Feature Fusion for Graph Convolutional Networks in Semi-Supervised Image Classification

Marina Chagas Bulach Gapski, Lucas Pascotti Valem, Daniel Carlos Guimarães Pedronette
Department of Statistics, Applied Mathematics and Computing
São Paulo State University (UNESP), Rio Claro, Brazil
marina.gapski@unesp.br, lucas.valem@unesp.br, daniel.pedronette@unesp.br

***Abstract*—In recent years, the volume of multimedia data has been rapidly increasing across various applications. Consequently, classification methods capable of handling scenarios with limited labeled data (e.g., semi-supervised, weakly supervised) have become critically important, particularly because acquiring labeled data is often expensive and time-consuming. Regarding image data, feature extraction approaches are commonly employed in many tasks. Feature extraction involves identifying and extracting key characteristics or patterns, such as edges, textures, shapes, and colors. Nowadays, most extractors consider deep learning strategies, such as Convolutional Neural Networks (CNNs) and Vision Transformers (ViT). With various feature extractors available in the literature, there is a wide diversity of features that can be considered. The features extracted from an image depend on the application, the extractor used, and its configuration. Therefore, combining different extractors can be a promising strategy to exploit complementary information. Graph Convolutional Networks (GCNs) are fundamental and promising strategies in the scenario of semi-supervised image classification, being able to leverage labeled and unlabeled data, and exploiting the graph structures that offer valuable information. This work proposes an approach for GCNs in scenarios where labeled data is scarce, combining sets of features and graphs considering different extraction approaches. Among the main contributions, the experimental results reveal that these combinations and the use of manifold learning to process these graphs improve the classification results in most cases.**

I. Introduction

Over the past few years, the volume of multimedia data has been increasing exponentially. Images, videos, and audio are shared daily on various platforms by millions of people. Due to the enormous amount of available material, tasks that were previously done manually have become unfeasible to be carried out by humans due to financial costs, labor, and time constraints [1].

Due to the difficulty in keeping up with the current pace of multimedia content creation, machine learning methods have proven to be very promising in classification tasks. Supervised approaches can save a lot of time but present a challenge: the need for a large amount of labeled data. In large datasets with millions of images, this becomes a significant challenge. This is a central issue, which constitutes one of the main advantages of semi-supervised and unsupervised strategies: the ability to operate with few or no labels.

In general terms, Semi-Supervised Learning [2] (SSL) can be defined as something that lies halfway between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervised information - but not necessarily for all examples. Frequently, this information will be the labels associated with some of the examples [2]. This strategy drastically reduces the need for manual work, as only a few data points need to be classified manually. Therefore, semi-supervised and unsupervised techniques can save a lot of time and labor in activities that, in the past, would have needed to be done manually [3].

The applications of such tasks are diverse and can involve audio, video, and images. Image datasets with millions of items and frequent new entries can greatly benefit from semi-supervised and unsupervised classification, for instance, databases with images of different flower species [4], various dog breeds [5], car brands [6], among others. These datasets simulate real-world problems and enable the evaluation of different classification strategies. Currently, applications involving these strategies are increasingly common in everyday life. An example includes photo organizing apps that automatically identify people, animals, places, and objects in images, catering to millions of users with thousands of new entries every day. Social media applications can also benefit from neural networks [7].

The diversity of applications has led to the development of many models and feature extractors. Convolutional Neural Networks (CNNs) [8], which use various types of layers, including convolutional layers, non-linear layers, and pooling layers, among others, have been frequently employed. Recently, Transformer-based models have also been applied for feature extraction. Examples include ResNet [9], ViT-b16 [10], and Swin-tf [11]. Combining different extractors might be a strategy to obtain complementary information from the same dataset because the mentioned extractors evaluate the dataset and extract the most relevant characteristics for their model, potentially having discrepancies among themselves.

From another perspective, graph-based models and convolutional networks have been proposed, achieving remarkable results. For instance, Graph Convolutional Networks (GCNs) [12], [13] have the ability to perform convolutions in the non-Euclidean domain defined by graphs and have achieved high effectiveness in semi-supervised classification scenarios.

