Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Meta-Analysis on Image Classification using GNN

Meet Patel
Fairleigh Dickinson University
Daksh Khurana
Fairleigh Dickinson University
Laura Mancinelli
Fairleigh Dickinson University
Avimanyou Vatsa
avatsa@fdu.edu
Fairleigh Dickinson University

Research Article

Keywords: GNN, GAT, GCN, SLIC, QuickShift, KNN, RAG, Image, Classification

Posted Date: June 15th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9817473/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Meta-Analysis on Image Classification using GNN

Meet Patel[1†], Daksh Khurana[2†], Laura Mancinelli[3†], Avimanyou Vatsa[4*†]

[1*]Computer Science, Fairleigh Dickinson University, 1000 River Road, Teaneck, 07666, NJ, USA.

*Corresponding author(s). E-mail(s): avatsa@fdu.edu;
Contributing authors: m.patel6@student.fdu.edu; d.khurana@student.fdu.edu; l.mancinelli@student.fdu.edu;
†These authors contributed equally to this work.

Abstract

A fundamental problem in computer vision is the need to model complex spatial relationships, such as in medical imaging, fine-grained recognition, and remote sensing. Traditional deep learning models, specifically CNNs, for extracting visual features from grid-structured image data have achieved remarkable success. However, they often struggle to capture long-range dependencies and contextual relationships across distant regions, as they primarily focus on local pixel patterns within a filter. Graph Neural Networks, however, have emerged as a powerful tool for analyzing structured data and predictive modeling in massive multimodal Datasets. Representing images as graphs, where superpixels (groups of pixels) serve as nodes and their relationships as edges, enables GNNs to learn visual features and better capture these structural relationships. This study focuses on Graph construction methods, followed by different GNN models, and the pros, cons, similarities, differences, and other concerns of each GNN. It further analyzes widely used GNN architectures for vision tasks. Specifically in fine-grained classification, the graph-based and hybrid architectures achieved strong performance across experiments on benchmark datasets, including MNIST, CUB-200-2011, Stanford Dogs, and EuroSAT, and demonstrated the potential of GNNs for understanding advanced images.

Keywords: GNN, GAT, GCN, SLIC, QuickShift, KNN, RAG, Image, Classification

1 Introduction

A graph consists of a set of objects (called vertices or nodes) together with the connections (edges) between them. The various forms of graphs may include images, texts, and graphs where each superpixels represents a node and is connected to an adjacent pixel via an edge. Representing images as rectangular grids with image channels as arrays. However, texts are represented as a straightforward directed graph, where each word or entity is a node and the linguistic relationships are edges. Additionally, molecules are represented as graphs, with covalent bonds connecting atoms at the nodes [1, 2].

While GNNs are designed for Graph structured Data, using them on image data requires converting images into graph representations. It is an optimizable transformation of all graph attributes (nodes, edges, global context) that preserves graph permutation invariance. The focus must shift from pixels to graphs for GNNs to apply to work on image data. Furthermore, it is necessary to group pixels with similar color and texture into superpixels rather than assigning a color to each pixel as a node. This allows mapping millions of pixels to a smaller graph with a few hundred labeled Nodes, with neighboring super pixels linked by edges. GNNs have two primary operations: propagation and aggregation. Standard GNN operations can be applied to learn meaningful feature representations after the image is converted into a graph of superpixels. Each node's feature representations are updated via a weight matrix, W, in the propagation step, and over each node's neighborhood, a permutation-invariant function is applied in the aggregation step. Through neighborhood information routing, aggregation enables GNNs to learn relationships between examples. Isotropic neighborhood aggregation is used by early GNN variants, including Graph Convolutional Networks (GCNs). However, variants like GraphSAGE introduce learnable aggregation functions [3–6].

One of the main areas where graph-based modeling can provide significant benefits is Computer Vision, but it encounters several fundamental problems, particularly in fine-grained image classification, where identifying the same class across many variations, even with small differences, makes generalization difficult. There are some datasets that show this problem, such as the Caltech-UCSD Birds 200 (CUB-200) dataset. This dataset is curated especially for fine-grained problems. One of the most prominent examples of a problem discussed is that in this dataset, birds of the same species look very different if the photo/video capturing angle or environment is changed. [7]. CNN-based models are useful to extract subtle local features. They lack explicitly modeling relationships between spatially distant regions, which play an important role in fine-grained recognition. These problems can be solved using GNNs because

GNN models not only capture subtle local features but also understand how different regions relate to each other by modeling the graph, treating an image region as a node, and the relationships between them as edges. These tasks cannot be performed by traditional CNN models, which primarily capture local spatial patterns but cannot model long-range relationships between distant pixels. As all models must deal with trade-offs, GNN models also have to contend with high computational power when data is too complex, a heavy dependence on graph construction, and over-smoothing in deeper layers. Some authors also address these issues and propose GNN approaches, such as adding convolutional layers to GNN mechanisms, using attention to model how the graph assigns importance to particular image regions, and hybrid methods such as GNN–DNN.

2 GNN for Image Classification

To evaluate how Graph Neural Networks (GNNs) are applied to image classification tasks, the review employed a narrative meta-analysis. The aim is to examine representative architectural strategies utilized in graph-based vision models, with particular focus on graph construction pipelines, superpixel-based image representations, and hybrid deep learning architectures that combine convolutional neural networks with graph-based reasoning. The Identification of relevant resources applied done through searches in major academic databases, including Google Scholar, IEEE Xplore, and arXiv by employing a combination of keywords such as Graph Neural Networks, Graph Convolutional Networks, Graph Attention Networks, superpixel graphs, graph-based image classification, and vision graph neural networks, and through citation tracking of highly cited papers and recent survey articles in the field. Initially, it began by collecting a large set of publications on graph-based learning for visual recognition tasks via search. Then, after filtering out unrelated works and reviewing titles and abstracts, a significant set of studies was selected for full review. However, among this subset, six studies were selected for detailed review for their diverse representation of architectural methodologies, including convolution-based graph models, attention-based graph networks, hybrid CNN-GNN architectures, and spatiotemporal graph neural networks. Likewise, other cited works throughout the paper were used to provide supporting evidence, contextual discussion, and methodological relevance to the proposed review. Then, the necessary information was extracted from various experimental sections of the respective resources, and these attributes were used in the model architecture, the graph construction methodology, the datasets used for evaluation, and the reported performance statistics. Lastly, to support the meta-analysis across benchmark datasets

evaluating architectural design choices, graph construction techniques, and model performance, results were compiled into comparative tables [6].

2.1 Theoretical Foundations

To understand better, GNNs can be broadly categorized into Recurrent Graph Neural Networks (RecGNNs), Convolutional Graph Neural Networks (ConvGNNs), Graph Autoencoders (GAEs), and Spatial-Temporal Graph Neural Networks (STGNNs). Each GNN has its own characteristics and computational challenges [8]. These methods rely on "Graph Signal Processing (GSP)", where convolution is defined in the Fourier domain of graphs [8].

- `The Laplacian`: The base math object in this is the "Graph Laplacian (L)", which, in simple terms, is a matrix representation of a graph structure. Same as the Fourier basis in Euclidean signal processing, there is an orthogonal basis that is made up from eigenvectors of this Laplacian form [8].
- `The Process`: It includes steps in both the signal and spatial domains (node features). First, a signal in the graph (also represented as a node feature) is converted into a spectral domain by using eigenvectors of L, which are later multiplied by learnable filter weights and then transformed back into a spatial domain. Also, the Graph Convolution Network can be efficiently scalable, operating directly on graphs. This model can be used with a layer-wise propagation rule and a first-order approximation of localized spectral graph convolutions. [9], and it can also scale linearly, learn both layer representations that take encodings of local graph structure and node features. [9] Moreover, by using first-order approximations, GCN simplifies the spectral convolutions and directly functions in the spatial domain, avoiding eigen-decomposition [9].
- `Limitations`: Dynamicity is the key challenge in spectral methods because these methods are so tightly coupled that if a small node is changed in the graph, the Laplacian will change, the eigenvectors will change, and the learned filter will be invalid. Moreover, it will be computationally expensive ($\bigcirc(N^3)$ for eigen-decomposition) due to this transductive behavior [10], and Approaches such as matrix factorization or the original GCN-based approaches are mostly Transductive. The primary task of these is to optimize embeddings at all the nodes in a single fixed graph, and does not naturally generate unseen data. These methods also require additional training, such as new gradient-descent loops, to make accurate predictions on the new nodes. For instance, DeepWalk and node2vec4 are random-walk embedding methods [11].

