Hierarchical Graph Neural Networks Based on Multi-Scale Image Representations

João Pedro Oliveira Batisteli and Zenilton K. G. Patrocínio Jr

Laboratory of Image and Multimedia Data Science (IMScience)
Pontifical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte, Brazil
Email: joao.batisteli@sga.pucminas.br, zenilton@pucminas.br

Abstract—**Image representation as graphs can enhance the understanding of image semantics and facilitate multi-scale image representation. However, existing methods often overlook the significance of the relationship between elements at each scale or fail to encode the hierarchical relationship between graph elements. To cope with that, we introduce four novel approaches for graph construction from images. These approaches utilize hierarchical image segmentation techniques to generate segmentations at multiple scales, and one of them incorporates edges to encode the relationships at each scale. Leveraging these representations, we present two new models: the Hierarchical Graph Convolutional Network for Image Classification (HGCIC) and the Hierarchical Image Graph with Scale Importance (HIGSI). HGCIC uses an adaptive depth to capture significant features and patterns at different scales, while HIGSI employs a novel readout function that weighs the importance of each scale when generating a fixed-size graph representation. Experimental results with CIFAR-10 and STL-10 datasets show that the HIGSI model outperforms (or closely matches) state-of-the-art models. The model also utilizes smaller graphs, reaching the point of using graphs with 50% of the number of nodes compared to other approaches. Additionally, HIGSI outperforms models trained with only the base graph used to create the hierarchy, achieving up to 11.54% better performance while using fewer parameters.**

I. Introduction

The recent surge in popularity of Graph Neural Networks (GNNs) stems from their ability to effectively leverage the power of neural networks for processing data structured as graphs. Unlike traditional methods, GNNs effectively navigate the challenges of non-Euclidean data by leveraging the inherent structure of the graph. This powerful approach enables them to exploit information from both neighboring nodes and edges, capturing crucial local and global structural information.

The application of GNNs in the field of computer vision marks a significant advancement in processing and interpreting visual data. Graph-based image representation leverages the spatial relationships between image elements to model image content more effectively [1]. This representation can amplify the comprehension of image semantics and context by integrating domain-specific knowledge into the learning process [2].

While graph-based image representation offers numerous benefits, constructing graphs from images remains an active area of research. The most common approach involves image segmentation methods, which partition the image into distinct regions, each represented by a node in the graph [3]–[8]. Superpixel algorithms group pixels into perceptually meaningful regions, replacing the rigid grid structure [9]. This approach captures image redundancy, provides a convenient basis for computing image features, and significantly reduces the complexity of subsequent image-processing tasks.

However, this straightforward approach comes with its own set of challenges. For example, the success of a given method may heavily depend on the quality and quantity of regions generated by the segmentation process. Additionally, image segmentation methods usually produce flat representations of images, failing to capture the hierarchical relationships between image elements.

To overcome these limitations, this work proposes a novel approach that leverages hierarchical segmentation methods to generate nodes from a base graph, which is computed using superpixel methods. This approach yields a multi-scale graph-based representation of an image that captures intricate details often missed by other methods. Our approach uniquely generates a new set of nodes at different scales and defines relationships between them in four different adjacency setups using edges that connect nodes at various scales.

To assess the effectiveness of our hierarchical representation in image graph classification, we used CIFAR-10 and STL-10 datasets. This evaluation involved two novel models specifically designed to leverage the hierarchical structure: the Hierarchical Graph Convolutional Network for Image Classification (HGCIC) and the Hierarchical Image Graph with Scale Importance (HIGSI). Notably, HIGSI incorporates the newly proposed Region Graph Readout (RGR) function. This function assigns importance weights to graph features at each level of the hierarchy, enabling the model to effectively capture and integrate information across all scales before generating a fixed-size representation of the entire image graph.

This paper is structured as follows. The theoretical background is presented in Section II. Section III details the proposed representations and models, followed by the experimental setup outlined in Section IV. Section V presents and analyzes the results, offering insights into their significance. Lastly, Section VI draws some conclusions.

This work derives from João's M.Sc. Dissertation from PUC Minas, available at: https://bib.pucminas.br/acervo/562234

II. Graphs, Hierarchical Segmentation and GNNs

A graph $G = (V, E)$ contains a set of nodes $V = \{1, 2, ..., N\}$ and a set of edges $E \subseteq V \times V$, such that $(i, j) \in E$ denotes a directed edge from a node i to a node j.

Each node $i \in V$ is associated with a feature vector $h_i \in \mathbb{R}^{d_1}$, and optionally there may be edge feature vectors (termed as weights) $\{w_{ij} \in \mathbb{R}^{d_2} \mid (i, j) \in E\}$.

A. Hierarchical Segmentation

Hierarchical image segmentation is a set of image segmentations at different detail levels, in which the segmentations at coarser detail levels can be produced from simple merges of regions from segmentations at finer detail levels [10]. The resulting hierarchy of regions/segments can be represented with a tree structure, where the root node represents the entire image, and each subsequent level represents a finer detail level.

Hierarchical approaches must obey the two principles of multi-scale image analysis [11]: (i) the *causality principle* which defines that a contour presented at a scale k_1 should be present at any scale $k_2 < k_1$; and (ii) the *location principle*, which defines that contours should be stable, in the sense that they do neither move nor deform from one scale to another.