Different from most classifiers, besides the input features, GCNs also require a graph for learning. Graphs have a wide range of applications, as they facilitate the identification and analysis of patterns and relationships among items. GCNs can provide significant results in semi-supervised classification scenarios since they can exploit both labeled and unlabeled data considering the graph structure and neighborhood information, which provide valuable information for learning.

These methods were originally proposed for datasets where the graph is readily available (e.g., citation networks) [12], [14]. However, for image datasets, in most cases, the graph needs to be computed. There are still not many approaches that use GCNs in image classification [15]–[17]. Therefore, proposing effective strategies to define and improve these graphs in image scenarios is crucial.

Since learning effective representations is at the core of many machine learning applications [18], different feature extractors have been proposed in the last decade, from handcrafted (e.g., color, texture, shape) to deep learning ones (e.g., CNNs and Vision Transformers). They can be used to compute a feature vector that represents the most relevant and discriminative information in an image. It is known that different extractors can provide complementary information [19], i.e., they can provide better results when combined in certain circumstances. However, how to combine different extractors is one of the key challenges. Strategies in the literature involve both early and late fusion approaches [19]. While early fusion combines features directly, late fusion involves combinations from higher-level structures, such as matrices, graphs, or ranked lists.

This work proposes an approach that combines different feature extractors considering the inherent properties of GCN models. Since these models receive two different inputs (e.g. a set of features and a graph), we use this characteristic to combine graphs and features from different extractors to leverage their complementarity. This work presents many contributions to the scenario of semi-supervised image classification using GCNs, among them, the key ones are:

- An investigation reveals that combining graphs and feature sets from different feature extractors can improve classification results by leveraging the complementary information of each extractor.
- Manifold learning re-ranking approaches can be used to improve the input graph of GCNs even further, especially when different graphs and features are used.
- The investigation also shows that the GCNs are more sensitive and there is a higher impact related to the input graph than the features, evincing that computing a highly effective graph is crucial in these applications.
- The experimental evaluation reveals better classification results in most cases, evincing the capacity of our approach to deal with scenarios where labeled data is limited.

The remainder of this paper is organized as follows: Section II presents the proposed approach. Section III reports and discusses the experimental results. Finally, Section IV concludes this work and discusses potential future work.

II. Proposed Approach

In this paper, we propose an approach and investigation that combines different feature extractors through graphs and feature sets using the inherent characteristics and properties of GCN models. The GCN models enable modeling information in terms of vector representations and graphs. Our proposed approach considers different methods for representations and graphs, aiming to combine complementary information. Since different extractors can encode various numerical characteristics from each image based on the technique and configuration parameters, we aim to combine the information from each extractor. By doing so, we intend to observe the behavior of the GCN and investigate how the accuracy is affected.

Additionally, we investigate the hypothesis that manifold learning methods can improve the effectiveness of the graph generated for semi-supervised classification tasks. Figure 1 illustrates the proposed method and its steps from A to E, which are described in the following subsections.

A. Feature Extraction

From the initial dataset, features are extracted using two extractors, 1 and 2. For each image in the dataset, the extractor processes the image and generates a feature vector encoding its relevant characteristics and patterns. A feature matrix is created for the set, where each row of the matrix is the feature vector of an image in the dataset. In this matrix, each element $f_{i,j}$ represents a numerical value, where row i represents an image and j represents a specific feature.

Among the great diversity of extractors available, this work considered three recent deep learning ones, contemplating both CNNs and ViT. The networks were all pre-trained in the ImageNet [20] dataset with features extracted from the last linear layer. The extraction models utilized in this work are described as follows:

