JGCL: Joint Self-Supervised and Supervised Graph Contrastive Learning

Selahattin Akkas
Indiana University Bloomington
Bloomington, Indiana, USA
sakkas@iu.edu

Ariful Azad
Indiana University Bloomington
Bloomington, Indiana, USA
azad@iu.edu

ABSTRACT

Semi-supervised and self-supervised learning on graphs are two popular avenues for graph representation learning. We demonstrate that no single method from semi-supervised and self-supervised learning works uniformly well for all settings in the node classification task. Self-supervised methods generally work well with very limited training data, but their performance could be further improved using the limited label information. We propose a joint self-supervised and supervised graph contrastive learning (JGCL) to capture the mutual benefits of both learning strategies. JGCL utilizes both supervised and self-supervised data augmentation and a joint contrastive loss function. Our experiments demonstrate that JGCL and its variants are one of the best performers across various proportions of labeled data when compared with state-of-the-art self-supervised, unsupervised, and semi-supervised methods on various benchmark graphs.

CCS CONCEPTS

• Computing methodologies → Supervised learning; Unsupervised learning; Neural networks; Learning latent representations.

KEYWORDS

graph representation learning, self-supervised learning, supervised contrastive learning

ACM Reference Format:
Selahattin Akkas and Ariful Azad. 2022. JGCL: Joint Self-Supervised and Supervised Graph Contrastive Learning . In Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3487553.3524722

1 INTRODUCTION

Graph representation learning aims to learn low-dimensional embeddings using neighborhood information and node features. Recently, Graph Neural Network (GNNs) have been very successful to learn graph representation and solve various downstream tasks such as node classification and link prediction [10, 16, 34, 37].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524722

In the GNN field, there are multiple learning methods including semi-supervised, unsupervised, supervised, and self-supervised learning. Semi-supervised learning uses a small portion of labeled and abundant unlabeled data for the model training [17, 18, 43]. Recently, many researchers have utilized semi-supervised learning for graph representation learning [2, 8, 16, 19, 31, 36, 38, 41, 42]. However, training GNNs in a (semi-)supervised fashion requires labeled graph data that is labor-intensive and mostly unavailable. To tackle this issue, researchers have applied contrastive loss with self-supervised learning (SSL), which is an unsupervised model training approach [11, 14, 24, 30, 32, 39, 40, 44, 45]. SSL creates two views by augmenting the data and then contrasts pairs. In SSL, nodes (in different views) that are originated from the same node have similar representations, while others have different representations. Several graph augmentation techniques have been discussed in the literature. For instance, GRACE [44] randomly drops edges and masks features, GCA [45] uses degree centrality, eigenvector or PageRank for edge and feature drop probabilities, DGI [35] shufles features and contrasts local and global embeddings. Overall, SSL has enriched graph representation learning, but it lacks supervision: it ignores limited but beneficial label information.

On the other hand, supervised contrastive learning (SupCon) [15], a supervised approach in computer vision, uses labels to improve the performance of SSL. In SupCon, pairs of training examples from the same class are considered positive pairs and others are considered negative pairs. In graph representation learning, CG^3 [36] uses SupCon with supervised and generative loss. The authors use GCN and hierarchical graph convolutions (HGCN) and contrast their outputs. While they show combining supervised, SSL, SupCon, and generative losses help, they give the same importance to SSL and SupCon. Another work [41] uses SupCon in a student-teacher network with pseudo-labels in training. While this work uses SupCon, their architecture is different from the contrastive-learning frameworks.

While the benefits of semi-supervised, SSL, and to some extent SupCon methods for graph representation learning have been demonstrated in the literature, it remains unclear which methods work the best for different settings. We argue that no single method works uniformly well for all settings. For example, Fig. 1 shows that SSL excels with very limited labeled data (0.5%), but GCN and SupCon perform better than SSL when more than 5% nodes are labeled for training. This observation is not surprising considering the fact that SSL is designed to tackle the scarcity of labeled data. Can we utilize SSL, SupCon and GCN simultaneously to unify their advantages in graph representation learning? This paper answers this questions afirmatively with a joint self-supervised and supervised graph contrastive learning (JGCL). Although joint learning