B. Graph Neural Networks

In recent years, GNNs became the standard for analyzing and learning from data in graphs [7]. In each layer of a GNN, the features of the nodes are updated by aggregating the features of their neighbors.

The input of each layer is the node features vector $\{h_i \in \mathbb{R}^d \mid i \in V\}$ and the set of edges E, the result of each layer is a new node's representation $\{h_i^{'} \in \mathbb{R}^{d^{'}} \mid i \in V\}$, where the same parametric function f_θ is applied to each node given its neighbors $\mathcal{N}_i = \{j \in V \mid (j, i) \in E)\}$, defined as $h_i^{'} = f_\theta(h_i, aggregate(\{h_j \mid j \in \mathcal{N}_i\}))$, in which f_θ is a parametric function of θ and $aggregate$ represents an aggregation function. One of the main differences between GNNs' architectures is the selection of parametric function f_θ and the aggregation function.

The Gated Graph Convolutional Network (GatedGCN) [12] combines the GCN and edge gating mechanism. The authors in [7] proposed enhancements to the GatedGCN architecture by incorporating residual connections and batch normalization. The updated node representation is computed following $h_i' = h_i + \mathrm{ReLU}(\mathrm{BN}(U^\ell h_i + \sum_{j \in \mathcal{N}_i} \alpha_{ij} \odot V^\ell h_j))$, in which U^ℓ and V^ℓ represent linear transformations, while the symbol $\odot$ denotes the Hadamard product. ReLU is an abbreviation for Rectified Linear Unit, and BN stands for batch normalization.

The edge gates, represented as α_{ij}, are defined by Equations 1 and 2,

$$\alpha_{ij} = \frac{\sigma(\widehat{w}_{ij})}{\sum_{j' \in \mathcal{N}_i} \sigma(\widehat{w}_{ij'}) + \varepsilon} \tag{1}$$

$$\widehat{w}_{ij} = A^\ell h_i^\ell + B^\ell h_j^\ell + C^\ell w_{ij} \tag{2}$$

in which A^ℓ, B^ℓ, and C^ℓ denote linear transformations, while σ represents the sigmoid function. The term ε is a small, fixed constant introduced for numerical stability. The edge gate, as defined by Equation 1, operates as a soft attention mechanism [7], enabling the model to learn the significance of different nodes within a neighborhood.

Lastly, the edge features are updated following $w_{ij}' = w_{ij} + \mathrm{ReLU}(\mathrm{BN}(\widehat{w}_{ij}))$, in which ReLU is an abbreviation for Rectified Linear Unit, and BN stands for batch normalization.

III. Hierarchical GNN's based on Multi-Scale Image Representations

A. Multi-Scale Image Representation Using Hierarchies

Let an image be represented by a RAG of superpixels $G = (V, E)$, in which the set of nodes V represents the collection of superpixels and the set of edges E describes the adjacency relation between the superpixels.

Let $\mathcal{H} = (\mathbf{P}_1, \ldots, \mathbf{P}_\ell)$ be the hierarchy derived from the graph G. Let $\mathcal{R}_j$ represent the set of regions in the partition $\mathbf{P}_j$ of the hierarchy $\mathcal{H}$. Consider that w_{ij} is the edge weight given by Euclidean distance of the average colors of superpixels i and j, chosen for its widespread use in the literature [6]–[8], ensuring comparability. Finally, let $\mathcal{R}$ be a set that consists of all regions belonging to all partitions $\mathbf{P}_j \in \mathcal{H}$.

By applying hierarchical segmentation to the base RAG, one can obtain a hierarchy of partitions at different scales, allowing for multi-scale representation and analysis. This multi-scale representation is useful in GNNs, as it allows for information sharing and message passing between different scales in the graph, enhancing the model's ability to capture complex patterns and structures within the graph, and leading to improved performance in tasks such as graph classification.

The selected adjacencies were chosen for their significant empirical gains. The hierarchy-based adjacency preserves the structure and relationships established by the hierarchy. The complete-based adjacency enables all vertices to share information during the message-passing stage. The kNN-based adjacency facilitates information sharing among similar vertices in the feature space, irrespective of their scale. Lastly, the HRG adjacency maintains the hierarchical structure and relationships while also allowing vertices at the same scale to exchange information.

The following subsections delve into the details of the four distinct graphs constructed from the hierarchy and explore the corresponding hierarchical GNN architectures.

B. Hierarchy-based Graph

This graph, denoted by $G_h = (V_h, E_h)$, is a graph computed from the hierarchical structure $\mathcal{H}$ in which the set of vertices V_h is equal to the $\mathcal{R}$. The set of edges E_h is defined by $E_h = \{(r_i, r_j), (r_j, r_i) \mid r_i, r_j \in \mathcal{R},\ r_i \neq\ r_j,$ in which r_j is the smallest region that contains $r_i\}$.