- **ResNet [21]**: ResNet is a convolutional neural network that uses residual blocks. The activation functions for some layers are connected, and the model has shortcuts that allow some layers to be skipped. ResNet learns by combining these residual blocks. So, the network is adjusted only in the residual mapping [21]. This work considers the implementation with 152 layers.
- **Vision Transformer (ViT) [10]**: A Transformer pre-trained on large amounts of data and transferred to various image recognition benchmarks. First, the image is divided into patches, which are flattened and mapped so that they all have the same dimensions. Then, lower-dimensional linear embeddings are produced from the flattened patches. Positional embeddings are added, and the input sequence is fed as it would be for a standard transformer encoder. Finally, the model is pre-trained with image labels in a supervised manner using a very large dataset. Afterward, transfer learning is performed on the target dataset for feature extraction. The base version with a patch size equal to 16 was used in this work.
- **Shifted windows Transformer (Swin-tf) [11]**: It uses a Vision Transformer called Swin Transformer, which serves as a general-purpose backbone for computer vision. The Transformer is hierarchical, and its representation is calculated with **S**hifted **win**dows [11]. The shifted windows scheme allows to achieve great efficiency by limiting self-attention computation to non-overlapping local windows while allowing connections between windows. This hierarchical architecture is flexible for modeling at various scales and has linear computational complexity with respect to the image size.

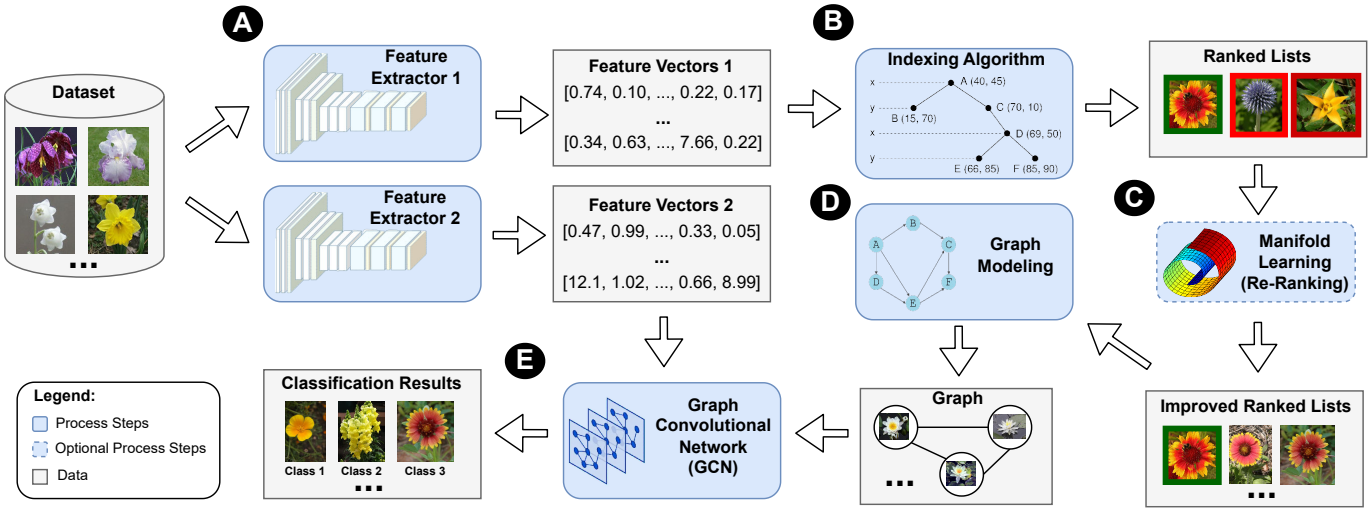

Fig. 1. Proposed method for combinations between different extractors using manifold learning.

B. Indexing Algorithm

The similarities between images are computed based on the features obtained in step A. For each feature extractor, a corresponding ranked list is generated. This can be efficiently performed by optimized indexing algorithms, such as the Ball-Tree [22]. There is a great variety of indexing approaches [23], and a discussion about them is beyond the scope of this work. For this work, this process is used to obtain a ranked list, which is a list that contains the nearest elements for each element in the set, ranked from closest to the most distant one (i.e., in descending order of similarity). This list is generated using the Euclidean distance, which calculates the distance between each image and every other image in the set.