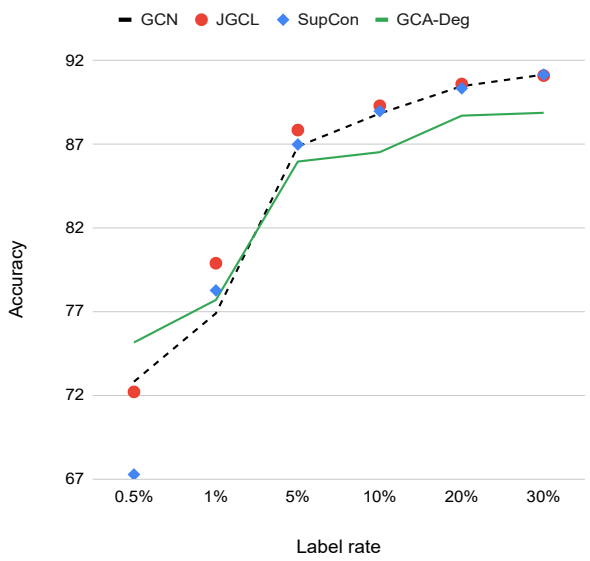

Figure 1: Accuracy vs training label rate for the Amazon-Computers dataset. While SSL works better for very limited labeled data (0.5%), SupCon works better for a reasonable amount of labeled data (1, 5, 10%). But after 20%, semi-supervised training gives better results. Our method JGCL is one of the best performers for all settings.

can be applied to any representation learning task, we focus on graph presentation learning in this work.

JGCL unifies SSL and SupCon both in data augmentation and in the contrastive loss function. In between data augmentation and contrastive loss, JGCL uses GCN encoders to generate node embeddings. Thus, it takes advantage of SSL, SupCon and GCN in a joint learning framework. Moreover, we propose an augmentation strategy for the joint training: for the SupCon part, we apply different edge drop probabilities for edges connecting to labeled nodes. Our results demonstrate that JGCL and its variants are one of the best performers across all settings when compared with state-of-the-art SSL methods, GCN, and SupCon for some benchmark graphs. The main contributions of this paper are:

- We developed JGCL for jointly training SSL and SupCon to improve generalization with different ratios of labeled data.
- We introduce a new label-based augmentation strategy for the joint training.
- Our results demonstrate that JGCL and it variants perform better than state-of-the-art SSL methods, GCN, and SupCon for some benchmark graphs.

2 RELATED WORK

2.1 Graph Representation Learning

Graph representation learning aims to construct an embedding representing the graph's structure and its data [27]. Traditional graph representation works can be categorized to two classes. Factorization approaches[1, 4, 22, 25] minimizes dot product of representations. Random walk based approaches like DeepWalk [23], node2vec[9], Line[33], Harp[5], and force2vec[26] sample neighborhood nodes and apply skip-gram models to obtain the representations. Recently, GNN models including GCN [16], GAT[34], and GIN[37] have been successful in graph representation learning and are widely used in SSL as graph encoders.

2.2 Contrastive Learning

Contrastive learning is an SSL approach that the encoder learns to give similar embeddings in the embedding space for similar input pairs while different embeddings for different pairs. To create similar pairs from the same input, augmentation techniques are used. There are many augmentation techniques for images [6, 7, 12, 13, 15] including cropping, cutout, color distortion Sobel filter, noise, blur, rotating, horizontal flipping, grayscale conversion. Although many augmentation approaches exist for images, there are quite limited augmentation options on graphs that mostly consist of edge dropping and feature masking. DGI [35] shufles features and uses contrastive learning to maximize local (node) - global (graph) mutual information, GRACE [44] drops edges and masks features randomly, GCA [45] utilizes degree centrality, eigenvectors or PageRank to keep some important edges and features, GROC [14] uses gradient information to add or remove edges. GROC's approach makes representation more robust to adversarial attacks; however, training requires more time. MVGLR [11] uses diffusion and subgraph sampling and contrast local-global pairs, GraphCL [40] and GraphCL-Automated [39] use node dropping, subgraph sampling, random edge dropping or adding, feature masking for augmentation and contrasts global information. Our work is based on GRACE and GCA. We extend GCA's degree-based augmentation to have label-based augmentation and utilize SupCon.

2.3 Graph-based Semi-Supervised Learning

In semi-supervised settings, labeled and unlabeled data are used together. GCN [16] is considered as semi-supervised training since it uses unlabeled node's relationships and features in training. Therefore, SupCon on graphs is also regarded as semi-supervised learning. Supervised contrastive learning has been used in computer vision first [15]. Recently, [36] has used SupCon in graph domain combining with supervised and generative loss. The authors use localized (GCN) and hierarchical graph convolutions (HGCN) and contrast their embeddings. The authors do not apply edge or feature-based augmentation. The study combines SSL, SupCon, cross-entropy loss, and generative loss. While generative loss weight can be adjusted in [36], SSL and SupCon share the same weight. Our work differs from [36] in the following ways: different weights for SSL and SupCon, no generative loss, a linear classifier trained after presentations are learned (cross-entropy loss), degree and label based augmentations, shared GNN encoders. Another work [41] uses SupCon in a student-teacher network settings. They use pseudo-labels in training. Since the architectures and the evaluation protocols are different, we are unable to compare our results with the studies mentioned above.