2.2 The General Design Pipeline of GNNs

To better understand the standard workflow for most Graph Neural Networks, the key idea is to transform raw graph data into a useful representation through following modular steps [8].

- `Input and Feature Transformation`: The foundation is built on G = (V, E), with A (Adjacency matrix) representing which nodes are connected, and X (Feature matrix) representing attributes for each node. Moreover, some models include edge features, which can provide richer information about the graph [8]. Many surveys state that, during the preprocessing stage, temporal nuances can be preserved through track information such as edge order and timing, and this data can be useful [12].
- `The Graph Convolutional Layer`: At the core of GNN computation, a node updates its representations by collecting information from its neighbors. It works in three basic steps; the first is "Aggregation," which applies to a node v when the feature vectors are collected from its neighbors N(v). Second is "Transformation," which means all the aggregated features being passed from a learnable weight matrix W, and third is a very famous non-linear "Activation function" like ReLU. Also, all GNNs use neighbor aggregation in a recursive manner, known as message-passing rules [13]. The process basically includes steps such as Aggregation and Combination. By performing aggregation, each node takes feature vectors from its neighbors and aggregates them all. Moreover, in the Combination step, the nodes update their feature vectors by combining the aggregated neighborhood information with their previous vectors. [13] Additionally, the readout function is important for work at the final iteration of the entire graph and aggregates the node features. [13]. The final step of Layer Stacking is adding multiple layers, allowing any node to learn from another node farther away. These steps increase each node's receptive field, allowing more of the graph structure to be understood [8].
- `Graph Pooling (Down-sampling)`: MaxPooling is used in the graphs where sometimes graphs like predicting whether a molecule is toxic or not are too complex to understand, and for learning the whole graph (it is important to understand the whole graph in this kind of scenario), it shrinks the graph by using the Pooling method [8]. Basic standard GNN models are referred to as flats, such as GCN, GraphSAGE, and GAT. The one limitation is that there are "flat" models. [14] from their original graphs, they only generate information across edges and do not predict hierarchical structures such as object distinctions. [14]. It uses global pooling after applying all GNN layers to all nodes in tasks such as graph classification, summing

or averaging all node embeddings, and generating a graph-level representation. This approach dismisses any hierarchical structure from the graph. [14]
Function works as nodes to the super nodes and then changes the layer. Pooling methods first make clusters similar to nodes, then derive a term called a super node, which is basically a cluster itself, but is represented as a super node when fed into the next layer. The information hierarchy sets for local structured information at lower layers and global structured patterns at higher layers [8]. In DIFFPOOL, it essentially provides a set of differentiable soft clusters learned by DIFFPOOL, and these clusters form the coarsened input for the next GNN layer [14].

- `Readout (Global Aggregation)`: This stage defines how a model produces its final output. The model must convert all node-level embeddings into a single fixed-size vector for output, including the first aggregation step, which can be a sum over all features of a node or mean/average pooling [8]. This aggregation generates a simple one embedding that represents the entire graph [12]. We also called it a fingerprint of a particular graph [15]. Then, after it is usually given to a Multi-Level Perceptron to perform any regression or classification [8, 12].

3 Methodology: Graph Construction and Image-to-Graph Pipeline

Previous studies on GNNs have focused on classifying images using graphs constructed from regular grids of pixels or from similarly sized groups of pixels with similar structure and characteristics, called super pixels (compression). Super pixels significantly reduce the number of primitives required for subsequent processing steps by grouping apparently similar pixels into visually meaningful identities. Multiscale superpixel representations can be considered between regular grid representations and similarly sized superpixel representations for image classification [16]. A widely used and effective way to reduce the number of image primitives is to use superpixels. Most popular practices [17] [18], such as SLIC, Quick Shift, and Felzenszwalb, use it and achieve significantly better results with low computational complexity for GNNs. [18]. The Simple Linear Iterative Clustering (SLIC) is an iterative clustering method carried out in the 5D space of position and color information. The algorithm is widely used for color boundaries detection due to its speed, storage efficiency, and successful segmentation. However, it does not significantly reduce data for graph generation, as SLIC often captures the background [18]. Quick Shift is a 2D algorithm based on the kernelized mean-shift approximation.

It segments an image based on ϵ (the Gaussian standard deviation), α (color scheme weighting), and S (which limits the calculation to an $S \times S$ window), called the weighting of the color scheme, and S, called to limit the calculation to an $S \times S$ window size, therefore applied to a 5D space consisting of color information and image location, and belongs to the family of local mode-seeking algorithms. The benefit of Quick Shift is that it simultaneously computes hierarchical segmentations at multiple scales. It reduces the input data for graph generation by 30% and does not capture background pixels [18]. Felzenszwalb is a 2D image segmentation method that uses a single scale parameter to influence segment size. The actual number and size of segments, depending on local contrast, can greatly vary. Due to various requirements, such as a consistent static choice of parameters and parameters that require special adjustments, the results are unstable. Therefore, this segmentation on a series of images appeared to be less suitable in tests [18].

3.1 Generating the Graphs (RAGs and KNNG) - From Super pixels to Generating Graphs

The Region Adjacent Graph (RAG) is generated by adding an edge between every pair of directly adjacent super pixels and treating each super pixel as a node. Based on Aggregate information provided on characteristics of the super pixels themselves, each graph node can have associated features. One way to divide the image into homogeneous regions is QuickShift. During the segmentation stage, the obtained regions are represented as vertices, and edges represent the relationship between neighboring regions. The graph is used to limit the search to regions directly connected by the graph structure. RAGs have less computational complexity than the K-Nearest Neighbors Graph (KNNG) for subsequent processing. To achieve high classification accuracy, RAGs are fed into a graph CNN, and Quickshift outperforms Felzenszwalb and SLIC for graph generation. With the graph produced, the GNN model uses Neural Message Passing. Each node gathers information about itself from its neighboring nodes. This is equivalent to saying: "Tell me who your neighbors are, and I'll tell you who you are."

K-Nearest Neighbor (KNN) is another common method for building graphs from arbitrary data, in which nodes are defined over an initial feature space and connected by a distance metric [19]. There are three main limitations to using these methods. To begin with, using an initial feature representation, the graph structure is determined, which may be poor prior to the downstream task. In addition, due to the nature of the KNN selection rule, this process is nondifferential. Lastly, placing new nodes into graphs before a forward pass, due to the requirement to compute in order $\bigcirc(n * t)$,

where n is the number of training nodes, and t is the number of test nodes, Inductive inference does not scale. To solve these problems, researchers propose constructing a bipartite graph which is constructed between the training examples and a fixed proxy set containing learning vectors and uniformly chosen anchor examples from the training data. Thus, a graph is generated from class information, directly aligned with a downstream task rather than an initial feature representation, thereby improving classification accuracy by adding a simple module to the backbone CNN [19]. In [20], two KNN approaches are used to generate graphs, including: 1. K-Nearest Neighbors with spatial distance (KNN Spatial); and 2. K-Nearest Neighbors with combined spatial and color distances (KNN-Combined). The edges between each node and its K nearest neighbors, excluding the node itself, are established in the KNN-Spatial and KNN-Combined methods. The difference between the two lies in the distance function. KNN-Combined combines spatial distance with distance between average color values, where KNN spatial compares two superpixel geometric centroids [20]. Moreover, when k =1, both KNN-Spatial and KNN-Combined achieved the best performance. In the MNIST dataset, the best performance was achieved with k = 2 for KNN-Combined and k = 4 for KNN-Spatial. This further suggests that the GCN layers in the selected model are most helpful when information flows only through uniform regions, and is corroborated by the tendency of the KNN-Combined method to outperform both KNN-Spatial and RAG (again, with the exception of MNIST, in which the best results are achieved with RAGs and then KNN-Spatial). In discriminating between similar regions, the KNN-Combined method tends to be more effective. Since in RAGs each node can have a variable number of neighbors, the average degree and standard deviation for the RAGs generated to represent images in each dataset are shown [20].