C. kNN-based Graph

This graph, denoted by $G_k = (V_k, E_k)$, is a graph computed from the k-nearest neighborhood of regions in the hierarchical structure $\mathcal{H}$ in the feature space, in which the set of vertices V_k is equal to the $\mathcal{R}$. Let $W(V) =$

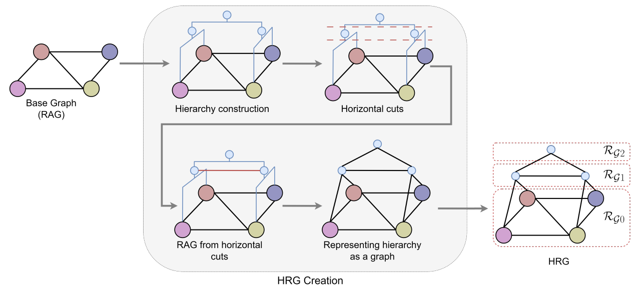

Base Graph
(RAG)
Hierarchy construction
Horizontal cuts
RAG from horizontal cuts
Representing hierarchy as a graph
HRG Creation
$\mathcal{R}_{\mathcal{G}2}$
$\mathcal{R}_{\mathcal{G}1}$
$\mathcal{R}_{\mathcal{G}0}$
HRG

Fig. 1. HRG construction pipeline. A hierarchical segmentation algorithm is applied to the base graph. Horizontal cuts are made in the hierarchy, and RAGs are created for every cut, forming Region graphs (RG). Ultimately, the hierarchy is represented as a Hierarchical Region Graph (HGR) which is used to train the HIGSI model.

$\{w(v),\ \forall v \in V\}$ be the set of feature vectors related to the vertex set V. The set of edges E_k is defined by $E_k = \{(r_i, r_j) \mid w(r_j)$ is one of the kNN of $w(r_i)\}$.

D. Complete-based Graph

This graph, denoted by $G_c = (V_c, E_c)$, is a graph computed from hierarchical structure $\mathcal{H}$, in which the set of vertices V_c is equal to the $\mathcal{R}$. The set of edges E_c is defined by $E_c = \{(r_i, r_j) \mid r_i, r_j \in \mathcal{R},\ r_i \neq\ r_j\}$.

E. Hierarchical Region Graph

Intending to extract pertinent information from various levels of the hierarchy, this work also introduces the HRG representation. Figure 1 illustrates the proposed method for computing the HRG representation from images.

Horizontal cuts $\mathcal{C}$ are made within the hierarchy $\mathcal{H}$ (usually, equivalent to the partitions of the hierarchy, so $\mathcal{C} = \Pi_{\mathcal{H}}$). For each cut, denoted as $c_j \in \mathcal{C}$, a RAG is computed with the hierarchy nodes in cut c_j that correspond to the set of regions $\mathcal{R}_j$ belonging to the partition $\mathbf{P}_j$ of the hierarchy $\mathcal{H}$. The RAGs computed from the cuts in the hierarchy and the base RAG are referred to as region graphs $\mathcal{R}_{\mathcal{G}_j}, j = 0, \ldots, |\mathcal{C}|$, with the base RAG denoted by $\mathcal{R}_{\mathcal{G}_0}$. Lastly, the edges of the hierarchy are also incorporated.

The spatial adjacency captured by the RAGs in different partitions (or scales) enables neighboring regions to exchange information during the GNN's message-passing step. Similarly, hierarchy adjacency allows information sharing across different scales.

F. Hierarchical Graph Convolutional Network for Image Classification Model – HGCIC

Figure 2 shows the Hierarchical Graph Convolutional Network for Image Classification (HGCIC) architecture. To embed the input edge and vertices features, two linear layers were applied to produce D-dimensional embeddings. The dimension of the edge and vertices embeddings remained the same across all layers. Inspired by [13], this model adopted $\mathcal{M} + 1$ convolutions, in which $\mathcal{M}$ is chosen at training and inference time. All $\mathcal{M}$ convolutional layers share the same weights, which improved the results and made the proposed architecture parameter-efficient [13].

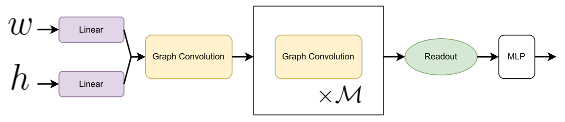


Fig. 2. The architecture of the HGCIC model comprises two Graph Convolutions, a Readout Function, and an MLP. The second convolution is applied $\mathcal{M}$ times.

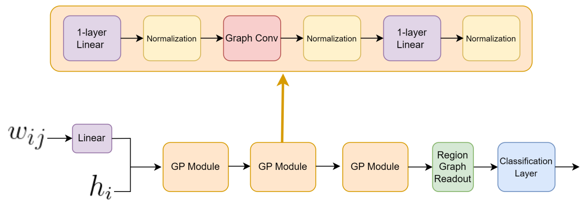


Fig. 3. The architecture of the HIGSI model comprises three GP modules, a Region Graph Readout, and an MLP. Each GP module is structured with a sequence of six components: a linear layer, layer normalization, graph convolution, another layer normalization, a second linear layer, and a final layer normalization.

An adaptive architecture can capture crucial features and patterns by dynamically adjusting its depth, leading to enhanced accuracy and performance. In this specific problem, the adaptive depth allows for efficient information propagation between distant nodes within the tree. This effectively utilizes the additional information encoded throughout the entire structure, leading to superior results.