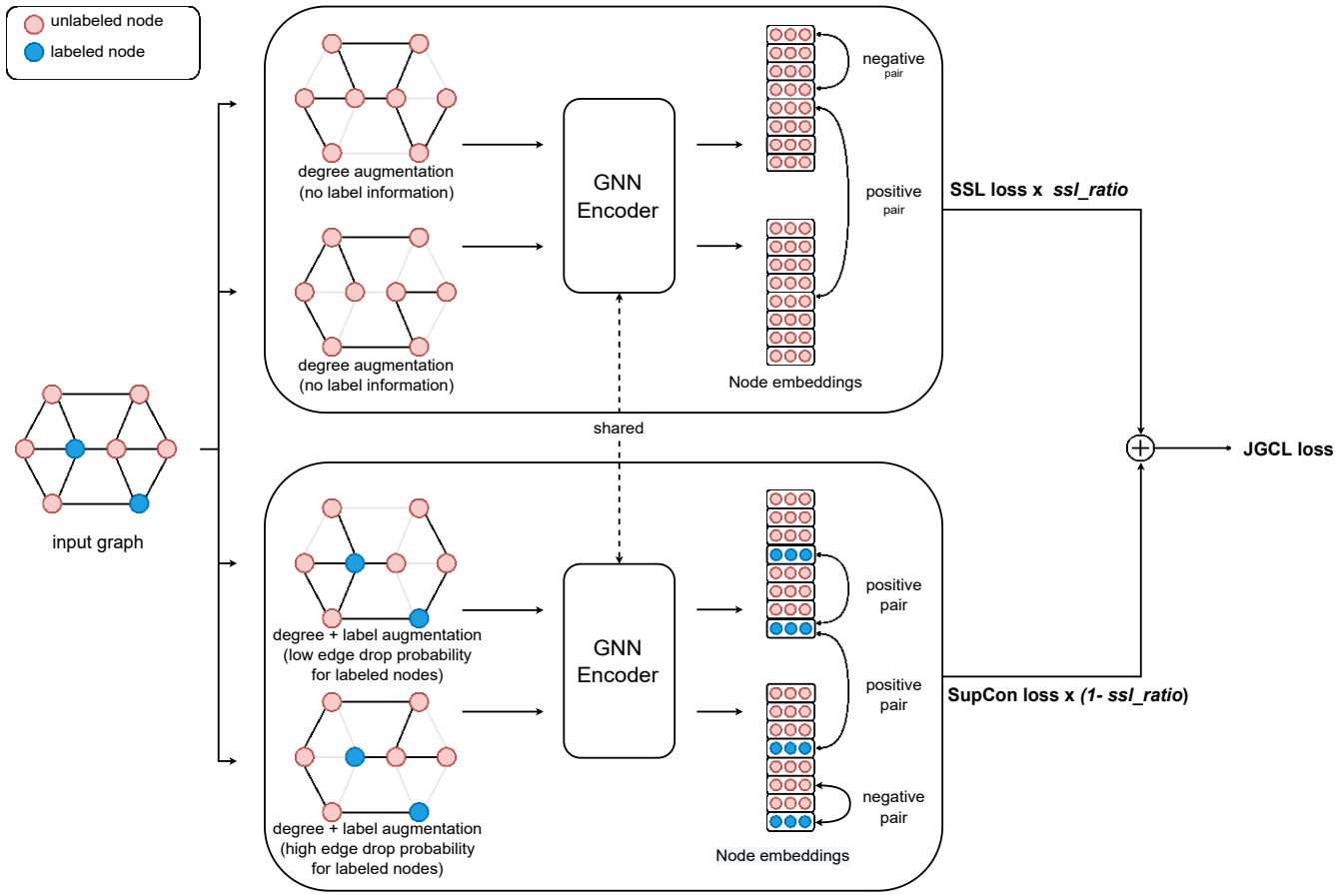

Figure 2: Overview of our proposed JGCL framework. There are four views generated where the top two views ignore labels, and the bottom two views are generated based on augmentations that consider node labels. Label based augmented views show high and low edge drop probability for labeled nodes. The top part represents SSL and bottom part represents SupCon. The combined SSL and SupCon losses are optimized jointly.