Beyond the superpixel algorithm itself, the quality of graph construction is also heavily influenced by how superpixels are initialized and distributed across the image. Recent studies have shown that uniformly distributed seeds (used in methods such as SLIC) are particularly sensitive to noise and often fail to sufficiently represent small objects or thin regions [21]. To address this limitation, adaptive strategies have been developed to dynamically determine both the number and placement of initial seeds based on image characteristics. These characteristics include boundary gradients and local contrast. By allocating more seeds to regions with sharp gradient changes and fewer seeds to homogeneous areas, non-uniform initialization improves how well super pixels align with object boundaries, remain connected, and stay consistent, while keeping the computation efficient. This is especially helpful for graph-based learning

because higher-quality superpixels create more meaningful nodes and connections, which improve subsequent graph and GNN processing [21].

Despite the effectiveness of region-based graphs in reducing computational complexity and preserving local spatial structure, they remain fundamentally limited to pairwise relationships between nodes. Recent studies have shown that modeling images solely as simple graphs can lead to redundant connections and an inability to capture higher-order semantic relationships among image regions. To address this, Vision HyperGraph Neural Network (ViHGNN) extends the conventional Vision Graph Neural Network foundation by representing images as hypergraphs. ViHGNN allows hyper edges to connect to multiple regions or patches simultaneously [22]. With this design, hyper-edges are dynamically constructed and updated during training and inference using the Fuzzy C-Means clustering algorithm [22]. This allows the model to group visually and semantically related regions while maintaining a low computational overhead. This hypergraph structure learning module enables the network to uncover complex inter-region dependencies beyond pairwise adjacency, thereby improving representational power and achieving "state of the art performance" in image classification [22].

3.2 GNN Architectures for Image Classification

Techniques used in recent research based on how GNN's handle and represent relational data can be categorized into three classes. The first group includes convolution-based models such as Graph Convolutional Networks (GCNs), which let each node gather information from nearby nodes, like how convolution works in images. The second group includes attention-based models, such as Graph Attention Networks (GATs). These models use an attention mechanism to determine the importance of each neighboring node, so nodes can focus more on the most relevant neighbors when learning and passing information. The third group includes hybrid GNN–DNN models that combine graph-based reasoning with deep neural networks or transformers to better learn both features and relationships simultaneously. These hybrid models are gaining popularity because they combine the strengths of convolutional and attention methods while addressing challenges such as limited data and small class differences.

3.2.1 Graph Convolutional Network (GCN)

Fig, 1 illustrates one of the oldest and most widely used types of GNN models is convolution-based GNNs. Even though it handles graph-structured input, it still functions similarly to Convolutional Neural Networks (CNNs), which use standard image grids. Similar to how CNNs incorporate data from surrounding pixels, GCNs

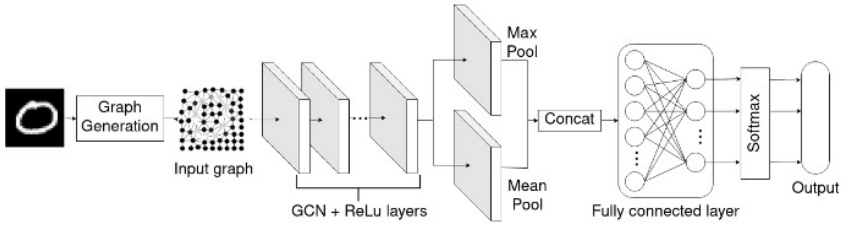


Fig. 1: Architecture of the model consisted of a group of sequential GCN layers [20].

incorporate the idea that each node collects information from its neighbors [23]. Graph Convolutional Networks (GCNs) were introduced in 2017, and a simplified method using transductive learning was presented shortly after, which operates by restricting the filters to a 1-step neighborhood around each node. It is a type of multi-layer convolutional neural network that operates directly on networks [9]. Each node in GCN represents an entity, and the relationships between these entities are represented by edges. The main objective of GCN is to learn node embeddings that are vector representations of nodes that capture the graph's feature and structural information. The learning process in GCN across the graph structure is based on the propagation of feature information. In GCN, supervised loss is denoted by l_0 with respect to the labeled part of the graph, $f(\cdot)$ is a neural network-like differentiable function used for learning node representation, the parameter λ is a weighing factor, X represents a feature matrix, and X_i is the feature vector of node i. Spectral graph theory uses a fundamental mathematical operator called the graph Laplacian. It enables signal-processing operations, such as graph convolution, by providing a matrix representation of the graph structure. The unnormalized graph Laplacian matrix of an undirected graph $G = (V, E)$ with N nodes $v_i \in V$, edges $(v_i, v_j) \in E$, an adjacency matrix $A \in \mathbb{R}^{N \times N}$ (binary or weighted) and a degree matrix

$$D_{ii} = \sum_j A_{ij}$$

Eqa. 1 is defined as

$$\Delta = D - A \tag{1}$$

And the formulation relies on the assumption that connected nodes in the graph are likely to share the same label.

This Laplacian matrix plays a key role in spectral graph convolution, where its eigenvectors form the graph Fourier basis [8]. However, the assumption might limit

modeling capacity, as graph edges do not necessarily represent node similarity and may contain additional information [9]. The Graph structure is encoded directly and trained on supervised targets for all labeled nodes using the $f(X, A)$ Neural Network model, thereby avoiding explicit graph-based regularization in the loss function. Adjacency matrix of the graph conditioning as $f(\cdot)$ allows the model to distribute gradient information from the supervised loss and enable it to learn the representation of nodes both with and without labels [9]. The layer propagation is defined as in Eqa. 2

$$H^{(l+1)} = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)}) \tag{2}$$

where node features matrix represented as $H^{(l)}$ and $W^{(l)}$ is trainable matrix of layer l. The activation function is denoted as $\sigma(\cdot)$, the adjacency matrix with added self-loops is defined as

$$\hat{A} = A + I$$

and $\hat{D}$ is the corresponding degree matrix [9]. Moreover, a two-layer GCN model is commonly used for node classification, where the final output is computed as in Eqa. 3

$$Z = \text{softmax}\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}\,\text{ReLU}\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}XW^{(0)}\right)W^{(1)}\right) \tag{3}$$

The first layer collects information from neighbouring nodes and applies non-linear functions such as ReLU. The second layer further generates the representations and applies a SoftMax function to produce class probabilities for each node [9]. GCN follows the message passing framework. Its ability to allow each node to update its information by averaging the information from its connected nodes helps it understand local structure and relationships within the data. In graph-based image classification, convolution-based GNNs can model how different parts of an image relate to each other, for example, how a bird's beak, wings, and feathers are connected. GCNs also served as the foundation for later models, such as attention-based and hybrid GNNs, because they introduced message passing, in which nodes share and update features through their connections to build stronger representations [23]. In 2025, research explained how information moves, combines, and is reduced in the graphs by utilizing the Transductive Propagation Network (TPN) to construct a similarity graph by embedding the entire support and query sets. Further labels are propagated from the support set to the query set, followed by applying transductive inference to enhance performance [24].

GCNNs are further categorized into two groups:

1. `Spectral-based GCNN`: The spectral-based model defines graph convolutions by introducing the filters from the perspective of graph signal processing. The core operator in spectral graph theory is the Laplacian. To perform spectral graph convolution, the following steps must be followed.

   - Compute the graph Laplacian, weighted adjacency matrix, and Laplacian matrix in normalized form.
   - Compute the Fourier functions, the Fourier Transform, and the inverse Fourier transform.
   - Then compute the spectral convolution. However, with a convolutional structure, several issues arise.

     – Suffers from $\bigcirc(n^3)$ time complexity, which is impractical for large-scale graphs because the eigenvector matrix U requires the explicit computation of the eigenvalue decomposition of the graph Laplacian matrix.
     – The time complexity of Spectral graph convolution is still $\bigcirc(n^3)$, though the eigenvectors can be precomputed.
     – In each layer, there are $\bigcirc(n)$ parameters to be learned.