C. Manifold Learning (Re-Ranking)

The ranked lists obtained in step B are processed by an unsupervised similarity learning method aimed at post-processing and improving the effectiveness of these lists. There are different methods in the literature that perform unsupervised similarity learning considering different strategies (e.g., graphs, diffusion processes, and others). Manifold learning approaches aim to capture and exploit the intrinsic manifold structure to compute a more effective distance/similarity measure [24]. In this work, we consider four unsupervised manifold learning methods to provide more effective similarity measures using rank-based formulations. They are all part of the Unsupervised Distance Learning Framework (UDLF) [25], which is an open-source framework that provides different approaches in this category:

- **Breadth-First Search Tree (BFSTree) [26]**: Utilizes a breadth-first tree to represent similarity information provided by ranking references. This tree is exploited to discover underlying similarity relationships, enabling a more effective similarity measure to be computed.
- **Correlation Graph (CORGRAPH) [27]**: The dataset structure is modeled as a graph, where nodes are images and correlations define the edges based on a threshold. The method analyzes the Strongly Connected Components (SCCs) of this graph. The manifold learning approach defines a more effective similarity between images, which is used to enhance the effectiveness of ranked lists.
- **Rank-Based Diffusion Process with Assured Convergence (RDPAC) [28]**: An efficient rank-based diffusion process combining diffusion approaches and ranking-based strategies. It approximates a diffusion process by leveraging rank-based information while ensuring convergence. It is a low asymptotic complexity algorithm and can be computed regionally.
- **Ranked Lists Similarity (RLSIM) [29]**: This method computes the correlation between ranked lists, which can be done by employing different rank correlation measures. The ranked lists are iteratively reordered according to the computed correlations.

D. Graph modeling

Such a step is essential since, in most image datasets, the graph is not readily available and an approach to generate it is crucial. Using the ranked lists obtained in step C, pre-processed by manifold learning approaches, we build a kNN graph to serve as input for the GCN. This kNN graph is constructed based on the ranked lists, where each image is represented as a node. The edges of each node connect to its top-k neighbors, which are the k most similar elements according to the ranked list.

E. Graph Convolutional Network (GCN)

A Graph Convolutional Network (GCN) [12] is a neural network classifier based on graphs, whose implementation is available in PyTorch Geometric [30][1] framework. It is a semi-supervised transductive classifier that enables learning from graph structures. Two different GCN models were considered:

[1]PyTorch Geometric: https://github.com/pyg-team/pytorch_geometric.

- **Graph Convolution Network (GCN)** [12]: The pioneering GCN that introduced the concept of applying convolutions to graph structures, commonly referred to as GCN-Net.
- **Approximate Personalized Propagation of Neural Predictions (APPNP)** [31]: A model that integrates a GCN with the PageRank algorithm, utilizing a propagation strategy derived from a modified PageRank approach.

In this work, the GCN model is trained considering the following inputs: *(i)* the set of feature vectors computed by extractor 1; and *(ii)* a graph computed on the ranked lists processed by manifold learning based on the features obtained from extractor 2.

III. Experimental Evaluation

The experimental protocol is divided into two subsections. While Section III-A presents the dataset, parameters, and other experimental settings; Section III-B reports and discusses the obtained results.

A. Experimental Protocol

In this work, we utilized the Oxford17Flowers [4] dataset, a widely used resource for various research scenarios. The dataset comprises 1,360 images categorized into 17 classes, with each class representing a distinct flower species. Each species is represented by 80 images, providing a balanced dataset for analysis.

The GCNs were trained considering a learning rate of 10^{-3} and 300 epochs with 500 neurons for each layer. For the manifold learning approaches, the default parameters of the UDLF framework[2] were used. The only parameter that was modified was the neighborhood size, defined by k. We set $k = 15$ in all cases, except for CORGRAPH [27], where $k = 40$ was used. The top-k neighbors for constructing the GCN kNN graph also used the same k value as the manifold learning approaches.

All the experiments employed a 10-fold cross-validation protocol, where each execution considered one fold as the training set and the others as the testing set, such that each fold is used as the training set at least once. This setting is important because it uses only 10% of the data for training and 90% for testing, creating a challenging scenario with limited labeled data and few labeled samples per class (i.e., 8 per class). In all cases, the average of executions is reported along with the standard deviation.

