One-Stage Supervised Contrastive Learning on Graphs

Linqing Mo, Jose Moreira, Qingyuan Wu
Michigan State University

April 28, 2024

Abstract

Graph neural networks (GNNs) have become increasingly popular and effective for analyzing graph-structured data. To address the challenges associated with the reliance on extensive labeled data for achieving high-quality results, deep graph contrastive learning (GCL) has gained prominence. Most GCL methods typically involve a two-stage process: first, using GNNs to transform graph data into embeddings, and then training another model to map these embeddings to actual labels. This dual-stage approach requires additional parameter tuning, which which demands excessive effort across multiple models. In this work, we utilize a one-stage supervised contrastive learning framework that simplifies this pipeline. We conduct experiments with seven different GNNs models on a real-world dataset and combine the results through an ensemble approach. This framework demonstrates great potential for real-world applications, streamlining the process while maintaining robust performance.

Contents

1 Introduction

In this project, we address the challenge of node classification within a graph-based dataset, with a primary focus on accurately predicting the categories of nodes. Our task is particularly challenging due to the limited availability of known labels. To navigate this limitation, we explore the application of deep graph contrastive learning (GCL) techniques.

Our approach involves experimenting with two prominent methods: GRACE[1] and SupCon[2], both of which are applied to our dataset to enhance model performance under the constraint of limited labels. While these methods initially yield satisfactory results within their conventional two-stage frameworks, we encounter significant challenges in conducting hyperparameter tuning across multiple models due to the complex interactions between two stages.

To streamline this process and improve model efficacy, we decompose the model's loss function into two components: supervised loss and contrastive loss. By incorporating the contrastive loss as a regularization term, we aim to leverage its potential to enhance generalization by pulling close the positive pairs while pushing away the negative pairs . This modification leads to enhanced performance in most of our models, outperforming their counterparts that do not utilize the contrastive loss term.

Overall, by refining our approach to incorporate contrastive learning directly into the loss function, we not only simplify the training process but also achieve more robust performance across our GNN models, effectively addressing the challenges posed by limited label availability.

2 Data Description

Each node in the graph is associated with a feature vector and a label (class). It's important to note that while the adjacency matrix provides the graph structure, classification can still be performed without it, potentially sub-optimally.

Parameter	**Value**
#Nodes	2480
#Edges	10100
#Classes	7
#Features	1390
Train/Test Split	496/1984

Table 1: Dataset Statistics

The given graph was specified through the following files:

- An adjacency matrix: adj.npz

  An adjacency matrix represents connections between nodes in the graph. Each entry A_{ij} indicates whether there is an edge between node i and node j. All edges are not weighted.

- A feature matrix: features.npy

  The feature matrix contains feature vectors associated with each node, representing node characteristics.

- A list of labels: labels.npy

  Labels assign a class label to each node, useful for the classification task.

- Data splits (train/test splits): splits.json

  Data splits divide the dataset into training and testing subsets to evaluate model performance.

3 Method

3.1 GRACE

In our work, the initial approach involves the GRACE framework, as outlined by Zhu et al. [1]. Our specific task involves creating two augmented views of a graph, denoted as $\widetilde{G}_1$ and $\widetilde{G}_2$. These views are generated using a series of augmentation techniques based on the results of preliminary tests: randomly remove 10% of the edges, mask 10% of node features, implement a dropout on 10% of the features, and exclude 20% of the nodes.

For the encoding process, we utilize a three-layer Graph Convolutional Network (GCN)[3], and for projecting, we employ a two-layer multilayer perceptron (MLP). Through these methods, we produce the node embeddings U and V. The contrastive loss objective J is then computed using the Information Noise Contrastive Estimation (InfoNCE)[4], setting the temperature parameter τ at 0.2. We optimize the parameters of the encoder by maximizing the InfoNCE.