   Besides, in the vertex domain, these non-parametric filters are not localized [25]. The ChebNet in the convolutional layer proposed the use of K-polynomial filters for localization and, in $\bigcirc(n)$ time, to compute fast spectral convolution; the compact kernels [25] must be defined. However, there are limitations with non-parametric filters, i.e., filters whose vector of Fourier coefficients is parameter $\theta \in \mathbb{R}^n$, such as the filter's learning complexity is in $\bigcirc(n)$, the dimensionality of the data, and they are not localized in space. The use of a Polynomial filter overcomes these issues, where $\theta \in \mathbb{R}^k$ is a vector of polynomial coefficients and can be expressed as a Chebyshev polynomial of order $K-1$ [18, 25].
2. `Spatial-based GCNNs`: Due to efficiency, flexibility, and generality, spatial-based convolutions have developed rapidly. From neighboring nodes in vertex domains, it propagates and aggregates the node representations. To perform spatial graph convolution, each input Superpixel/Node Embedding sends a message, and the messages are aggregated across all neighbors. To update node representations, the neighbor information is utilized. The prediction is based on the global readout of node embeddings [18].

   The model proposed in [20] comprises a sequence of Graph Convolutional Network (GCN) layers, each followed by a ReLU activation. Let the matrix of node embeddings produced by the final GCN layer be defined as

$$H \in \mathbb{R}^{N \times |F|}$$

where N is the number of nodes in the graph and $|F|$ denotes the number of features that characterize each node. The feature-wise maximum across all nodes is computed using the vector $r_{\max}$, while r_{mean} computes the feature-wise mean. Further, the resulting vectors are concatenated to form an equation as 4

$$r = [r_{\text{mean}} \,\|\, r_{\max}] \in \mathbb{R}^{2|F|} \tag{4}$$

Followed by softmax layer producing the final prediction

$$\hat{y} = \text{softmax}(z).$$

The representation, as in **??**, is passed through a fully connected layer with linear activation

$$z = Wr + b$$

.

The overall architecture of the model is illustrated in Fig. 1. Furthermore, to determine the number of GCN layers used in the model, the impact of varying the number of layers on the model's performance is evaluated while considering a fixed graph generation method [20].

3.3 Attention-Based Models (GATs)

Many problems, such as generating text, understanding context, and producing accurate outputs, are efficiently solved by Large Language Models using the attention mechanism. Moreover, in the series of Graph Neural Networks, Attention-Based Models (GATs) are used to address problems involving understanding background noise in visual data. By using trainable, nonlinear attention-gates, attention is applied to neural networks [27, 28]. In simple terms, it uses attention mechanisms to identify which superpixels are important and which are near the subject but are actually background noise. These outcomes lead to good computational efficiency, noise suppression, and overall sample efficiency in graph learning applications [27–29]. This theory has solid examples, such as one of its variants, the GGANet architecture, which was designed to isolate noise and subject in visual data by avoiding unnecessary pixel computation [30]. This superior work is done into two channels (i) Group Channel Attention (GCA) and (ii) Learnable

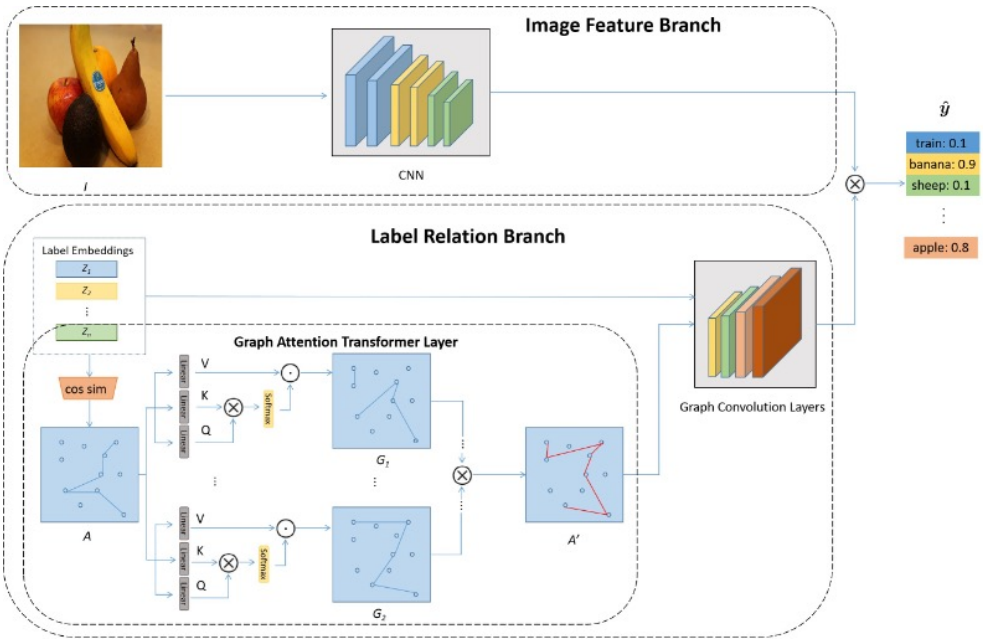


Fig. 2: First, the adjacency matrix for the label node embedding using cosine similarity is initialized in the label relation branch. Then, a graph attention transformer layer is adopted to mine subgraph structures [26].

Graph Attention (LGA), which provides attention factors to features and makes feature channels to feature vertices, respectively [30].

Attention mechanisms, beyond just noise filtering, also enable models to capture information at multiple spatial scales. Attention-based Graph Neural Networks are also significantly useful in extracting multi-level information. A model called MSPA-GAT is a great example of multi-scaling image processing using multi-scale pooling attention (MSPA) [31]. Because of this capability, images can be seen at different scales while maintaining highly focused details [31]. The attention architecture has several interesting properties: 1) As attention architectures are parallelizable across node neighbor pairs therefore it operation is efficient; 2) it can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbors; and 3) the model is directly applicable to inductive learning problems, including tasks where the model has to generalize to completely unseen graphs [10]. Although the attention-based GNNs allow for fixing major issues found in convolution-based models by dynamically focusing on informative neighbors [23]. They can be computationally expensive, especially in a few-shot setting where fast generalization is needed [32].

It is useful to analyze the structure of a graph's attention layer to better understand how GAT operates. The GAT starts with a single graph attentional layer, which is used throughout the GAT architecture. The input to the layer is a set of node features, where N is the number of nodes and F is the number of features in each node. A new set of node features (of potentially different cardinality F') is produced by the layer as the output [10]. To obtain sufficient expressive power to transform input features into higher-level features, at least one learnable linear transformation is required. A shared linear transformation parametrized by a weight matrix is applied to every node as an initial step. Then, self-attention is performed on the nodes by a shared attentional mechanism, which computes attention coefficients, indicating the importance of node j's features to node i. By performing the mask attention, the graph structure is injected into the mechanism. We only compute e_{ij} for nodes where $j \epsilon N_i$, where N_i is some neighborhood of node i in the graph. Further, the coefficients are normalized using the SoftMax function for all choices of j to make them easily comparable across different nodes. The attention mechanism is a single-layer feedforward neural network parameterized by a weight vector, with LeakyReLU nonlinearity (negative input slope = 0.2). Transposition is denoted T, and $\|$ is the concatenation operation. Also, a linear combination of features is computed for each node's final output using normalized attention coefficients. To stabilize the learning process of self-attention, the mechanism is extended to employ multi-head attention [33]. K independent attention heads are applied to stabilize and enrich the learning process, and their outputs are concatenated. Specifically, if the multi-head attention is performed on the final (prediction) layer of the network, instead of employing averaging, and the final nonlinearity is applied.

3.4 Hybrid GNN-DNN Models

Recent developments in computer vision suggest that structured feature extraction with relational and spatial data mapping can be more effective at producing more accurate, more human-like reasoning results. CNN or transformer encoders extract strong visual features, while GNN modules capture relational dependencies among image regions; therefore, hybrid architectures outperform pure GNN models. Simply, Hybrid models don't just extract features or weigh them; they reason about how features are connected and interact, and incorporate attention mechanisms to improve how relationships between features are modeled. For example, an attention-based reconstruction network proposed by [34] can capture relationships between spatial and semantic features, helping the model focus on subtle yet important visual differences. The article [20] indirectly supports hybrid models by showing how sensitive pure

GCNs are to graph construction quality. The authors explain that the inaccuracies or noise in a graph structure can significantly reduce performance. This highlights a key limitation of relying solely on graph convolutions and emphasizes the need to integrate a more hybrid solution. While hybrid models reduce the weaknesses of individual architectures, they introduce greater complexity by combining two already complex and troubled learning models. A prominent example of GNN hybrid models is the I2-HOFI (Interweaving Insights) approach. As it can overcall traditional GNN capabilities by separating two graphs, one is for inter-region capturing long-range dependencies, and the second one is for intra-region capturing finer details [35]. This new hybrid ability completely removes the need for bounding boxes; this algorithm uses the Approximate Personalized Propagation of Neural Predictions (APPNP) message-passing algorithm with an attention mechanism on top of residual connections [35].