B. Results

The first set of experiments evaluated the impact of considering different extraction models for computing the set of features and the graph without manifold learning, which is step C in the diagram of the proposed approach. Table I presents the accuracy results in this scenario for both GCN-Net [12] and GCN-APPNP [31]. To assist the reader, we have standardized the highlighting of results as follows: the best result for each graph is marked in bold, results involving a combination of different extractors are highlighted with a gray background, and the best overall result in the table is colored in blue.

Notice that the best result of each graph consists of a combination of different extractors, except for Swin-tf [11]. The highest average accuracy in the table (highlighted in blue) is 87.20%, achieved by combining the ViT-b16 graph with features from Swin-tf. This demonstrates the potential of feature combination while also highlighting the distinct behavior of each extraction model.

TABLE I
Accuracy (%) results for GCN models combining features and graphs generated by different extractors.

GCN Model	Graph	Features	Accuracy (%)
GCN-NET	ViT-b16	ViT-b16	85.5662 ± 0.0989
		Swin-tf	**87.2059 ± 0.0735**
		ResNet	85.1324 ± 0.1995
	Swin-tf	ViT-b16	86.6691 ± 0.0739
		Swin-tf	**86.7353 ± 0.1364**
		ResNet	82.5000 ± 0.2568
	ResNet	ViT-b16	79.5809 ± 0.4068
		Swin-tf	**80.6397 ± 0.1233**
		ResNet	72.8015 ± 0.2894
GCN-APPNP	ViT-b16	ViT-b16	85.1471 ± 0.2712
		Swin-tf	**86.7353 ± 0.1324**
		ResNet	85.6838 ± 0.2960
	Swin-tf	ViT-b16	86.1765 ± 0.2790
		Swin-tf	**86.7574 ± 0.1250**
		ResNet	83.4559 ± 0.5453
	ResNet	ViT-b16	78.5882 ± 0.5187
		Swin-tf	**80.5882 ± 0.4674**
		ResNet	73.1691 ± 0.9906

Legend:
Bold: Best result for a given graph.
Gray Background: Best result in bold is a combination of graph and features from different extractors.
Blue: Best result in the table.

Given the potential of combinations, the same set of experiments was repeated but now considering the use of manifold learning (i.e., unsupervised similarity learning) approaches. Table II presents the results of experiments combining different extractors considering the GCN-NET network. The experiments revealed that manifold learning improved the results in all cases, except when ResNet [21] is used as a graph. In most cases, the best result is given by the combination of graphs and features from different extraction models.

It is also interesting to note that, despite the gains achieved through manifold learning, the optimal choice of manifold learning technique varies significantly depending on the specific combination of graph and features. There is no single method that consistently yields the best results in all cases, but all of them provide close results in most cases.

Similarly, Table III presents the results for the GCN-APPNP [31] model. In this case, manifold learning provided gains in all cases, for all features and graphs. Different from GCN-Net, the best result (colored in blue) consists of the combination of Swin-tf graph and ViT-b16 features. Also, notice that there are many results marked with gray background

[2]Version 1.60 of UDLF was used: https://github.com/UDLF/UDLF.

TABLE II
COMBINATIONS BETWEEN DIFFERENT EXTRACTORS, WITH DISTANCE MATRICES, WITH THE GCN-NET NETWORK.