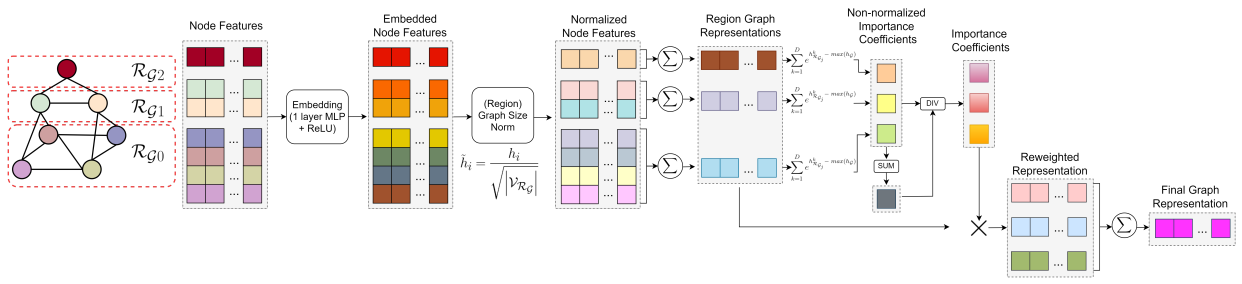
Following the graph convolution layers, a *readout layer* is employed to generate a fixed-size vector representation of the graph features. The output of the readout layer is then fed into a multi-layer perceptron (MLP) that learns to make class predictions based on the graph features.

G. Hierarchical Image Graph with Scale Importance Model – HIGSI

Figure 3 shows the Hierarchical Image Graph with Scale Importance (HIGSI) architecture. Inspired by 14, this model applies a linear and normalization layer before and after each graph convolution. This approach aims to project the node features into the same domain, thereby enhancing the diversity of the features, which helps to alleviate the over-smoothing problem. The sequence of layers (linear-norm-convolution-norm-linear-norm) is referred to as a Graph Processing Module (GP Module).

An additional linear layer is employed to embed the edge features in the same dimensional space as the node features. The transformation of edge features is solely handled by graph convolutions. The node and edge embeddings have the same dimension throughout all the graph convolutional and linear layers. After the GP modules, the proposed *Region Graph Readout* (RGR) is applied to generate fixed-size vectors that reflect the importance of each region graph $\mathcal{R}_{\mathcal{G}_j}$ for the task. The readout output is then fed into a linear (CL) to perform the classification.

Figure 4 illustrates the RGR scheme. After all GNN layers, a linear layer with the ReLU activation function embeds node



Node Features
Embedded Node Features
Normalized Node Features
Region Graph Representations
Non-normalized Importance Coefficients
Importance Coefficients
Reweighted Representation
Final Graph Representation
$\mathcal{R}_{\mathcal{G}2}$
$\mathcal{R}_{\mathcal{G}1}$
$\mathcal{R}_{\mathcal{G}0}$
Embedding (1 layer MLP + ReLU)
(Region) Graph Size Norm
$\tilde{h}_i = \frac{h_i}{\sqrt{|\mathcal{V}_{\mathcal{R}_\mathcal{G}}|}}$
$\sum_{k=1}^{D} e^{h^k_{\mathcal{R}_{\mathcal{G}_j}} - max(h_\mathcal{G})}$
DIV
SUM

Fig. 4. Illustration of the Region Graph Readout function steps: (i) embedding of node features; (ii) graph size normalization; (iii) generation of region graphs' representations; (iv) computation of importance coefficients; (v) reweighting of region graphs' representations; and (vi) calculation of the final graph representation.

features into a higher dimension. Then, the node features are normalized using the graph size norm [7]. This prevents large graphs from having an unbalanced influence on the coefficients since they have more vertices than the other graphs. The graph size norm is given by $\tilde{h}_i = h_i/\sqrt{|\mathcal{V}_{\mathcal{R}_{\mathcal{G}_j}}|}, \forall i \in \mathcal{V}_{\mathcal{R}_{\mathcal{G}_j}}$, in which h_i and $\tilde{h}_i$ are the features of node i before and after normalization, respectively, and $|\mathcal{V}_{\mathcal{R}_{\mathcal{G}_j}}|$ is the number of nodes in the region graph that i belongs to. To obtain a fixed-size vector for the region graph, denoted by $h_{\mathcal{R}_{\mathcal{G}_j}}$, one should sum up the features of all the nodes that belong to the same region graph, as described by $h_{\mathcal{R}_{\mathcal{G}_j}} = \sum_{i \in \mathcal{R}_{\mathcal{G}_j}} \tilde{h}_i, \ \forall j = 0, \ldots, |\mathcal{C}|$.