One major finding showed a significant reduction in computational power by strictly focusing on task-relevant connections. This novel solution is provided by the FC-SBAAT framework, which applies a Top-t scarification mechanism directly to the Transformer's decoder [36]. This approach is a great example of developing a highly efficient solution using Transformers and GNNs, along with simultaneously improving semantic focus in few-shot image classification [36]. As a step forward, a model used a Vision Transformer (ViT) in Graph Neural Networks, achieving a state-of-the-art result in skin cancer classification by allowing ViT to capture global image dependencies and GNNs to explicitly learn spatial relationships [37]. It improves image feature extraction in diagnostically salient areas and outperforms advanced deep learning models by utilizing a Region-Adaptive Attention (RAA) mechanism. While surpassing advanced deep learning models, it serves crucial SHAP-enhanced interpretability for medical professionals [37]. As all these technologies show the peak of advanced computer vision's precision, a recent research paper introduces a highly complex multi-model and hybrid framework that uses all three architectures simultaneously. In this architecture, CNN extracts local pixel features, ViT models global visual dependencies, and GNN represents the final visual relations [38]. It uses self-supervised pretraining and GAN-based image augmentation, which together yield exceptional 99.33% accuracy while being computationally efficient for real-time edge deployment. While the hybrid multi-model excels in accuracy, another study showed that combining two components with distinct advantages into a single model is possible. The Feature Dual Reconstruction Network (FDRN) introduced by [34] comprises two main components that work together to improve feature learning and separation. The first part of their module combines spatial and channel attention and is called the "Mixed Residual Attention

(MRA) module". Spatial attention helps models identify where important details are in an image, while channel attention helps it determine which features (such as color or texture) matter most for classification. The second part, referred to as the Dual Reconstruction Feature Fusion (DRFF) module, rebuilds features from the spatial and semantic views. This work shows that attention mechanisms greatly improve a model's ability to focus on important visual details in fine-grained images. [34] provides a clear example with their FDRN model, which highlights key regions such as textures, patterns, and boundaries while reducing background noise.

4 Meta-Analysis and Performance Comparison on different Datasets

To understand how GNN models perform on different datasets, their capabilities, architectural benefits, and their impact on the computer vision industry, this section includes a meta-analysis of multiple papers. This meta-analysis uses several standard benchmarks, MNIST, CIFAR-10, and popular datasets, CUB-200-2011, Stanford Dogs, and Euro_SAT, which is exceedingly helpful for a comprehensive evaluation of Graph Neural Networks against traditional baselines. Here, CUB-200-2011 and Stanford Dogs are widely used for fine-grained visual categorization, and Euro_SAT is a dataset for geographical remote sensing.

Table 1 presents the categories and the methods and mechanisms used to apply them. Whereas Table 2 exemplifies the advantages and limitations of the methods. The model's performance metrics, including accuracy, across different datasets listed in the Table 3. However, the key structural innovations across the architectures of the listed models are demonstrated in Table 4. Additionally, Table 5 exhibits the advantages and associated limitations of the different data models. Finally, Table 6 compares different models/architectures on the same datasets.

5 Image Classification on Datasets: MNIST and CIFAR-10

For establishing baseline performance metrics, GNN often uses a standard low-resolution dataset. In [18] authors evaluates three models:

- ChebNet [25] is a spectral-based model, while
- EGCNN and
- GNNs are spatial-based models, based on image-based graph classification.

Table 1: Methods, Category, and Mechanism

Methods	Category	Mechanism
SLIC [39]	Superpixel Extraction	5D space iterative clustering (position + color)
Quick Shift [40]	Superpixel Extraction	Local mode-seeking (GSD[1] + color weight)
Felzenszwalb [18]	Superpixel Extraction	2D segmentation via single scale parameter
RAG (Region Adjacency) [41]	Graph Generation	Connects adjacent superpixels directly

[1]Gaussian standard deviation

Table 2: Methods, advantages, and limitations.

Methods	Advantages	Limitations
SLIC [39]	Speed, storage efficiency, successful segmentation	Fails to reduce enough data; often captures background
Quick Shift [40]	Computes hierarchical segmentations; reduces input data by 30%	Computationally heavier than simpler 2D methods
Felzenszwalb [42]	Adapts to local contrast	Unstable results; requires constant manual parameter adjustments
RAG (Region Adjacency) [41]	Lower computational complexity; limits search to connected regions	Relies entirely on the quality of the initial superpixel cut

The following datasets are used to evaluate these models.

- `MNIST`, a large database of handwritten digits from 0 to 9, consisting of 55000 training, 5000 validation, and 10000 test images, having only one-color channel with gray values and uniform scale of $28 \times 28 \times 1$ pixels.
- `CIFAR-10`, a large dataset of 60000 color images, each of which is exactly one of 10 classes: airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck, and consists of 45000 training, 5000 validation, and 10000 test images. The images are a uniform size of $32 \times 32 \times 3$ pixels and have three color channels, compared with the classical CNN model VGG-16.

Overall, the Spectral-based GNN, ChebNET [25], performed better than the Embedded GCN Model and the VGG-16 classical model. However, the GCN uses superpixel- or pixel-based RAG, while the Classical CNN uses the original images for classification.

Table 3: Models, Datasets, and performances for Meta-Analysis

Models & Architecture	Dataset	Accuracy / Performance Metrics
GPH, Backbone: ConvNextLarge [43]	Stanford Dogs, CUB-200-2011, NABirds	95.79% (Stanford Dogs), 87.80% (CUB-200) and 88.11% (NABirds)
GaRD, Backbone: ResNet-50 [44]	CUB-200-2011, FGVC-Aircraft, Stanford-Cars, NA-birds	89.6% (CUB-200)94.3% (FGVC-Aircraft)95.1% (Stanford-Cars)
UFENet, Backbone: ResNet-12 [45]	CUB-200-2011	84.90% (1-shot classification) and 94.09% (5-shot classification)
EIMGDNet, Backbone: Dual ResNet18 [46]	LEVIR-CD, DSIFN-CD, S2Looking, CLCD-CD, GVLM-CD	91.49% F1 / 82.93% IoU (LEVIR-CD), 77.32% F1 (DSIFN-CD), 49.19% IoU (S2Looking)
HGNN [47]	MNIST, Euro_SAT	98.7% on MNIST (using 75 nodes) and 98.4% on Euro_SAT (using 25 nodes)
Spatiotemporal GNN [48]	EuroSAT, Custom Sentinel-2 Spatiotemporal Dataset	96.69% Overall Accuracy (EuroSAT)92.51% and Overall Accuracy (Custom Sentinel-2)

Table 4: Method, Category and Mechanism

Model / Architecture	Architecture Type	Key Structural Innovation
GPH, Backbone: ConvNextLarge [43]	Hybrid GNN Plugin (DNN Encoder + GNN Encoder)	Take DNN feature vectors as node and make fully connected graph
GaRD, Backbone: ResNet-50 [44]	Spatial GNN / Feature-based Relational Learning	Use graph over tensor bank for semantic grouping to embed high dimensional features into a low dimensional manifold.
UFENet, Backbone: ResNet-12 [45]	CNN-based Feature Estimation Framework	Optimization of feature representation and reduce feature bias from input images.
EIMGDNet, Backbone: Dual ResNet18 [46]	Siamese Convolutional Network with Multi-Scale Fusion	Used Edge-Induced Group Enhancement Module which sharpen boundaries for spatial details.
HGNN [47]	Spatial GNN (Superpixel to RAG with Attention)	Converts superpixels into a Region Adjacency Graph (RAG).
Spatiotemporal GNN [48]	Spatiotemporal Graph Neural Network	Integrates spatial topological relationships with temporal sequencing to accurately map complex geographical boundaries.