3 METHOD

3.1 Preliminaries

Here, we introduce the GNN notation we use throughout the paper. Let $G = \{V, E\}$ denote a graph where $V = \{v_1, v_2, ..v_N\}$ are the set of N nodes and $e_{ij} \in E$ is an edge between v_i and v_j. Additionally, $X \in \mathbf{R}^{N \times F}$ denotes a feature matrix where each node has an F-dimensional feature vector. $A \in \{0, 1\}^{N \times N}$ denotes the adjacency matrix of the graph where $A_{ij} = 1$ if there is an edge between v_i and v_j; otherwise $A_{ij} = 0$. Finally, $y = \{y_1, y_2, ..., y_N\}$ denotes labels for all nodes. In most graph learning tasks, only a small fraction of nodes are labeled and they are used to train GNN models. The task is to learn representations $H = f(X, A)$ where f is a GNN encoder (e.g. GCN, GAT).

3.2 Overview of the JGCL and Supervised Contrastive Learning

Our proposed framework jointly trains SSL and SupCon as illustred in Fig. 2. We generate four views by augmenting the input graph. While no label information is used in the SSL part, we use label information in SupCon to generate views. We introduce a new hyperparameter for each view to control edge drop probabilities for edges connecting to a labeled node. The bottom two views in Fig. 2 show low and high drop probability effects. Detailed augmentation method is explained in section 3.3. After the view generation process, we use a shared GNN encoder and get the embeddings of the views. We apply a contrastive loss to the first two views and a supervised contrastive loss to the other two views. While contrastive loss uses all node embeddings, the supervised contrastive loss is applied to nodes in the training set. Finally, we combine two losses with the ssl_{ratio} that controls the relative importance of losses.

3.3 Augmentation

We follow GCA's degree based augmentation strategy since it provides good results and it is easy to implement. Let k_u and k_v are degrees for node u and v. An edge centrality value is defined as $w^e_{uv} = (k_u + k_v)/2$ for undirected graphs and $w^e_{uv} = k_v$ for directed graphs since an edge importance is generally characterized by the target node [21]. In GCA, the authors use $s^e_{uv} = \log w^e_{uv}$ to reduce the effect of the nodes with very dense connections. Then

the centrality values are normalized and the edge drop probability is defined by:

$$p^e_{uv} = \min(\frac{s^e_{max} - s^e_{uv}}{s^e_{max} - \mu^e_s}.p_e, p_\tau), \quad (1)$$

where p_e is the overall edge removing probability, μ^e_s is average s^e_{uv}, s^e_{max} is maximum s^e_{uv} value, and p_τ is a cut-off probability to prevent the overly corrupt graph structure.

3.3.1 Label augmentation. We modify GCA's overall edge removing probability for edges connecting to labeled nodes in the SupCon part of JGCL (see Fig. 2). Depending on the dataset, increasing or decreasing the contrast ratio affects the performance. Therefore we use 0.5, 1.0, 1.5, and 2.0 values for the labeled drop multiplier. For instance, if $p_e = 0.3$ and labeled drop multiplier is 2.0, edge drop probability for edge connecting to labeled nodes becomes $0.3 \cdot 2.0 = 0.6$.

3.4 GNN Encoder

We use a 2-layer GCN [16] as encoder.

$$H_1(X, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}XW_1), \quad (2)$$

$$f(X, A) = H(H_1, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H_1W_2). \quad (3)$$

Here, σ is an activation function (i.e. ReLU and pReLU), $\hat{A} = A + I$ is the self loop added adjacency matrix, $\hat{D}$ is the degree matrix of the $\hat{A}$ and W_i is the weight matrix that we train.

3.5 Contrastive-Learning Methods

3.5.1 Self-supervised learning. Self-supervised learning is an unsupervised technique that does not use label information. The task is to learn representations by contrasting augmented pairs. The augmentation can be done by distorting the adjacency matrix A(e.g., dropping or adding edges, subgraph sampling) and the feature matrix X (e.g., feature masking). We obtain two views after the augmentation: $G_1 = \{X_1, A_1\}$ and $G_2 = \{X_2, A_2\}$. After representations are obtained for both views, representations are fed to the projection head, which is two-layer MLP, and projected representations Z_1 and Z_2 are obtained. We can merge them as $Z = [Z_1, Z_2]$ where contains $2N$ pairs. While z_i is a projected representation from a view, $z_{j(i)}$ is the corresponding pair from the other view. For each node, there is one positive pair (i.e. $<z_i, z_{j(i)}>$), there are $2N - 2$ negative pairs.