It's important to note that within the GRACE framework, negative-pair nodes are defined as all other nodes across two views, resulting in negative samples automatically originating from inter-view or intra-view nodes. This method does not depend on provided labels. However, in this study, where a small portion of labels is available, it proves advantageous to utilize these labels to construct positive and negative pairs. These pairs can then be incorporated into the contrastive loss to enhance the model's performance.

3.2 SupCon

To maximize the benefits from the 20% of known labels in our dataset, we adopt the SupCon approach, which is detailed in [2], to enhance our model's performance. This framework is closely aligned with GRACE in terms of augmentation techniques, encoder structure, projector design, and loss objectives. However, SupCon introduces an additional supervision for semi-supervised learning through the creation of extra positive and negative masks based on the available labels.

- **Extra Positive Mask Construction**: This mask is defined by a matrix M^+ where an entry M^+_{ij} is set to `True` if nodes i and j in the training set share the same label, implying that their embeddings should be drawn closer. Conversely, M^+_{ij} is `False` if the labels differ.
- **Extra Negative Mask Construction**: This matrix, M^+, identifies pairs of nodes with differing labels. Entries for these pairs are marked as `True`, meaning that during training, the model will work to push their embeddings apart, enhancing the discriminative power of the model.

Both masks are duplicated to accommodate inter-view and intra-view comparisons. During the training process, these extra masks can help driving the model to minimize distances between embeddings of nodes marked as positive in the extra positive mask, and to maximize distances for those marked as negative in the extra negative mask.

3.3 One-Stage Supervised Contrastive Learning

GRACE and SupCon have shown slightly improved performance compared to using GCN alone on the validation set, with GRACE achieving an accuracy of 86%, SupCon 86.5%, and GCN alone 85%. However, both frameworks implement a two-stage process:

- **Upstream Encoding**: In this stage, an encoder is employed to learn the graph's representation and generate node embeddings.

- **Downstream Modeling**: The embeddings produced by the encoder are then utilized in a downstream task, specifically through a simple $l2$-regularized logistic regression classifier. This classifier is trained to map the embeddings to specific labels associated with the nodes.

Both the upstream and downstream stages present challenges regarding potential overfitting. This risk of over-fitting necessitates careful hyperparameter tuning to ensure that both models generalize well to unseen data. Moreover, the complexity of tuning increases significantly during cross-validation processes, especially when multiple Graph Neural Networks (GNNs) are used as encoders. Each GNN architecture might respond differently to hyperparameter settings on both stages, requiring extra efforts on finding the optimal configuration that works consistently across different GNNs.

To optimize the hyperparameter tuning process, we have implemented a one-stage supervised contrastive learning framework in our work. This framework separates the total loss objective, J_{total}, into two main components:

- **Supervised Loss (J_{sup})**: This is derived from a traditional GNNs model and serves as the foundational training objective.

- **Contrastive Loss (J_{con})**: Calculated using the InfoNCE from two graph views, with the integration of extra positive and negative masks. Here, J_{sup} acts as a regularization term, with a parameter λ controlling the degree of regularization to improve the generalization of results from traditional GNNs.

We evaluate the necessity of using extra MLPs for projecting the embeddings, as employed in frameworks like GRACE and SupCon. Our experimental findings indicate that the validation accuracy remains consistent whether or not the projector is used. Therefore, for the contrastive loss J_{con}, we opted to maintain the same output dimension as the supervised loss J_{sup}. This decision reduces model complexity and aligns the embedding dimensions directly with the target outputs, leading to a more efficient training process. The learning algorithm is summarized in Algorithm 1.