Graph	Features	Method	Accuracy (%)
ViT-b16	ViT-b16	—	85.5662 ± 0.0989
	ViT-b16	BFSTREE	87.1544 ± 0.0574
	ViT-b16	RDPAC	87.1176 ± 0.0550
	ViT-b16	RLSIM	**87.5294 ± 0.0488**
	ViT-b16	CORGRAPH	86.9485 ± 0.1952
	Swin-tf	—	87.2059 ± 0.0735
	Swin-tf	BFSTREE	87.6103 ± 0.0753
	Swin-tf	RDPAC	87.5368 ± 0.0753
	Swin-tf	RLSIM	**87.6471 ± 0.0000**
	Swin-tf	CORGRAPH	87.2500 ± 0.0674
	ResNet	—	85.1324 ± 0.1995
	ResNet	BFSTREE	87.5515 ± 0.1678
	ResNet	RDPAC	**87.6471 ± 0.0658**
	ResNet	RLSIM	86.9412 ± 0.3679
	ResNet	CORGRAPH	87.2059 ± 0.1091
Swin-tf	ViT-b16	—	86.6691 ± 0.0739
	ViT-b16	BFSTREE	89.0588 ± 0.0720
	ViT-b16	RDPAC	**89.4338 ± 0.0574**
	ViT-b16	RLSIM	87.2132 ± 0.4178
	ViT-b16	CORGRAPH	88.3456 ± 0.1326
	Swin-tf	—	86.7353 ± 0.1364
	Swin-tf	BFSTREE	**89.7647 ± 0.0641**
	Swin-tf	RDPAC	89.7574 ± 0.0337
	Swin-tf	RLSIM	86.7500 ± 0.2151
	Swin-tf	CORGRAPH	87.8750 ± 0.1659
	ResNet	—	82.5000 ± 0.2568
	ResNet	BFSTREE	89.1544 ± 0.1000
	ResNet	RDPAC	**89.5000 ± 0.0441**
	ResNet	RLSIM	86.5882 ± 0.3650
	ResNet	CORGRAPH	87.9338 ± 0.2289
ResNet	ViT-b16	—	**79.5809 ± 0.4068**
	ViT-b16	BFSTREE	77.7353 ± 0.2606
	ViT-b16	RDPAC	79.3235 ± 0.3998
	ViT-b16	RLSIM	76.5074 ± 0.4618
	ViT-b16	CORGRAPH	77.2721 ± 0.2949
	Swin-tf	—	**80.6397 ± 0.1233**
	Swin-tf	BFSTREE	77.7868 ± 0.2740
	Swin-tf	RDPAC	77.7353 ± 0.1940
	Swin-tf	RLSIM	74.4265 ± 0.1995
	Swin-tf	CORGRAPH	75.8897 ± 0.3518
	ResNet	—	72.8015 ± 0.2894
	ResNet	BFSTREE	72.6912 ± 0.4676
	ResNet	RDPAC	**74.3603 ± 0.3603**
	ResNet	RLSIM	70.6397 ± 0.5706
	ResNet	CORGRAPH	74.1691 ± 0.2752

Legend:
Bold: Best result for a combination of graph and feature.
Gray Background: Best combination uses graph and features from different extractors.
Blue: Best result in the table.

TABLE III
COMBINATIONS BETWEEN DIFFERENT EXTRACTORS, WITH DISTANCE MATRICES, WITH THE GCN-APPNP NETWORK.