The idea is that positive pairs originated from same source node should have more similar representations compared to all other pairs. This can be learned using the contrastive loss:

$$L_{SSL} = \frac{-1}{2N}\sum_{i=1}^{2N} \log \frac{\exp((z_i \cdot z_{j(i)})/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k,i]}\exp((z_i \cdot z_k)/\tau)} \quad (4)$$

Here, $\cdot$ is cosine similarity, τ is temperature parameter, and $\mathbf{1}$ is an indicator function that is 1 if $i \neq k$ else 0.

3.5.2 Supervised contrastive learning. Supervised contrastive learning (SupCon) is considered semi-supervised representation learning since it requires labels and aggregates information from unlabeled neighbors on graphs. SupCon can be defined as:

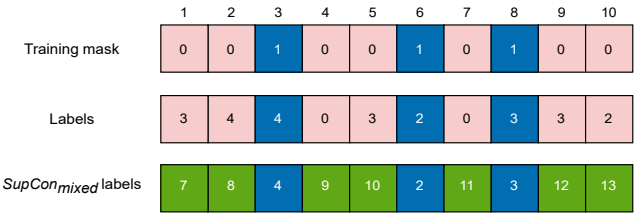

Figure 3: $SupCon_{mixed}$ labels for Cora dataset's first 10 nodes. We use the labels for the nodes that are in the training set(blue colors). We assign a unique number for the rest of nodes starting from max value of labels + 1 (i.e. 7 for Cora).

$$L_{SupCon} = \frac{1}{2N}\sum_{i=1}^{2N}\frac{-1}{|P(i)|}\sum_{p\in P(i)} \log \frac{\exp((z_i \cdot z_p)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k,i]}\exp((z_i \cdot z_k)/\tau)} \quad (5)$$

Here $P(i)$ denotes all positive samples which belong to the same class with z_i, and $|P(i)|$ represents the number of positive samples for z_i. In this approach, representations belonging to the same class should be more similar than the others. SupCon improves generalization [15], but it can only be used for labeled nodes which are highly limited for most of the graph data.

3.5.3 $SupCon_{mixed}$. While SSL ignores labels, SupCon can only be used for the labeled nodes. Since the labeled nodes are a small percentage of the dataset, there will be quite a small number of positive and negative samples in SupCon which slightly reduces the performance of the SupCon. Here, we propose a simple technique to combine SupCon and SSL in a single loss function: we assign a unique class id for unlabeled nodes (see Fig. 3). We consider unlabeled nodes as single instance classes. This way, we can use all nodes in semi-supervised settings. One limitation is that a pair of labeled and unlabeled samples belonging to the same class will be considered negative samples.

Algorithm 1 JGCL Algorithm

Require: feature matrix X, adjacency matrix A, labels vector y, max epoch ep, ssl_ratio

for k ← 1 to ep do
  X1, X2 ← aug_feature(X), aug_feature(X) ▷ Augments features
  A1, A2 ← aug_adj(A), aug_adj(A) ▷ Augments adjaceny matrix based on GCA-degree
  A3, A4 ← aug_adj_lbl(A, y), aug_adj_lbl(A, y) ▷ Augments adjacency matrix using different drop ratios for labeled nodes
  z1, z2 ← model(X1, A1), model(X2, A2)
  z3, z4 ← model(X1, A3), model(X2, A4)
  L_{SSL} ← ssl_loss(z1, z2)
  L_{SupCon} ← supcon_loss(z3, z4)
  L_{JGCL} ← ssl_ratio ▷ L_{SSL} + (1 − ssl_ratio) ▷ L_{SupCon}
  update model
end for

3.5.4 JGCL. We jointly train SSL and SupCon and compute the loss:

Table 1: Dataset Summary

Dataset	Nodes	Edges	Features	Classes
Amazon-Computers [29]	13752	491722	767	10
Amazon-Photo [29]	7650	238162	745	8
Cora [38]	2708	10556	1433	7
PubMed [38]	19717	88648	500	3
DBLP [3]	17716	105734	1639	4

$$L_{JGCL} = ssl\_ratio \cdot L_{SSL} + (1 - ssl\_ratio) \cdot L_{SupCon}. \quad (6)$$

Here, $ssl\_ratio \in [0, 1]$ denotes the ratio. We provide separate augmented views to SSL and SupCon. We use our label based augmentation approach for the SupCon views.