Algorithm 1 one-stage supervised contrastive learning training algorithm

```
for epoch ← 1 to $N$ do
    Generate two augmented views $\tilde{G}_1$ and $\tilde{G}_2$ from the original graph $G$ by applying random augmentations
    Supervised Learning:
        • Compute embeddings for the original graph $G$ using the encoder $f$.
        • Calculate the supervised loss $J_{\text{sup}}$ using the embeddings from $G$ and the training labels.
    Contrastive Learning:
        • Compute embeddings for the augmented views $\tilde{G}_1$ and $\tilde{G}_2$ using the same encoder $f$.
        • Compute the contrastive loss $J_{\text{con}}$ using embeddings from $\tilde{G}_1$ and $\tilde{G}_2$, including extra positive and negative masks
    Compute the total loss: $J_{\text{total}} = J_{\text{sup}} + \lambda \times J_{\text{con}}$
    Update the encoder parameters by applying gradient descent to minimize $J_{\text{total}}$
end for
```

3.4 Model Ensembling

To enhance the robustness of our final predictions, we employ a different GNNs architectures as the encoder, each designed to capture distinct patterns and features within the graph structure. By ensembling the results from these models, we leverage their unique perspectives to contribute to a more comprehensive and accurate final decision. In this work, we utilize seven different GNNs, including:

- GCN[3]: a three-layer GCN, used to effectively leveraging local node features and topological structure by its straightforward layer-wise propagation rule.

- GraphSAGE[5]: a three-layer GraphSAGE, used to facilitating inductive learning on large graphs by sampling and aggregating features.
- GAT[6]: a three-layer GAT with 16 attention heads, used to dynamically weigh the importance of nodes in a neighborhood.
- GATv2[7]: a three-layer GATv2 with 16 attention heads, improves upon the original GAT by employing a more flexible, expressive attention mechanism.
- Transformer[8]: a two-layer TransformerConv,leveraging self-attention mechanisms to better capture global dependencies between nodes.
- APPNP[9]: a two-layer MLP with an APPNP, set with $K = 2$ and $\alpha = 0.1$, to effectively blend local and global node features.
- ChebConv[10]: a two-layer ChebConv with $K = 2$, used to capture more complex signal features on graphs by utilizing Chebyshev polynomials.

For each encoder, we apply a five-fold cross-validation to determine the CV accuracy of each model. Based on these accuracies, we employ a weighted voting mechanism to ensemble the predictions from each model to determine the final results. Specifically, for each model, we calculate the predicted probabilities for each node. These probabilities are then subjected to weighted voting, based on the CV accuracy of each model, to arrive at the final predictions. This method ensures that models with higher accuracy have a greater influence on the ensemble outcome, enhancing the overall reliability and precision of the predictions.

4 Experiments

4.1 Comparative Analysis of Regularization Terms

To assess the efficacy of Contrastive Loss, , J_{con}, as the regularization term against traditional l_2 regularization, we compare the training and validation accuracy of GCN and GraphSAGE models under varying conditions of λ and l_2 decay values. A portion constituting 20% of the training set is reserved for validation purposes and remains consistent across all tests. The outcomes of this comparative analysis are detailed in Table 2.

Model	λ	Train Acc	Val Acc	l_2 decay	Train Acc	Val Acc
GCN	0	0.94	0.85	**0**	**0.94**	**0.85**
	0.1	0.96	0.85	5×10^{-5}	0.94	0.85
	0.2	0.94	0.85	1×10^{-4}	0.94	0.85
	0.4	0.96	0.87	5×10^{-4}	0.94	0.84
	0.8	0.95	0.87	1×10^{-3}	0.94	0.84
	1.6	0.92	0.86	5×10^{-3}	0.93	0.83
	3.2	**0.89**	**0.87**	1×10^{-2}	0.92	0.83
GraphSAGE	0	0.94	0.84	**0**	**0.94**	**0.84**
	0.1	0.98	0.86	5×10^{-5}	0.94	0.84
	0.2	0.93	0.86	1×10^{-4}	0.94	0.84
	0.4	0.94	0.86	5×10^{-4}	0.94	0.84
	0.8	0.94	0.86	1×10^{-3}	0.94	0.84
	1.6	**0.92**	**0.88**	5×10^{-3}	0.94	0.84
	3.2	0.88	0.86	1×10^{-2}	0.98	0.84

Table 2: Training and validation accuracy for GCN and GraphSAGE models with various λ and l_2 decay values. The highest performance on the validation set is highlighted in boldface.

