, Jacob C Walker, Lars Holger Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=9p2ekP904Rs.

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

Felix L Opolka, Aaron Solomon, Cătălina Cangea, Petar Veličković, Pietro Liò, and R Devon Hjelm. Spatio-temporal deep graph infomax. In *RLGM Workshop, ICLR*, 2019.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. Unsupervised attributed multiplex network embedding. In *Conference on Artificial Intelligence, AAAI*, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5985.

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

Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. *Graph Representation Learning via Graphical Mutual Information Maximization*, pp. 259–270. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450370233. URL https://doi.org/10.1145/3366423.3380112.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, 2014. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk. *ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, 2014. doi: 10.1145/2623330.2623732. URL http://dx.doi.org/10.1145/2623330.2623732.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan L. Yuille. Weight standardization. *CoRR*, abs/1903.10520, 2019. URL http://arxiv.org/abs/1903.10520.

Pierre H. Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. BYOL works even without batch statistics. In *NeurIPS 2020 Workshop on Self-Supervised Learning: Theory and Practice*, 2020. URL http://arxiv.org/abs/2010.10241.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *International Conference on World Wide Web*, 2015. ISBN 9781450334730. doi: 10.1145/2740908.2742839. URL https://doi.org/10.1145/2740908.2742839.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lfF2NYvH.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020. doi: 10.1007/978-3-030-58621-8_45. URL https://doi.org/10.1007/978-3-030-58621-8_45.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rklz9iAcKQ.

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

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, 2019.

Haohang Xu, Xiaopeng Zhang, Hao Li, Lingxi Xie, Hongkai Xiong, and Qi Tian. Hierarchical semantic aggregation for contrastive representation learning. *arXiv preprint arXiv:2012.02733*, 2020.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe230348e9a12c13120749e3f9fa4cd-Abstract.html.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, S. Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. *ArXiv*, abs/2010.14945, 2020a.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *ArXiv*, abs/2006.04131, 2020b.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, Jul 2017. ISSN 1460-2059. doi: 10.1093/bioinformatics/btx252. URL http://dx.doi.org/10.1093/bioinformatics/btx252.

A 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., 2020b;a). 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).

B Graph Augmentations

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., 2020b). 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. The exact values of these hyperparameters follow very closely those reported in prior works and are summarized in Table 6.

Some prior works have also investigated *adaptive* augmentations (Zhu et al., 2020a; 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 Table 1, our method is competitive with baselines that use adaptive augmentations.

C Dataset Details

	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 5: Statistics of datasets used in our experiments.

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.

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.

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.

D GNN MODELS

GCN for transductive experiments 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,

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

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.

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

$$\text{MP}_i(\mathbf{X}, \mathbf{A}) = \sigma(\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}\mathbf{X}\mathbf{W}_i) \tag{2}$$

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. Thus, we compute:

$$\mathbf{H}_1 = \sigma(\text{MP}_1(\mathbf{X}, \mathbf{A})) \tag{3}$$
$$\mathbf{H}_2 = \sigma(\text{MP}_2(\mathbf{H}_1 + \mathbf{X}\mathbf{W}_{skip}, \mathbf{A})) \tag{4}$$
$$\mathcal{E}(\mathbf{X}, \mathbf{A}) = \sigma(\text{MP}_3(\mathbf{H}_2 + \mathbf{H}_1 + \mathbf{X}\mathbf{W}_{skip'}, \mathbf{A})) \tag{5}$$

Graph Attention Networks We also consider GAT (Veličković et al., 2018) 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 $\text{LeakyReLU}(\mathbf{a}[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j])$, 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 $\alpha_{ij} = \text{softmax}_j(e_{ij})$.

E IMPLEMENTATION DETAILS

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), as suggested in prior works (Grill et al., 2020; Richemond et al., 2020). 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 6 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 6: Hyperparameter settings for unsupervised BGRL learning.

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 graph 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 $\{2^{-10}, 2^{-8}, 2^{-6}, \ldots 2^{6}, 2^{8}, 2^{10}\}$.

F 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.

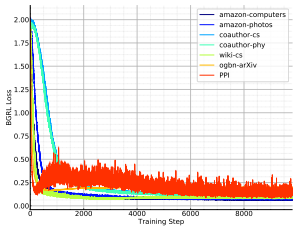

Figure 3: BGRL Loss

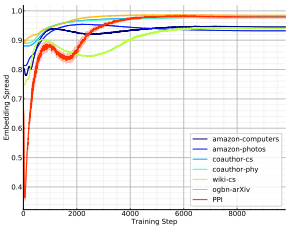

Figure 4: Embedding spread

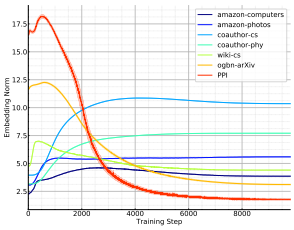

Figure 5: Average embedding norm

G ANALYZING LEARNED GRAPH ATTENTION MODELS

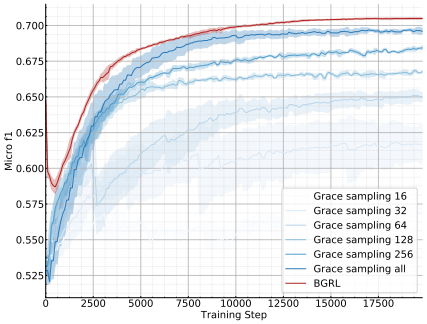

Figure 6: GAT model performance

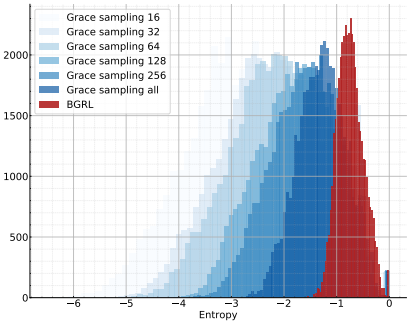

Figure 7: Histogram of GAT attention entropies

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.