Algorithm 1 describes how JGCL works. In line 3 and 4, we generate views of the graph based on SSL and label-based augmentations, respectively. Line 5 and 6 use GCN encoders to embed all four views. Line 7 and 8 then compute SSL and SupCon losses that are combined using Equation 6.

3.6 Cross-Entropy Loss

We train a linear classifier after representation learning to evaluate learned representations. The linear classifier and the cross-entropy loss can be defined as:

$$\hat{y} = log\_softmax(HC + b) \quad (7)$$

$$L_{ce} = \sum_{i=1}^{n} \sum_{j=1}^{c} y_{ij} log(\hat{y}_{ij}) \quad (8)$$

where H learned representations, C and b linear classifier's parameters, $\hat{y}$ predictions, y labels, c number of classes, and n number of samples in the training set.

4 EXPERIMENTS

4.1 Datasets

We use Cora, PubMed, DBLP, Amazon-Computers, and Amazon-Photo datasets to evaluate the proposed methods. Summaries of datasets are provided in Table 1.

- Cora and PubMed [38] are citation networks where nodes represent papers, edges represent citations, and features are bag-of-words representations.
- Amazon-Computers, and Amazon-Photo [29] are created from Amazon co-purchase dataset [20]. Here, nodes represent products, and edges represent products frequently bought together. Features are product reviews that are embedded using bag-of-words.
- DBLP [3] is another citation network where nodes represent papers, edges represent citations, and features are bag-of-words representations.

We use the same split ratios used in GCA [45]: we use 10% of the data for training, 10% of the data for validation, and the remaining for the testing.

4.2 Evaluation protocol

We use linear evaluation protocol which is introduced in [35], and used in GRACE [44] and GCA [45]. We train the encoder and use the embeddings to train the logistic regression classifier. While GRACE and GCA train the encoder once and train the classifier twenty times, we train both the encoder and the classifier ten times. The classifier was trained for 3000 epochs using Adam optimizer with a 0.01 learning rate for each repeat. Note that GRACE and GCA create a new random split for the logistic regression training. However, our approach uses a training split in the representation learning phase and learns the node representations better for the nodes in the training split. Creating a new random split for the classifier can put some of these nodes to test split which can give confusing results. To make the comparison fair, we fix the subset for the linear classifier training for all methods: we use the same splits mentioned above. We also use the same split for each experiment repeat.

For the representation learning, we use the same hyperparameters with GRACE and GCA. We introduce three extra hyperparameters: ssl_ratio, and two labeled drop probabilities, which are explained in Section 3.5.4. We find these parameters by grid-search for each dataset. The hyperparameters are summarized in Table 2.

4.3 Baselines

We compare our results with traditional methods including DeepWalk [23], node2vec [9], raw features classification, raw features + DeepWalk classification, and deep learning methods including GCN [16], GRACE [44] and GCA [45]. For the traditional methods, we use the Karate Club framework [28] for DeepWalk and node2vec experiments. Note that we use the same embedding dimensions for each method. With different architectures, there can be better results obtained.

Table 2: Hyperparameters. We use same hyperparameters with GRACE and GCA. The bold columns are newly introduced hyperparameters. Values are found by grid search.

Dataset	edge drop prob 1	edge drop prob 2	feature drop prob 1	feature drop prob 2	encoder dim	projection dim	tau	epochs	learning rate	ssl ratio	labeled drop prob 1	labeled drop prob 2
Amazon-Computers	0.6	0.3	0.2	0.3	128	128	0.2	2000	0.01	0.5	1.0	2.0
Amazon-Photo	0.3	0.5	0.1	0.1	256	64	0.3	2000	0.1	0.7	1.5	0.5
Cora	0.2	0.4	0.3	0.4	128	128	0.4	1000	0.0005	0.7	1.0	2.0
PubMed	0.4	0.1	0.0	0.2	256	256	0.7	1500	0.001	0.7	1.5	0.5
DBLP	0.1	0.4	0.1	0.0	256	256	0.9	1000	0.001	0.7	2.0	1.0

Table 3: Classification accuracies of compared methods on Amazon Computers, Amazon Photo, Cora, PubMed, and DBLP datasets. Standard deviations are also reported for GNN-based methods. The highest accuracy on each dataset is highlighted in bold.