Drawing inspiration from the attention mechanism in transformers, the method computes importance coefficients as follows. Considering a D-dimensional region graph feature vector $h_{\mathcal{R}_{\mathcal{G}_j}} = (h^1_{\mathcal{R}_{\mathcal{G}_j}}, h^2_{\mathcal{R}_{\mathcal{G}_j}}, \ldots, h^D_{\mathcal{R}_{\mathcal{G}_j}})$, the non-normalized importance coefficient, denoted as $\alpha_{\mathcal{R}_{\mathcal{G}_j}}$, is obtained by taking the exponential of each component of $h_{\mathcal{R}_{\mathcal{G}_j}}$ subtracted by the maximum value over the features of the graph $max(h_\mathcal{G})$ (to avoid softmax overflow), and summing them all, defined by $\alpha_{\mathcal{R}_{\mathcal{G}_j}} = \sum_{k=1}^{D} e^{h^k_{\mathcal{R}_{\mathcal{G}_j}} - max(h_\mathcal{G})}, \ \forall j = 0, \ldots, |\mathcal{C}|$.

The importance coefficients are normalized by dividing each coefficient by the total sum of all coefficients within the same graph, i.e., $\alpha'_{\mathcal{R}_{\mathcal{G}_j}} = \alpha_{\mathcal{R}_{\mathcal{G}_j}} / \sum_{j=0}^{|\mathcal{C}|} \alpha_{\mathcal{R}_{\mathcal{G}_j}}, \forall j = 0, \ldots, |\mathcal{C}|$.

Reflecting the relevance of each region graph to the classification task, the importance coefficients are utilized to pool the corresponding region graph features. This results in a fixed-size vector $h_\mathcal{G}$ representing the entire graph, as expressed as $h_\mathcal{G} = \sum_{j=0}^{|\mathcal{C}|} \alpha'_{\mathcal{R}_{\mathcal{G}_j}} \times h_{\mathcal{R}_{\mathcal{G}_j}}$.

The graph feature $h_\mathcal{G}$ is the input for a classifier to generate the final prediction. This method can capture the diversity and significance of the region graphs in the hierarchical region graph HGR and produce a robust representation for classification.

The hierarchical GNN plays a central role in this process. It learns to extract relevant information from the hierarchy and generates the region graph representation, $h_{\mathcal{R}_\mathcal{G}}$. This $h_{\mathcal{R}_\mathcal{G}}$ captures the diversity and significance of the individual region graphs within the larger HGR structure. By understanding these relationships, the classifier can then leverage this robust representation created by the GNN to generate accurate final predictions.

In both models, a modified version of GatedGCN was adopted for graph convolution due to its ability to incorporate edge features into the message-passing step, allowing the model a more detailed understanding of the graph structure. The soft attention mechanism in Equation 2 enables the model to learn how important each neighbor $j \in \mathcal{N}_i$ is for the node i.

IV. Experimental Setup

A. Datasets and Baseline Model

The methods were evaluated using CIFAR-10 and STL-10 datasets. The CIFAR-10 dataset contains 60,000 32×32 color images across ten classes, split into 45,000 training, 5,000 validation, and 10,000 test images. The validation set was obtained by randomly sampling 5,000 images from the training set [7]. The STL-10 dataset contains 13,000 96×96 color images across ten classes, divided into 4,500 training, 500 validation, and 8,000 test images. The validation split was obtained using the same method used for the CIFAR-10 dataset.

A baseline model, named *Base RAG Model* (BRM), is established by training solely on the base RAG, which is used to derive the hierarchy. The performance of this baseline serves as a benchmark against which the effectiveness of the proposed methods can be measured. The baseline model architecture also includes three GP Modules. The adopted readout function combines global mean and max pooling. This readout function is applied after each GP module. Lastly, all the results of readout functions are summed and fed into an MLP consisting of two linear layers with layer normalization.

B. Graph Construction

Due to its simplicity, speed, and memory efficiency, the SLIC method [9] was chosen for superpixel segmentation. This choice also allows for a fair comparison since most of the works in comparative analysis utilize this method.

Superpixel segmentation quality can significantly impact the performance of computer vision tasks. To evaluate this impact, the study additionally employed the DISF method [15], which demonstrably outperforms the SLIC method in segmentation accuracy and optimal delineation.

TABLE I
HIGSI, BRM AND HGCIC MODELS RESULTS ON CIFAR-10 DATASET.

Model	# params	Superpixel Method	# nodes	# edges	Avg. Accuracy (± std dev.)	# Region Graphs
HGCIC$_h$	97,208	SLIC	47.13	92.26	0.6167(±0.007)	×
HGCIC$_c$	97,208	SLIC	47.13	2177.84	**0.6210(±0.004)**	×
HGCIC$_k$	97,208	SLIC	47.13	377.06	0.6125(±0.006)	×
BRM$_{20}$	128,394	SLIC	24.06	110.62	0.6472(±0.002)	×
HIGSI$_{20}$	111,562	SLIC	35.45	214.45	0.6474(±0.007)	5.3
BRM$_{40}$	128,394	SLIC	35.71	176.71	0.6539(±0.004)	×
HIGSI$_{40}$	111,562	SLIC	52.72	350.97	0.6642(±0.008)	6.38
BRM$_{60}$	128,394	SLIC	63.10	339.66	0.6612(±0.006)	×
HIGSI$_{60}$	111,562	SLIC	92.92	686.47	**0.6750(±0.003)**	8.27
BRM$_{80}$	128,394	SLIC	63.54	344.20	0.6611(±0.005)	×
HIGSI$_{80}$	111,562	SLIC	93.57	694.07	0.6728(±0.003)	8.3
BRM$_{20}$	128,394	DISF	20.00	97.13	0.6316(±0.004)	×
HIGSI$_{20}$	111,562	DISF	28.27	173.98	0.6450(±0.005)	4.55
BRM$_{40}$	128,394	DISF	40.00	221.62	0.6390(±0.002)	×
HIGSI$_{40}$	111,562	DISF	57.35	412.15	0.6623(±0.005)	6.36
BRM$_{60}$	111,562	DISF	60.00	352.10	0.6381(±0.009)	×
HIGSI$_{60}$	111,562	DISF	86.51	671.67	0.6649(±0.007)	7.73