Table 5: Models' advantages and limitations.

Model / Architecture	Advantage	Limitation
GPH, Backbone: ConvNextLarge [43]	Learns from entire image batches at once. Don't have to retrain the base model.	Computing data for massive batches causes significant memory overhead.
GaRD, Backbone: ResNet-50 [44]	Captures interactions of multiple features at once, which keeps the model stable.	Extremely computationally heavy.
UFENet, Backbone: ResNet-12 [45]	It heavily reduces bias when your training data is extremely scarce.	Built strictly for sparse data. Standard CNNs will easily outperform it on massive, fully-labeled datasets.
EIMGDNet, Backbone: Dual ResNet18 [46]	Uses multi scale fusion and edge sharpening to track shifting boundaries in satellite imagery.	Demands immense memory and compute power.
HGNN [47]	Delivers perfect accuracy on remote sensing data (like EuroSAT) with very little computation.	Struggles on complex, natural images because reducing the superpixels strips semantic detail.
Spatiotemporal GNN [48]	Merges spatial data and outperforms static, grid-based CNNs on geographical tasks.	Simple environmental noise (like heavy cloud cover) can break the whole graph.

Table 6: Comparison between CNN and GNN-based models on the same datasets

Model / Architecture	CUB-200-2011 Accuracy	Stanford Dogs Accuracy
ResNet-50 (Standard CNN Baseline) [49]	85.40%	80.00%
Standard GCN (Convolutional GNN) [9]	89.80%	Not Evaluated
GaRD (Attention-Based GNN) [44]	89.60%	Not Evaluated
GPH Plugin (Hybrid GNN) [43]	87.80%	95.79%
UFENet (Hybrid Few-Shot GNN) [45]	94.09%	89.52%

Nevertheless, ChebNET achieves higher accuracy than all other models. Spectral-based models performed better than spatial-based models and Classical CNN, with the additional perk of less computational cost [18].

In all WaveMesh experiments, the Graculus pool performs worse than no pooling/WavePool, indicating that, when learning with multiscale compression, a poor choice of pooling can reduce performance. Compared with SLIC, WaveMesh performed

better without pooling on CIFAR-10 and Fashion-MNIST, and in the presence of pooling, WaveMesh and wavepool are better or on par with SLIC and Graculus. As per ASA and EV metrics, Wave mesh is comparable with SLIC in ASA and inferior in EV.

6 Fine-Grained Image Classification (CUB & DOGS)

Fine-grained image classification is a problem that should be overcome after solving the classification of low-resolution visuals, and GNNs show exceptional performance here, too. Such a dataset, CUB-200-2011, with subtle inter-class differences, is tackled by the GaRD (Graph-based Relation Discovery) approach, which constructs high-order relational graphs rather than relying on simple pairwise connections. It groups semantic features to extract complex contextual details from regions of a bird's visual, yielding high accuracy on fine-grained benchmarks [44].

There is also evidence that UFENet-like ResNet-12 backbone models are effective for few-shot fine-grained tasks. The UFENet reached 84.9% and 94.09% for 1-shot and 5-shot classification, respectively, on the CUB-200-2011 dataset [45]. But performance can be improved by batch-level collective knowledge. Adding a Graph Neural Network (GNN) plugin (GPH) to the end of the process lets the GNN compute a graph of all photos and examine them all at once, which naturally forces the model to cluster similar photos. The GNN-based plugin achieves a 3% accuracy boost on the bird tests and a 4% boost on the Stanford Dogs (dogs) dataset. For the dog's dataset, it achieves an accuracy of 95.79%, the best score ever recorded on the same dataset worldwide.

7 Geographical and Remote Sensing Data (Euro_SAT)

Since the data is geographical and satellite-based, a Graph Neural Network can deliver its best results, as the data is heavily graph-based. GNNs can capture long-range dependencies while maintaining context, overcoming the grid-based limitations of traditional Convolutional Neural Networks. Moreover, in geographical data, GNNs can extract multi-scale spatial topological relationships, which CNNs can not solve [48].

The HGNN approach makes Region Adjacency Graphs utilizing superpixels and is evaluated on the Euro_SAT dataset. Furthermore, standard images like MNIST require more superpixels, whereas geographical remote sensing data are highly aggregated, so the model can fit aggregate points and boundaries into a single class and thus requires fewer superpixels [47]. Popular architectures like EIMGDNet use these findings and achieve excellent results on many high-resolution bi-temporal remote sensing

datasets, such as LEVIR-CD, DSIFN-CD, S2Looking, CLCD-CD, and GVLM-CD. It demonstrates great possible results in domain of geographical data [46].

8 Limitations of GNN

Making highly meaningful relationships by using Graph Neural Networks and training models, but there are some drawbacks to GNNs, too. The limitations are low construction quality graphs; computational power overheads are restricting deployment in the real world.

8.1 Graph Construction Dependency

There is a simple rule: if the graph is bad, the result will be bad as well. In technical terms, subsequent message passing layers propagate data through the graph. If you assume that raw data is very good at problem-solving satisfaction and, based on that raw data, a graph is constructed with bad quality, the subsequent layers will extract information from graphs and fail to evaluate true information. While scarification using hybrid frameworks improves efficiency, it risks creating sparse subgraphs with insufficient connections, leading to 30% more errors [50]. This means GNN models will be significantly worse when the environment is too noisy or incomplete. Moreover, how the graph is built in the first place also strongly affects the GCN performance [20]. The results show that changes in superpixel segmentation, node features, or edge similarity can significantly affect accuracy, indicating that GCNs are not very stable when graph quality varies [20].

8.2 Homophily Assumption and Heterophilic Graphs

Graph Neural Network models assume that connected nodes tend to share similar labels or attributes. In heterophilic graphs, where neighboring nodes belong to different classes, GNN architectures struggle to aggregate useful information. As a result, message passing may mix incompatible/ faulty node representations, significantly impacting classification performance. Graph Convolutional Networks rely to some extent on the Homophily assumption that connected nodes tend to share similar labels. Therefore, when graphs exhibit heterophily, neighborhood aggregation can mix incompatible nodes [51]. To improve GCN and address the limitations of the homophily assumption, a novel geometric aggregation scheme, Geom-GCN (Geometric Graph Convolutional Networks), was presented in 2020. This method aims to overcome the loss of neighborhood structural information and the lack of modeling long-range dependencies. Furthermore, it addresses

two fundamental weaknesses of MPNN aggregators that limit their ability to represent graph-structured data: the loss of structural information in node neighborhoods and the inability to capture long-range dependencies in disassortative graphs. Message-Passing Neural Networks (MPNNs) have been successfully applied to graph representation learning across many real-world applications. The aggregation scheme is permutation-invariant, with the basic idea that a graph can benefit from a continuous space underlying it. and consists of three modules: node embedding, structural neighborhood, and bi-level aggregation [52].

8.3 Computational Bottlenecks

A major bottleneck of vision-based GNNs is computational power, as static graph conversion often uses the K-Nearest Neighbors (KNN) operation, which is not an efficient choice, making standard GNNs hard to run locally on devices such as mobile phones and drones. To solve this problem and enable GNNs to run on edge devices, we need to limit unnecessary graph connections so that GNNs focus only on the parts of the graph that matter. This can be implemented by GreedyViG to introduce the Dynamic Axial Graph Construction (DAGC) architecture, which uses a dynamic mask computed from Euclidean distance. This method reduces GMACs (Multiply-Accumulate Operations) by 71.3% while maintaining state-of-the-art accuracy [53]. Another method, SageAttention, uses 8-bit quantization for the attention mechanism [54].

8.4 Scalability and Model Depth Challenges

Over the Cora, Citeseer, and Pubmed datasets [55] 5-fold cross-validation experiment using all labels showcased best result over 2-3 layer model but that for deeper models than 7 layers, the use of residual connections without training can become difficult, as for each node the effective context size by size of its K^{th} order neighbor with each its additional layer. Additionally, the dependence on the K^{th}-order neighborhood for a GCN with K layers and equal importance of self-connections vs. edges in terms of neighboring-node locality is implicitly assumed. however, For some datasets it might be beneficial to introduce a trade-off parameter λ in the definition of

$$\tilde{A} : \tilde{A} = A + \lambda I_N$$

and now this parameter plays a similar role as the trade-off parameter between supervised and unsupervised loss in the typical semi-supervised setting, can be learned via gradient descent. Furthermore, the number of parameters increases with model

depth, and the overfitting can become an issue [9]. Especially in few-shot settings where fast generalization is needed, attention mechanisms can be computationally expensive [32].