	Method	Input	Amazon-Computers	Amazon-Photo	Cora	PubMed	DBLP
Unsupervised	raw features	X	73.30±0.00	80.11±0.00	55.03±0.00	80.48±0.00	67.90±0.00
	node2vec	A	79.54±0.00	84.54±0.00	55.86±0.00	63.93±0.00	59.12±0.00
	DeepWalk	A	79.36±0.00	82.45±0.00	56.69±0.00	64.03±0.00	59.66±0.00
	DeepWalk + features	X, A	79.70±0.00	83.02±0.00	57.43±0.00	75.78±0.00	63.78±0.00
Semi-supervised	GCN	X, A, Y	88.82±0.23	93.08±0.21	81.17±0.40	85.55±0.16	81.33±0.15
Self-supervised	GRACE	X, A	87.13±0.17	92.22±0.91	82.81±0.30	85.73±0.42	84.32±0.19
	GCA-Degree	X, A	86.51±0.46	81.54±23.75	83.63±0.51	85.08±0.50	84.36±0.14
Supervised Contrastive	SupCon	X, A, Y	88.96±0.11	30.08±5.18	80.01±0.76	84.03±0.39	79.42±0.41
Joint (Ours)	$\text{SupCon}_{\text{mixed}}$	X, A, Y	88.08±0.12	92.21±1.54	84.03±0.36	85.99±0.40	52.87±1.39
	JGCL-no label aug	X, A, Y	89.14±0.17	79.99±29.35	82.04±0.98	84.12±0.42	80.47±0.53
	JGCL	X, A, Y	89.28±0.11	93.16±0.10	82.93±0.66	84.84±0.22	81.30±0.15

Table 4: Amazon-Computers results with different training label rates.

Label Rate	0.1%	0.5%	1.0%	5.0%	10.0%	20.0%	30.0%	40.0%	50.0%
GCN	52.08±0.91	72.81±0.53	76.89±1.48	86.85±0.32	88.82±0.23	90.45±0.13	91.14±0.10	91.42±0.12	91.82±0.04
GCA-Deg	57.63±1.45	75.15±1.31	77.69±2.11	85.95±0.68	86.51±0.46	88.69±0.16	88.86±0.31	88.80±0.18	89.07±0.17
GRACE	57.76±1.84	74.58±0.81	77.57±1.50	85.27±0.46	87.13±0.17	87.88±0.32	88.27±0.35	88.30±0.39	88.57±0.23
SupCon	50.31±1.68	67.27±0.62	78.26±2.29	86.97±0.59	88.96±0.11	90.32±0.15	91.14±0.20	91.34±0.23	91.21±0.13
$\text{SupCon}_{\text{mixed}}$	55.19±0.74	75.20±0.45	77.40±0.35	86.36±0.29	88.08±0.12	89.53±0.14	90.35±0.22	90.96±0.35	91.11±0.14
JGCL-no label aug	57.96±1.98	70.73±1.10	79.52±0.68	87.64±0.18	89.14±0.17	90.74±0.14	91.16±0.07	91.49±0.20	91.56±0.15
JGCL	57.58±1.15	72.20±1.31	79.88±0.32	87.83±0.30	89.28±0.11	90.58±0.19	91.08±0.17	91.28±0.08	91.21±0.25

4.4 Node Classification Results

We evaluate the performance of the joint SSL and SupCon methods on semi-supervised node classification tasks and show results in Table 3. Our joint approaches give better results than pure semi-supervised methods such as GCN and pure self-supervised methods such as GRACE, and GCA in four out of five datasets.

For Amazon-Computers, SupCon gives better results than the other baselines. However, $\text{SupCon}_{\text{mixed}}$ performance is worse than SupCon. This is caused by false negatives since a true positive pair (i.e., one from the labeled set and the other from the unlabeled set) repel each other. On the other hand, our JGCL gives the best result.

For Amazon-Photo, degree-based augmentation causes training to be stuck in the local minima. We can see this looking at the variances of GCA-Deg. That is why SupCon under-performs. But, $\text{SupCon}_{\text{mixed}}$ does not stuck in the local minima since there are more pairs in training. However, the GCN baseline is quite strong; only JGCL outperforms the GCN. In Cora and PubMed, SupCon under-performs since the number of training samples is not sufficient. Increasing the samples by using $\text{SupCon}_{\text{mixed}}$ improves the accuracy. Finally, for DBLP, SupCon doesn't help since the graph structure is less important for the datasets. Raw features give better results than the node2vec and DeepWalk for the datasets. Thus, combining SSL methods with SupCon harms DBLP.

We also compare JGCL with and without label augmentation (no label aug) to see if the improvement is caused by labeled augmentation. We set the drop probability multipliers to 1.0. Table 3 shows that label augmentation improves the results in all datasets.