1) Complete-based, kNN-based and Hierarchy-based Graphs: Experiments were only conducted using 20 superpixels. The hierarchical segmentation method employed was *Watershed by area* [16]. The number of nearest neighbors k was set to 8 on the kNN-based graph setup.

2) Baseline RAG and HRG: To create both the baseline RAG and the HRG, two superpixel segmentation methods, SLIC and DISF, were adopted and evaluated. Additionally, the *Watershed by area* [16] was employed for hierarchical segmentation in a canonical setup.

For the DISF method, 200 initial seeds were used on the CIFAR-10 dataset and 2,000 on the STL-10 dataset. Experiments targeted 20, 40, 60, and 80 superpixels for the CIFAR-10 dataset, while 50, 185, and 250 superpixels were the targets for the STL-10 dataset. It is important to note that the actual number of generated superpixels might vary depending on individual image characteristics when using the SLIC method.

C. Node and Edge Features

For graphs created from CIFAR-10 images, each node is characterized by the following set of features: (i) color; (ii) texture; (iii) region; (iv) X and Y mean position; and (v) node altitude in the segmentation tree. These features are concatenated into a feature vector $h_i \in \mathbb{R}^{104}$ for each $i \in V$. The same set is used in the base RAG setup, excluding the node altitude in the segmentation tree. Hence, each base RAG node is represented by $h_i \in \mathbb{R}^{103}$. The edge feature, represented by w_{ij}, is computed by the L2 norm of the difference in features between the nodes connected by the edge, i.e., $w_{ij} = \|h_i - h_j\|_2$.

The larger image size in the STL-10 dataset compared to CIFAR-10 allows for the extraction of more superpixels. However, this increased complexity sometimes makes simpler feature sets outperform more intricate ones. Therefore, we adopted a simpler approach, using only the mean color values and the mean X and Y positions, similar to the methods used in [3], [6], [8]. Thus, in the graphs derived from images belonging to the STL-10 dataset, each node $i \in V$ is represented by a feature vector $h_i \in \mathbb{R}^5$. The edge feature w_{ij} is computed using the same procedure used in the CIFAR-10 dataset.

D. Parameter Settings for GNN Training

The models trained for 300 epochs, starting with a learning rate of 10^{-3}. This rate gets halved if validation accuracy does not improve for 10 epochs, stopping at a minimum of 10^{-5}. The training used cross-entropy loss with the Adam optimizer and an L2 penalty to control overfitting. The model with the best validation performance was saved. To ensure reliable results, each experiment was run five times with different random seeds, and the reported accuracy is the average of these five runs.

V. QUANTITATIVE RESULTS

A. Results for HGCIC and HIGSI Model on CIFAR-10 Dataset

Table I presents the results for the HGCIC models, which were trained using three different graphs: kNN-based (HGCIC$_k$), hierarchy-based (HGCIC$_h$), and complete-based (HGCIC$_c$). It also includes the performance of HIGSI models and BRM variants for comparison. The subscript preceding each HIGSI and BRM model name indicates the target number of superpixels of the RAG used for training each BRM model or to create the HRG representation for training each HIGSI model.

Notably, the HGCIC$_h$ model, trained on the graph with hierarchical adjacency, outperforms the HGCIC$_k$ model despite having significantly fewer edges and achieves results comparable to the HGCIC$_c$ model. This underscores the importance of edge relationships in graph neural networks, which rely on these connections to extract structural information. While the complete adjacency approach has been explored in previous studies [17], it does not yield substantial improvements in model performance.

When using 20 superpixels, the HIGSI and BRM models exhibit nearly identical performance with the SLIC method, but a difference of 2.12% emerges with the DISF method. As the number of superpixels increases, the HIGSI model consistently outperforms the BRM model, regardless of the superpixel method used. This advantage is due to the proportional increase in nodes and region graphs generated by the hierarchy with more superpixels. The proposed RGR function in HIGSI models effectively manages this growth better than the BRM method.

Unlike the DISF method, the SLIC method tends to generate more superpixels than the target value. Given that images from the CIFAR-10 dataset are small and relatively simple, the superior segmentation quality of the DISF method does not have as significant an impact as the number of superpixels.

B. Comparison with State-of-the-Art Methods on CIFAR-10 Dataset

Table II presents the results for the best-performing HIGSI and HGCIC models, alongside the top BRM variant and some state-of-the-art methods on the CIFAR-10 dataset (for the complete version of this table and a detailed discussion, please refer to the dissertation text).