8.5 Impact on Humanity

While deep learning has the potential to make a strong and impactful presence in fields like medicine, it can also cause more harm than good. Utilizing these advances helps medical personnel by reducing human errors that the human eye could miss and reducing personal judgment. While it can be a very useful tool for the future, extensive research has shown that AI can learn and reinforce biases present in the data it is trained on. Medical imaging is one area of concern regarding the accuracy of computer vision applications compared to other areas of the medical field. For this reason, it is important that research is being done on how and where bias enters the process of using AI. One source of bias could come from a demographic imbalance in the data being provided to the training model. Imbalances in the training data, such as the overrepresentation of certain racial, ethnic, gender, age, or socioeconomic groups, can lead some models to perform well for some but result in incorrect and/or delayed diagnoses for others. Bias can also stem from the quality and the source of medical images. If a model becomes overly adapted to the distinct characteristics of the training data, it will perform poorly in new clinical environments [56]. To comprehend relationships and generate predictions, Graph Neural Networks (GNNs) combine data from a node and its neighbors. This can work well for many tasks, but it can also run into problems and inaccuracies when a network structure reflects real-world biases. In many graphs, such as social networks, nodes with similar sensitive characteristics tend to connect to each other. When GNNs aggregate information across neighboring nodes, they can begin to learn representations that strongly reflect these "sensitive traits". As a result, the model's predictions can become closely tied to sensitive attributes, leading to biased or unfair outcomes [57].

9 Conclusion

Graph-based representations provide a powerful alternative to traditional grid-based image modeling, enabling deep learning models to capture complex relationships among image regions. The findings indicate how Graph Neural Networks support image classification, centering on graph construction techniques, superpixel segmentation methods, and different GNN architectures. Additionally, GNN-based approaches showed a clear edge over traditional CNNs in tasks that require modeling long-range

spatial dependencies and contextual relationships, particularly in fine-grained recognition, remote sensing, and medically relevant imaging problems. The performance of GNN-based vision models largely depends on the quality of graph construction and superpixel selection, as well as the ability to capture both local and global relationships. Attention-based GNN models outperformed traditional GCN models because attention mechanisms assign higher weights to important nodes and relationships within the graph. Transformer attention outperforms GAT attention in terms of scalability. Furthermore, we analyzed three models, including Spectral-based GNN: ChebNET, which performed better than the Embedded GCN Model and the VGG-16 classical model. Although the GCN uses a superpixel- or pixel-based RAG, the Classical CNN uses the original images for classification. Nevertheless, ChebNET's accuracy is better than that of all other models. Spectral-based models performed better than spatial-based models and Classical CNNs, with the added perk of lower computational cost. Compared with SLIC, WaveMesh performed better without pooling on CIFAR-10 and Fashion-MNIST, and with pooling, WaveMesh and Wavepool are on par with SLIC and Graculus. By shifting our viewpoint from the grid mentality to the graph mentality, we must begin to observe relationships rather than fixed pixels and factors, including Superpixel choice, node definition, edge construction, and pooling strategy, all of which directly influence classification accuracy and computational cost. However, Deeper graph models may suffer from over-smoothing and scalability issues when used on large, noisy datasets, as poor graph construction can weaken message passing. Overall, GNNs are a very promising direction for future developments in next-generation image construction, particularly in applications where relationships among image regions carry important semantic meaning.

Acknowledgements. We gratefully acknowledge the support of the Research release time and other lab members and collaborators of the Deep Chain (DC) lab at the Olsen College of Engineering and Science (OCES), Fairleigh Dickinson University (FDU), Teaneck, New Jersey 07666.

Declarations

- Funding: There is no external funding supported this reseach.No external and internal funding supported this research and manuscript.
- Conflict of interest/Competing interests: The authors and co-authors have no conflict of interest. Authors contributed equally to this research article.
- Ethics approval and consent to participate: It agreeable to us.

- Consent for publication: It would our pleasure.
- Data availability: The dataset used is cited in the reference section. After acceptance, the GitHub URL will be shared with the Editor/publisher.
- Materials availability: It has been cited in the paper.
- Code availability: It will be shared after acceptance.
- Author contribution: authors contributed equally to this work.

References

[1] Khurana, D., Patel, M., Mancinelli, L., Vatsa, A.: Role of gnn in skin cancer classification. In: Proceding - 16th IEEE Integrated STEM Education Conference 2026 (2026). https://ieee-isec.info/day/1

[2] Khurana, D., Mittal, H., Joseph, O., Vatsa, A.: Empirical analysis of video segmentation for sustainability. In: Proceding - 16th IEEE Integrated STEM Education Conference 2026 (2026). https://ieee-isec.info/day/1

[3] Small, Q., Vatsa, A.K., Jan, T.R., Miller, A., Kumar, A.: Unsupervised gan for melanoma. In: 2022 IEEE Integrated STEM Education Conference (ISEC), pp. 357–357 (2022). https://doi.org/10.1109/ISEC54952.2022.10025272

[4] Miller, A., Jan, T.R., Small, Q., Kumar, A., Vatsa, A.K.: Gan assistance in diagnosis of melanoma. In: 2022 IEEE Integrated STEM Education Conference (ISEC), pp. 232–232 (2022). https://doi.org/10.1109/ISEC54952.2022.10025328

[5] Jan, T.R., Miller, A., Small, Q., Kumar, A., Vatsa, A.K.: Effect of cycle gan in melanoma classification. In: 2022 IEEE Integrated STEM Education Conference (ISEC), pp. 195–195 (2022). https://doi.org/10.1109/ISEC54952.2022.10025273

[6] Gautam, R.R., Vatsa, A.: Meta-analysis of image classification using graph neural networks (gnns). In: Proceding - 16th IEEE Integrated STEM Education Conference 2026 (2026). https://ieee-isec.info/day/1

[7] Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-ucsd birds 200 (2010)

[8] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2021) https://doi.org/10.1109/TNNLS.2020.2978386

[9] Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks (2017). https://arxiv.org/abs/1609.02907

[10] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks (2018). https://arxiv.org/abs/1710.10903

[11] Hamilton, W.L., Ying, R., Leskovec, J.: Inductive Representation Learning on Large Graphs (2018). https://arxiv.org/abs/1706.02216

[12] Shabani, N., Wu, J., Beheshti, A., Sheng, Q.Z., Foo, J., Haghighi, V., Hanif, A., Shahabikargar, M.: A Comprehensive Survey on Graph Summarization with Graph Neural Networks (2024). https://arxiv.org/abs/2302.06114

[13] Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How Powerful are Graph Neural Networks? (2019). https://arxiv.org/abs/1810.00826

[14] Ying, R., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical Graph Representation Learning with Differentiable Pooling (2019). https://arxiv.org/abs/1806.08804

[15] Cai, H., Zheng, V.W., Chang, K.C.-C.: A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications (2018). https://arxiv.org/abs/1709.07604

[16] Vasudevan, V., Bassenne, M., Islam, M.T., Xing, L.: Image classification using graph neural network and multiscale wavelet superpixels. CoRR **abs/2201.12633** (2022) 2201.12633

[17] Tochukwu, U.A., Maduabuchuku, A., Nwakeze, O.M., Akachukwu, O.C., Chinasa, I.S.: Ai-driven diagnostic imaging: Hybrid cnn-gnn models for early detection of cancer from pathological images. International Journal of Latest Technology in Engineering, Management & Applied Science **14**(9), 579–588 (2025) https://doi.org/None

[18] Nazir, U., Wang, H., Taj, M.: Survey of Image Based Graph Neural Networks (2021). https://arxiv.org/abs/2106.06307

[19] Trivedy, V., Latecki, L.J.: Cnn2graph: Building graphs for image classification. In: 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 1–11 (2023). https://doi.org/10.1109/WACV56688.2023.00009

[20] Rodrigues, J., Carbonera, J.L.: Graph convolutional networks for image classification: Comparing approaches for building graphs from images. In: International Conference on Enterprise Information Systems (2024). https://api.semanticscholar.org/CorpusID:269552841