Graph	Features	Method	Accuracy (%)
ViT-b16	ViT-b16	—	85.1471 ± 0.2712
	ViT-b16	BFSTREE	87.4559 ± 0.1683
	ViT-b16	RDPAC	87.3824 ± 0.0750
	ViT-b16	RLSIM	**87.5809 ± 0.0768**
	ViT-b16	CORGRAPH	87.1029 ± 0.1618
	Swin-tf	—	86.7353 ± 0.1324
	Swin-tf	BFSTREE	87.9265 ± 0.0294
	Swin-tf	RDPAC	87.5221 ± 0.0471
	Swin-tf	RLSIM	87.7868 ± 0.0611
	Swin-tf	CORGRAPH	**88.3603 ± 0.0739**
	ResNet	—	85.6838 ± 0.2960
	ResNet	BFSTREE	**87.9559 ± 0.1348**
	ResNet	RDPAC	87.5588 ± 0.0720
	ResNet	RLSIM	87.6838 ± 0.0753
	ResNet	CORGRAPH	**87.9559 ± 0.1463**
Swin-tf	ViT-b16	—	86.1765 ± 0.2790
	ViT-b16	BFSTREE	**89.6691 ± 0.0945**
	ViT-b16	RDPAC	89.5000 ± 0.0441
	ViT-b16	RLSIM	88.9044 ± 0.0396
	ViT-b16	CORGRAPH	88.7647 ± 0.1029
	Swin-tf	—	86.7574 ± 0.1250
	Swin-tf	BFSTREE	89.4779 ± 0.0611
	Swin-tf	RDPAC	**89.5956 ± 0.0368**
	Swin-tf	RLSIM	88.2132 ± 0.3434
	Swin-tf	CORGRAPH	88.8750 ± 0.1042
	ResNet	—	83.4559 ± 0.5453
	ResNet	BFSTREE	89.3456 ± 0.0768
	ResNet	RDPAC	**89.4706 ± 0.2176**
	ResNet	RLSIM	87.2426 ± 0.1000
	ResNet	CORGRAPH	88.5735 ± 0.1923
ResNet	ViT-b16	—	78.5882 ± 0.5187
	ViT-b16	BFSTREE	81.0735 ± 0.5485
	ViT-b16	RDPAC	**82.4265 ± 0.4399**
	ViT-b16	RLSIM	79.8824 ± 0.2614
	ViT-b16	CORGRAPH	80.9118 ± 0.4641
	Swin-tf	—	80.5882 ± 0.4674
	Swin-tf	BFSTREE	81.7941 ± 0.1683
	Swin-tf	RDPAC	82.2941 ± 0.4091
	Swin-tf	RLSIM	78.6103 ± 0.3668
	Swin-tf	CORGRAPH	**83.4338 ± 0.2207**
	ResNet	—	73.1691 ± 0.9906
	ResNet	BFSTREE	75.0000 ± 0.5609
	ResNet	RDPAC	**75.1324 ± 0.5999**
	ResNet	RLSIM	73.0294 ± 1.0022
	ResNet	CORGRAPH	74.9265 ± 0.3049

Legend:
Bold: Best result for a combination of graph and feature.
Gray Background: Best combination uses graph and features from different extractors.
Blue: Best result in the table.

evincing that the combinations were effective in the majority of cases.

IV. CONCLUSION

This work proposed an approach and investigation into the behavior of GCN in semi-supervised learning scenarios, where different extractors are employed for features and graphs.

Combinations of different feature extractors, without the use of any manifold learning approach, revealed promising results when graphs and features from extractors with varying performances were combined. The use of feature matrices with better results can enhance the accuracy of matrices that provide complementary information. A clear example is the combination of ResNet and Swin-tf, where there is a great improvement in results, especially when manifold learning is used with GCN-APPNP.

The results demonstrate that the use of unsupervised manifold learning methods is consistently advantageous with the

application of GCN-APPNP. It can also be concluded that combinations of extractors are beneficial for those with lower performance. For instance, in the case of ResNet, it is advantageous to combine it with both Swin-tf and ViT-b16.

In this work, manifold learning was exploited to improve and pre-process the input graph of GCNs. In future work, we intend to investigate strategies to improve the feature representations. Besides that, we also plan to evaluate this approach for more datasets, extractors, and GCN models.

ACKNOWLEDGMENT

The authors are grateful to São Paulo Research Foundation - FAPESP (grants #2023/12736-3 and #2018/15597-6), Brazilian National Council for Scientific and Technological Development - CNPq (grants #313193/2023-1 and #422667/2021-8), and Petrobras (grant #2023/00095-3) for financial support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code: 88887.949418/2024-00.

REFERENCES

[1] S. Tripathi and C. R. King, "Contrastive learning: Big data foundations and applications," in *Proceedings of the 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD)*, ser. CODS-COMAD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 493–497.

[2] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. The MIT Press, 2006. [Online]. Available: http://dblp.uni-trier.de/db/books/collections/CSZ2006.html

[3] L. P. Valem, D. C. Guimarães Pedronette, and L. J. Latecki, "Graph convolutional networks based on manifold learning for semi-supervised image classification," *Computer Vision and Image Understanding*, vol. 227, p. 103618, 2023.

[4] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1447–1454.

[5] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *Workshop on Fine-Grained Visual Categorization, CVPR*, June 2011.