The analysis indicates that with the increase in l_2 decay values, validation accuracy shows a tendency to plateau or even decline. This observation suggests that traditional l_2 regularization may not significantly contribute to enhancing model performance within this graph learning context. On the other hand, employing J_{con} as a regularization term appears to effectively prevent over-fitting while preserving the model's capacity, proving that J_{con} assists in learning more generalized representations.

4.2 Selection of λ Values

To determine the optimal λ value for each of the seven models used in this study, we conduct a five-fold cross-validation for each model across various λ

settings. Consistency of CV folds is maintained across all models and λ values to ensure comparability. The selection criterion for the best λ is based on achieving the lowest CV loss. The results of the models' performance under different λ values are detailed in Tables 3, 4, 5, and 6. Within each table, the λ value yielding the best performance is highlighted in boldface.

Table 3: Performance of GCN and GraphSAGE Models

GCN			GraphSAGE		
λ	CV Accuracy	CV Loss	λ	CV Accuracy	CV Loss
0	0.8408	0.5133	0	0.8408	0.534
0.1	0.8468	0.5067	0.1	0.8428	0.5212
0.2	0.8468	0.5053	0.2	0.8448	0.5221
0.4	0.8468	0.5004	0.4	0.8448	0.5163
0.6	0.8448	0.4986	0.6	0.8448	0.5175
0.8	0.8468	0.4985	**0.8**	**0.8448**	**0.5148**
1	0.8368	0.5012	1	0.8448	0.5189
1.5	0.8529	0.4983	1.5	0.8448	0.5229
2	0.**8529**	**0.4971**	2	0.8388	0.5266
3	0.8508	0.5073	3	0.8428	0.5534

Table 4: Performance of GAT and GAT-v2 Models

GAT			GAT - v2		
λ	CV Accuracy	CV Loss	λ	CV Accuracy	CV Loss
0	**0.8408**	**0.5035**	0	0.8307	0.5246
0.1	0.8367	0.5079	0.1	0.8267	0.5203
0.2	0.8367	0.5085	0.2	0.8327	0.5185
0.4	0.8368	0.5104	0.4	0.8307	0.5203
0.6	0.8368	0.5118	0.6	0.8307	0.5188
0.8	0.8448	0.5128	0.8	0.8287	0.5168
1	0.8448	0.5137	1	0.8307	0.5163
1.5	0.8408	0.5188	1.5	0.8347	0.5161
2	0.8489	0.5210	**2**	**0.8368**	**0.5159**
3	0.8448	0.5294	3	0.8428	0.5215

Table 5: Performance of Transformer and APPNP Models

Transformer			APPNP		
λ	CV Accuracy	CV Loss	λ	CV Accuracy	CV Loss
0	0.8327	0.5316	0	0.8146	0.5699
0.1	0.8226	0.5277	0.1	0.8186	0.5407
0.2	0.8145	0.5340	0.2	0.8065	0.5450
0.4	0.8166	0.5445	0.4	0.8166	0.5540
0.6	0.8206	0.5311	0.6	0.8206	0.5336
0.8	0.8307	0.5240	0.8	0.8266	0.5354
1	0.8307	0.5174	1	0.8267	0.5566
1.5	0.8408	0.5094	1.5	0.8206	0.5270
2	0.8327	0.5135	2	0.8287	0.5200
3	**0.8448**	**0.4959**	**3**	**0.8347**	**0.5166**