[21] Xie, X., Fan, J., Xu, X., Xie, G.: Adaptive superpixel segmentation with non-uniform seed initialization. IEEE Transactions on Big Data **11**(2), 620–634 (2025) https://doi.org/10.1109/TBDATA.2024.3423719

[22] Han, Y., Wang, P., Kundu, S., Ding, Y., Wang, Z.: Vision hgnn: An image is more than a graph of nodes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 19878–19888 (2023)

[23] Mbiaya, F.A., Vrain, C., Ros, F., Dao, T.-B.-H., Lucas, Y.: Knowledge graph-based image classification. Data and Knowledge Engineering **151**, 102285 (2024) https://doi.org/10.1016/j.datak.2024.102285

[24] Lim, J.Y., Lim, K.M., Lee, C.P., Tan, Y.X.: A review of few-shot image classification: Approaches, datasets and research trends. Neurocomputing **649**, 130774 (2025) https://doi.org/10.1016/j.neucom.2025.130774

[25] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering (2017). https://arxiv.org/abs/1606.09375

[26] Yuan, J., Chen, S., Zhang, Y., Shi, Z., Geng, X., Fan, J., Rui, Y.: Graph Attention Transformer Network for Multi-Label Image Classification (2024). https://arxiv.org/abs/2203.04049

[27] Zhang, J., Shi, X., Xie, J., Ma, H., King, I., Yeung, D.-Y.: GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs (2018). https://arxiv.org/abs/1803.07294

[28] Li, S., Tang, M., Zhang, J., Jiang, L.: Attentive gated graph neural network for image scene graph generation. Symmetry **12**(4) (2020) https://doi.org/10.3390/sym12040511

[29] Chen, J., Seng, K.P., Ang, L.M., Smith, J., Xu, H.: Towards a gated graph

neural network with an attention mechanism for audio features with a situation awareness application. Electronics **14**(13) (2025) https://doi.org/10.3390/electronics14132621

[30] Guo, X., Gao, M., Zou, G., Bruno, A., Chehri, A., Jeon, G.: Object counting via group and graph attention network. IEEE Transactions on Neural Networks and Learning Systems **35**(9), 11884–11895 (2024) https://doi.org/10.1109/TNNLS.2023.3336894

[31] Tang, J., Miao, Y., Xia, Y., Zhou, Q., yi, C.: A multiscale pooling attention-based graph attention network for remaining useful life prediction. IEEE Transactions on Instrumentation and Measurement **PP**, 1–1 (2025) https://doi.org/10.1109/TIM.2025.3557109

[32] Ren, Malik: Learning a classification model for segmentation. In: Proceedings Ninth IEEE International Conference on Computer Vision, pp. 10–171 (2003). https://doi.org/10.1109/ICCV.2003.1238308

[33] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need (2023). https://arxiv.org/abs/1706.03762

[34] Liu, S., Zhong, W., Guo, F., Cong, J., Gu, B.: Fine-grained few-shot image classification based on feature dual reconstruction. Electronics **13**(14) (2024) https://doi.org/10.3390/electronics13142751

[35] Sikdar, A., Liu, Y., Kedarisetty, S., Zhao, Y., Ahmed, A., Behera, A.: Interweaving insights: High-order feature interaction for fine-grained visual recognition. International Journal of Computer Vision **133**(4), 1755–1779 (2025) https://doi.org/10.1007/s11263-024-02260-y . Epub 2024 Oct 20

[36] Wang, M., Yang, C., Sha, L., Li, J., Tang, S.: Fc-sbaat: A few-shot image classification approach based on feature collaboration and sparse bias-aware attention in transformers. Symmetry **18**, 95 (2026) https://doi.org/10.3390/sym18010095

[37] Dogga, A., R., S., S., S.: Hybrid vision transformer and graph neural network model with region-adaptive attention for enhanced skin cancer prediction. Scientific Reports **16** (2026) https://doi.org/10.1038/s41598-025-32502-z

[38] V, P., Karthik, A., Mynampati, S., L, P., Saraswathi, D.: A Hybrid CNN-ViT-GNN Framework with GAN-Based Augmentation for Intelligent Weed Detection in Precision Agriculture (2025). https://arxiv.org/abs/2511.15535

[39] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(11), 2274–2282 (2012) https://doi.org/10.1109/TPAMI.2012.120

[40] Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) Computer Vision – ECCV 2008. Lecture Notes in Computer Science, vol. 5305, pp. 705–718. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88693-8_52

[41] Tremeau, A., Colantoni, P.: Regions adjacency graph applied to color image segmentation. IEEE Transactions on Image Processing **9**(4), 735–744 (2000) https://doi.org/10.1109/83.841950

[42] Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision **59**(2), 167–181 (2004) https://doi.org/10.1023/B:VISI.0000022288.19776.77

[43] Bui, B.Q., Le, D.M., Tran, C., Tran, A.T., Pham, C.: Leveraging Graph Neural Networks to Boost Fine-Grained Image Classification (2024). https://openreview.net/forum?id=NJ6nyv3XWH

[44] Zhao, Y., Yan, K., Huang, F., Li, J.: Graph-based high-order relation discovery for fine-grained recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15079–15088 (2021)

[45] Wang, J., Lu, J., Yang, J., Wang, M., Zhang, W.: An unbiased feature estimation network for few-shot fine-grained image classification. Sensors **24**(23) (2024) https://doi.org/10.3390/s24237737

[46] Sun, L., Ding, M., Ye, Q., Zheng, Y., Wu, Z., Lu, W.: Eimgdnet: An edge-induced and multi-dimensional grouped difference network for remote sensing image change detection. Remote Sensing **18**(4) (2026) https://doi.org/10.3390/rs18040649

[47] Long, J., yan, Z., chen, H.: A graph neural network for superpixel image classification. Journal of Physics: Conference Series **1871**, 012071 (2021) https://doi.org/10.1088/1742-6596/1871/1/012071

[48] Domen, Mongus, D., Žalik, B., Lukač, N.: Graph neural network-based method of spatiotemporal land cover mapping using satellite imagery. Sensors **23**(14) (2023) https://doi.org/10.3390/s23146648

[49] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

[50] Samir, M., Fathy, N., Gad, W.: Knowledge-based question answering using graph neural networks and contextual language representations. Scientific Reports **16**(1), 2640 (2026) https://doi.org/10.1038/s41598-025-33854-2

[51] Jin, D., Yu, Z., Huo, C., Wang, R., Wang, X., He, D., Han, J.: Universal graph convolutional networks. In: Neural Information Processing Systems (2021). https://api.semanticscholar.org/CorpusID:247359006

[52] Pei, H., Wei, B., Chang, K.C.-C., Lei, Y., Yang, B.: Geom-GCN: Geometric Graph Convolutional Networks (2020). https://arxiv.org/abs/2002.05287

[53] Munir, M., Avery, W., Rahman, M.M., Marculescu, R.: GreedyViG: Dynamic Axial Graph Construction for Efficient Vision GNNs (2024). https://arxiv.org/abs/2405.06849

[54] Zhang, J., Wei, J., Zhang, P., Zhu, J., Chen, J.: Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In: International Conference on Learning Representations (ICLR) (2025). https://arxiv.org/abs/2410.02367

[55] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3), 93 (2008) https://doi.org/10.1609/aimag.v29i3.2157

[56] Koçak, B., Ponsiglione, A., Stanzione, A., Bluethgen, C., Santinha, J., Ugga, L., Huisman, M., Klontzas, M.E., Cannella, R., Cuocolo, R.: Bias in artificial intelligence for medical imaging: fundamentals, detection, avoidance, mitigation, challenges, ethics, and prospects. Diagnostic and Interventional Radiology **31**(2),

75–88 (2025) https://doi.org/10.4274/dir.2024.242854 . Epub 2024 Jul 2

[57] Dai, E., Zhao, T., Zhu, H., Xu, J., Guo, Z., Liu, H., Tang, J., Wang, S.: A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. Machine Intelligence Research **21**(6), 1011–1061 (2024) https://doi.org/10.1007/s11633-024-1510-8

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- snmathphysnum.bst
- snjnl.cls
- snbibliography.bib