[6] A. Dehghan, S. Z. Masood, G. Shu, and E. G. Ortiz, "View independent vehicle make, model and color recognition using convolutional neural network," *CoRR*, vol. abs/1702.01721, 2017. [Online]. Available: http://arxiv.org/abs/1702.01721

[7] Y. A. Argyris, Z. Wang, Y. Kim, and Z. Yin, "The effects of visual congruence on increasing consumers' brand engagement: An empirical investigation of influencer marketing on instagram using deep-learning algorithms for automatic image classification," *Computers in Human Behavior*, vol. 112, p. 106443, 2020.

[8] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[9] F. He, T. Liu, and D. Tao, "Why resnet works? residuals generalize," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5349–5362, 2020.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *International Conference on Learning Representations*, 2021. [Online]. Available: https://arxiv.org/abs/2010.11929#

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *ICCV*, 2021.

[12] 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

[13] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6861–6871. [Online]. Available: https://proceedings.mlr.press/v97/wu19e.html

[14] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, Sep. 2008. [Online]. Available: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2157

[15] J. Rodrigues. and J. Carbonera., "Graph convolutional networks for image classification: Comparing approaches for building graphs from images," in *Proceedings of the 26th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, INSTICC. SciTePress, 2024, pp. 437–446.

[16] V. Vasudevan, M. Bassenne, M. T. Islam, and L. Xing, "Image classification using graph neural network and multiscale wavelet superpixels," *Pattern Recogn. Lett.*, vol. 166, no. C, p. 89–96, feb 2023. [Online]. Available: https://doi.org/10.1016/j.patrec.2023.01.003

[17] W. Tang, "Review of image classification algorithms based on graph convolutional networks," *EAI Endorsed Transactions on AI and Robotics*, vol. 2, no. 1, 7 2023.

[18] T. Uelwer, J. Robine, S. S. Wagner, M. Höftmann, E. Upschulte, S. Konietzny, M. Behrendt, and S. Harmeling, "A survey on self-supervised representation learning," *arXiv preprint arXiv:2308.11455*, 2023.

[19] L. Piras and G. Giacinto, "Information fusion in content based image retrieval: A comprehensive overview," *Information Fusion*, vol. 37, pp. 50–60, 2017.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, June 2016, pp. 770–778.

[22] S. M. Omohundro, "Five balltree construction algorithms," International Computer Science Institute, Tech. Rep., 1989.

[23] M. Wang, X. Xu, Q. Yue, and Y. Wang, "A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search," *Proc. VLDB Endow.*, vol. 14, no. 11, p. 1964–1978, jul 2021. [Online]. Available: https://doi.org/10.14778/3476249.3476255

[24] J. Jiang, B. Wang, and Z. Tu, "Unsupervised metric learning by self-smoothing operator," in *2011 International Conference on Computer Vision*, 2011, pp. 794–801.

[25] L. P. Valem and D. C. G. Pedronette, "An unsupervised distance learning framework for multimedia retrieval," *International Conference on Multimedia Retrieval (ICMR)*, 2017. [Online]. Available: https://github.com/UDLF/UDLF

[26] D. C. G. Pedronette, L. P. Valem, and R. da Silva Torres, "A bfs-tree of ranking references for unsupervised manifold learning," *Pattern Recognition*, vol. 111, 2021, 107666, ISSN 0031-3203.

[27] D. C. G. Pedronette and R. da S. Torres, "Unsupervised manifold learning by correlation graph and strongly connected components for image retrieval," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 1892–1896.

[28] D. C. Guimarães Pedronette, L. Pascotti Valem, and L. J. Latecki, "Efficient rank-based diffusion process with assured convergence," *Journal of Imaging*, vol. 7, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2313-433X/7/3/49

[29] D. C. G. Pedronette and R. da S. Torres, "Image re-ranking and rank aggregation based on similarity of ranked lists," *Pattern Recognition*, vol. 46, no. 8, pp. 2350–2360, 2013.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[31] J. Klicpera, A. Bojchevski, and S. Günnemann, "Personalized embedding propagation: Combining neural networks on graphs with personalized pagerank," *CoRR*, vol. abs/1810.05997, 2018. [Online]. Available: http://arxiv.org/abs/1810.05997