The GAT model [6] consists of three Graph Attention layers [19], a global sum readout function, and utilizes an RAG

TABLE II
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE CIFAR-10 DATASET. AN ASTERISK (*) INDICATES THE TARGET NUMBER OF NODES, AS THE AUTHORS DO NOT REPORT THE AVERAGE VALUE. CELLS MARKED WITH '—' DENOTE DATA THAT HAS NOT BEEN REPORTED. A QUESTION MARK (?) INDICATES THAT THE STANDARD DEVIATION WAS NOT REPORTED BY THE AUTHORS.

Model	# params	Superpixel Method	# nodes	# edges	Avg. Accuracy (± std dev.)	# Region Graphs
Proposed Spatial GNN						
$HGCIC_c$	97,208	SLIC	47.13	2177.84	0.621(±0.007)	×
BRM_{60}	128,394	SLIC	63.1	339.66	0.6612(±0.006)	×
$HIGSI_{60}$	111,562	SLIC	92.92	686.47	**0.6750(±0.003)**	8.27
Spatial GNN from the Literature						
GAT [6]	55,364	SLIC	75*	–	0.4593(±?)	×
GatedGCN [7]	104,357	SLIC	117.63	941.04	0.6731(±0.003)	×
ShapeGNN [18]	862,000	FH	70*	–	**0.8004(±?)**	×
Spectral GNN from the Literature						
H-L Cheby-net [3]	200,000	SLIC	253*	–	**0.7318(±0.005)**	×

derived from images for training. Although $HGCIC_c$ has only slightly more parameters than GAT [6], it achieves a significant 35.21% improvement in performance. Notably, it also utilizes the smallest graphs among the compared models, with the lowest average number of nodes.

While BRM_{60} utilizes more parameters than $HGCIC_c$, it strikes a better balance between graph size and performance. It achieves a respectable 6.47% performance improvement over $HGCIC_c$ despite employing smaller graphs with fewer nodes and edges. Notably, BRM_{60} achieves this without using the computationally expensive $\mathcal{M}$ convolutions applied in $HGCIC_c$, demonstrating its potential for efficiency.

In contrast to BRM_{60}, $HIGSI_{60}$ utilizes larger graphs due to its inherent hierarchical extraction method. However, it effectively leverages this hierarchy to outperform GatedGCN [7], despite having smaller graphs and slightly more parameters. GatedGCN [7] uses four GatedGCN layers [12], a global sum for readout, and single-scale RAGs for training. This demonstrates the advantage of using a hierarchical approach to extract meaningful information from larger graphs without suffering from computational inefficiency.

While methods like ShapeGNN [18] and H-L Cheby-net [3] have achieved superior results compared to the proposed models, they come with trade-offs. These include increased model complexity, the need for additional processing blocks, and the use of a spectral approach that requires the calculation of eigenvalues and eigenvectors of the graph's Laplacian matrix.

C. Results for HIGSI Model on STL-10 Dataset

Table III presents the performance of various HIGSI model configurations alongside BRM variants on the STL-10 dataset. Notably, no prior research has addressed image-graph classification on this specific dataset. The subscript preceding each model name indicates the target number of superpixels of the RAG used for training each BRM model or to create the HRG representation for training each HIGSI model.

The segmentation quality positively influences the performance on STL-10 images. Unlike the experiments conducted with the CIFAR-10 dataset, models trained on graphs generated from superpixels obtained via the DISF method outperformed those trained on graphs generated from superpixels obtained via the SLIC method. The superior segmentation achieved in the base RAG is propagated through the hierarchy, resulting in improved segmentation at all scales and region graphs.

TABLE III
HIGSI MODELS RESULTS ON STL-10 DATASET.

Model	# params	Superpixel Method	# nodes	# edges	Avg. Accuracy (± std dev.)	# Region Graphs
BRM_{50}	122.122	SLIC	48.46	242.02	0.4848(±0.010)	×
$HIGSI_{50}$	105,226	SLIC	71.25	491.24	0.5151(±0.009)	7.34
BRM_{185}	122.122	SLIC	193.91	1116.53	0.5303(±0.011)	×
$HIGSI_{185}$	105,226	SLIC	280.94	2368.54	0.5469(±0.008)	13.76
BRM_{250}	122.122	SLIC	253.75	1498.64	0.5337(±0.008)	×
$HIGSI_{250}$	105,226	SLIC	366.52	3197.53	0.5481(±0.012)	15.51
BRM_{50}	122.122	DISF	50.00	263.26	0.5024(±0.004)	×
$HIGSI_{50}$	105,226	DISF	69.77	491.40	0.5608(±0.006)	6.74
BRM_{185}	122.122	DISF	185.00	1124.98	0.5333(±0.012)	×
$HIGSI_{185}$	105,226	DISF	260.36	2231.07	**0.5847**(±0.009)	12.36
BRM_{250}	122.122	DISF	250.00	1557.88	0.5445(±0.006)	×
$HIGSI_{250}$	105,226	DISF	353.53	3150.68	0.5815(±0.004)	14.33

It is worth noting that utilizing the HRG representation and the HIGSI model resulted in significant improvements across all configurations compared to the equivalent baseline model. The performance boost was particularly noticeable when using the DISF method for superpixel generation. The $HIGSI_{50}$ model trained with DISF-based graphs outperformed all other baseline models trained exclusively with RAG, even those trained on graphs with more nodes and edges.