Table 6: Performance of ChebConv Model

Lambda	CV Accuracy	CV Loss
0	0.8246	0.5592
0.1	0.8327	0.5383
0.2	0.8367	0.5403
0.4	0.8327	0.5462
0.6	0.8347	0.5463
0.8	0.8388	0.5433
1	0.8388	0.5335
1.5	**0.8448**	**0.5306**
2	0.8428	0.5339
3	0.8488	0.5415

Except for the GAT encoder, all seven models benefit from incorporating J_{con} as a regularization term. Notably, APPNP and ChebConv models show a significant improvement, with CV accuracy increasing by more than 2%. Additionally, there is no universal trend for the optimal λ value across all models, underscoring the necessity for model-specific tuning of this parameter. After applying the optimal parameters to train these encoders on the while training set, we employed an ensemble approach using the weighted-vote method discussed earlier. This strategy led to a final prediction accuracy of 87%, marking a significant improvement over traditional GNNs that do not incorporate contrastive learning.

5 Conclusion

In this study, we implemented a one-stage supervised contrastive learning framework that employs contrastive loss as a regularization term to enhance the generalization capabilities of supervised learning. The loss objective of our model comprises two components: the supervised loss derived from the original graph G and its training labels, and the contrastive loss generated from embeddings of two distinct augmented views, $\tilde{G}_1$ and $\tilde{G}_2$. The balance between these two losses is controlled by the parameter λ. This approach demonstrated a significant advantage over traditional l_2 decay.

We conducted extensive experiments to identify the optimal λ values for seven different GNNs encoders. The experimental results show that most models benefit from incorporating contrastive loss, underscoring its effectiveness in enhancing learning dynamics and improving model performance. This study highlights the potential of integrating supervised contrastive learning into a one-stage training framework, which not only achieves superior results but also streamlines the hyperparameter tuning process.

6 Future Work

Building on the successful integration of supervised contrastive learning in GNNs, as detailed in this study, several avenues of future work emerge that could further refine and expand the capabilities of this approach:

1. **Mechanism Behind Optimal λ Values:** In our experiments, there is no universal rule for selecting the optimal λ value, making it necessary to tailor this parameter for each model. Analyzing the aggregation functions of each model to reveal the mechanisms behind each optimal λ could provide valuable insights into the interaction between model architecture and regularization, potentially guiding more effective training strategies.

2. **Impact of Data Augmentation:** In this study, all our experiments are conducted under the same graph augmentations. Studying how different data augmentation techniques can affect the optimal λ value could significantly enrich our understanding of model sensitivity and adaptation.

3. **Different Methods for Combining Loss Objects:** Instead of using a fixed λ throughout training, several other methods can be explored for combing supervised loss and contrastive loss, for example: dynamically adjust λ based on certain performance metric for each epoch, using geometric or harmonic mean, or combining losses based on specific conditions, such as the accuracy exceeding a threshold.

4. **Parameter Exploration for GRACE Method:** While we have retained the code to apply the GRACE method for calculating contrastive loss, detailed experiments to determine the optimal λ values and to measure the corresponding performance enhancements have not yet been conducted. This approach will help comparing the performance enhancements between supervised and semi-supervised contrastive learning.

References

[1] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," *ArXiv*, vol. abs/2006.04131, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219531264

[2] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 18 661–18 673. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/d89a66c7c80a29b1bdbab0f2a1a94af8-Paper.pdf

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning*

Representations, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[4] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *ArXiv*, vol. abs/1807.03748, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49670925

[5] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Neural Information Processing Systems*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:4755450

[6] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', and Y. Bengio, "Graph attention networks," *ArXiv*, vol. abs/1710.10903, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3292002

[7] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *ArXiv*, vol. abs/2105.14491, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235254358

[8] Y. Shi, Z. Huang, W. Wang, H. Zhong, S. Feng, and Y. Sun, "Masked label prediction: Unified massage passing model for semi-supervised classification," *ArXiv*, vol. abs/2009.03509, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221534325

[9] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:67855539

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Neural Information Processing Systems*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:3016223