4.5 Label Rate Results

We also study the performance JGCL for different label rates. Table 4 shows results for the Amazon-Computers dataset. The results show that combining SSL and SupCon increases the accuracy. However, GCN gives good results if there are adequate labeled data (more than 50%) for training. Note that we use the same hyperparameters from Table 2 in the label rate experiments. Changing the ssl_ratio or drop ratios for different label rates will lead to better accuracies for joint training approaches.

5 CONCLUSION

In this work, we have demonstrated that no single method from semi-supervised and self-supervised learning works well for all settings. We have introduced our label-based augmentation strategy and joint graph contrastive learning (JGCL) to capture the benefits of both self-supervised and supervised contrastive learning. Our preliminary results show that JGCL and SupCon mixed approaches improve classification accuracy over baselines. As future work, we want to explore how we can improve the label-based augmentation and JGCL and automatically identify the ssl_ratio parameter.

ACKNOWLEDGMENTS

This research is supported by the NSF OAC-2112606 grant.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In Proceedings of the 22nd international conference on World Wide Web. 37–48.

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of machine learning research 7, 11 (2006).
[3] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017).
[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In Proceedings of the 24th ACM international on conference on information and knowledge management. 891–900.
[5] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. Harp: Hierarchical representation learning for networks. In Proceedings of the AAAI conference on artificial intelligence, Vol. 32.
[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In International conference on machine learning. PMLR, 1597–1607.
[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020).
[8] Chen Gong, Tongliang Liu, Dacheng Tao, Keren Fu, Enmei Tu, and Jie Yang. 2015. Deformed graph Laplacian for semisupervised learning. IEEE transactions on neural networks and learning systems 26, 10 (2015), 2261–2274.
[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 855–864.
[10] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems. 1025–1035.
[11] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In International Conference on Machine Learning. PMLR, 4116–4126.
[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 9729–9738.
[13] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670 (2018).
[14] Nikola Jovanović, Zhao Meng, Lukas Faber, and Roger Wattenhofer. 2021. Towards robust graph contrastive learning. arXiv preprint arXiv:2102.13085 (2021).
[15] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. arXiv preprint arXiv:2004.11362 (2020).
[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In International Conference on Learning Representations (ICLR).
[17] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In Thirty-Second AAAI conference on artificial intelligence.
[18] Wanyu Lin, Zhaolin Gao, and Baochun Li. 2020. Shoestring: Graph-based semi-supervised classification with severely limited labeled data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 4174–4182.
[19] Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. 2019. A flexible generative framework for graph-based semi-supervised learning. Advances in Neural Information Processing Systems 32 (2019), 3281–3290.
[20] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. 43–52.
[21] Mark Newman. 2018. Networks: An Introduction (Second Edition). Oxford university press.
[22] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 1105–1114.
[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 701–710.
[24] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 1150–1160.
[25] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In Proceedings of the eleventh ACM international conference on web search and data mining. 459–467.
[26] Md Khaledur Rahman, Majedul Haque Sujon, and Ariful Azad. 2020. Force2Vec: Parallel force-directed graph embedding. In 2020 IEEE International Conference on Data Mining (ICDM). IEEE.
[27] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. arXiv preprint arXiv:2004.11198 (2020).
[28] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. 2020. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20). ACM, 3125–3132.
[29] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018).
[30] Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 2021. 3D Infomax improves GNNs for Molecular Property Prediction. arXiv preprint arXiv:2110.04126 (2021).
[31] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. arXiv preprint arXiv:1908.01000 (2019).
[32] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. arXiv preprint arXiv:2106.05819 (2021).
[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web. 1067–1077.
[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).
[35] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. ICLR (Poster) 2, 3 (2019), 4.
[36] Sheng Wan, Shirui Pan, Jian Yang, and Chen Gong. 2020. Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. arXiv preprint arXiv:2009.07111 (2020).
[37] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018).
[38] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In International conference on machine learning. PMLR, 40–48.
[39] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. arXiv preprint arXiv:2106.07594 (2021).
[40] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems 33 (2020), 5812–5823.
[41] Hanlin Zhang, Shuai Lin, Weiyang Liu, Pan Zhou, Jian Tang, Xiaodan Liang, and Eric P Xing. 2020. Iterative graph self-distillation. arXiv preprint arXiv:2010.12609 (2020).
[42] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In Proceedings of the 20th International conference on Machine learning (ICML-03). 912–919.
[43] Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. (2005).
[44] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131 (2020).
[45] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In Proceedings of the Web Conference 2021. 2069–2080.