VI. CONCLUSION

This work introduces four distinct multi-scale image-graph representations. These representations are constructed by applying hierarchical segmentation techniques to a base graph of the image, which is a RAG of superpixels. In addition to the novel representations, two new models, HGCIC and HIGSI, were introduced. The latter utilizes the proposed *Region Graph Readout* function, a mechanism designed to reweight nodes based on the significance of their associated scale when obtaining a fixed-size vector from the graph.

Remarkably, HIGSI, trained with the proposed *Hierarchical Region Graph* representation, achieved superior or comparable results to other methods that employ spatial GNNs for the graph-image classification task, despite utilizing smaller graphs or having fewer model parameters.

VII. PUBLICATIONS AND AWARDS

The author's dissertation yielded two conference papers [20], [21] and an article accepted to a journal [22]. The article [21] was also nominated for the best paper of IEEE ISM 2023.

ACKNOWLEDGMENT

The authors thank the Pontifícia Universidade Católica de Minas Gerais – PUC-Minas, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – (Grant PROAP 88887.842889/2023-00 – PUC/MG, Grant PDPG 88887.708960/2022-00 – PUC/MG - Informática, and Finance Code 001), the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (Grants 303808/2018-7, 407242/2021-0, 306573/2022-9) and Fundação de Apoio à Pesquisa do Estado de Minas Gerais – FAPEMIG (Grant APQ-01079-23 and Grant PCE-00417-24).

References

[1] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3668–3678.

[2] W. Liang, Y. Jiang, and Z. Liu, "GraghVQA: Language-guided graph neural networks for graph-based visual question answering," in *Proceedings of the Third Workshop on Multimodal Artificial Intelligence*. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2021, pp. 79–86. [Online]. Available: https://www.aclweb.org/anthology/2021.maiworkshop-1.12

[3] B. Knyazev, X. Lin, M. Amer, and G. Taylor, "Image classification with hierarchical multigraph networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, K. Sidorov and Y. Hicks, Eds. BMVA Press, September 2019, pp. 223.1–223.13.

[4] S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, and J. Yang, "Multiscale dynamic graph convolutional network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 5, pp. 3162–3177, 2019.

[5] Q. Liu, L. Xiao, J. Yang, and Z. Wei, "Cnn-enhanced graph convolutional network with pixel-and superpixel-level feature fusion for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 10, pp. 8657–8671, 2020.

[6] P. H. Avelar, A. R. Tavares, T. L. da Silveira, C. R. Jung, and L. C. Lamb, "Superpixel image classification with graph attention networks," in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2020, pp. 203–209.

[7] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 43, pp. 1–48, 2023.

[8] V. Vasudevan, M. Bassenne, M. T. Islam, and L. Xing, "Image classification using graph neural network and multiscale wavelet superpixels," *Pattern Recognition Letters*, 2023.

[9] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[10] S. Guimarães, Y. Kenmochi, J. Cousty, Z. Patrocinio, and L. Najman, "Hierarchizing graph-based image segmentation algorithms relying on region dissimilarity," *Mathematical Morphology-Theory and Applications*, vol. 2, no. 1, pp. 55–75, 2017.

[11] L. Guigues, J. P. Cocquerez, and H. Le Men, "Scale-sets image analysis," *International Journal of Computer Vision*, vol. 68, no. 3, pp. 289–317, 2006.

[12] X. Bresson and T. Laurent, "Residual gated graph convnets," *arXiv preprint arXiv:1711.07553*, 2017.

[13] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković, "Principal neighbourhood aggregation for graph nets," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 260–13 271, 2020.

[14] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision gnn: An image is worth graph of nodes," *arXiv preprint arXiv:2206.00272*, 2022.

[15] F. C. Belém, S. J. F. Guimaraes, and A. X. Falcao, "Superpixel segmentation using dynamic and iterative spanning forest," *IEEE Signal Processing Letters*, vol. 27, pp. 1440–1444, 2020.

[16] J. Cousty and L. Najman, "Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts," in *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, 2011, pp. 272–283.

[17] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 501–14 515, 2022.

[18] R. A. Cosma, L. Knobel, P. van der Linden, D. M. Knigge, and E. J. Bekkers, "Geometric superpixel representations for efficient image classification with graph neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 109–118.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[20] J. P. O. Batisteli, S. J. F. Guimarães, and Z. K. G. Patrocínio, Jr, "Hierarchical graph convolutional networks for image classification," in *Brazilian Conference on Intelligent Systems – BRACIS'23*, 2023, pp. 63–76.

[21] J. P. O. Batisteli, S. J. F. Guimarães, and Z. K. G. Patrocínio, "Multiscale image graph representation: A novel gnn approach for image classification through scale importance estimation," in *2023 IEEE International Symposium on Multimedia (ISM)*, 2023, pp. 62–68.

[22] J. P. O. Batisteli, S. J. F. Guimarães, and Z. K. G. Patrocínio, Jr, "Hierarchical graph neural networks with scale-aware readout for image classification," *International Journal of Semantic Computing*, 2024.