POLITECNICO DI TORINO

Master's Degree in Data science and Engineering

Master's Degree Thesis

Vision Graph Neural Networks for Remote Sensing

Supervisors

Prof. Paolo GARZA

Dott. Luca COLOMBA

Candidate

Giovanni SCIORTINO

ACADEMIC YEAR 2022/2023

Abstract

Modern computer vision approaches mainly relied on convolutions neural networks, which view the images as regular grid structures. More recently, different approaches have been proposed to overcome the limitations, such as the lack of flexibility, and enhance the receptive fields of neural network architectures. To address these limitations, graph-based neural networks have garnered increasing interest for computer vision tasks. Instead of a grid, these methods represent images as graphs that encapsulate relationships between spatial regions. In the graph, nodes correspond to image patches or regions, while edges characterize the spatial and semantic connections between them. Consequently, this representation provides a more adaptable way of encoding both local and long-range dependencies within the visual scene. In this thesis, we investigate the application of Vision Graph Neural Network (ViG) architecture for multi-label land cover classification. Moreover, we evaluate different variation of ViG architecture, analyzing the effectiveness of different message passing layers compared to the original formulation. We utilize the large-scale BigEarthNet Sentinel-2 multispectral dataset, one of the largest existing remote sensing archives. Given the pyramidal architecture of ViG, we examine the performance of three graph convolutional layers: max-relative, GCN, and graph attention (GAT) convolution. We further compare the model with and without relative positional encoding and using all 12 Sentinel-2 spectral bands versus only the red-green-blue (RGB) bands. Experimental results demonstrate that Pyramid ViG provides superior performances over architectures like ResNet-101 in terms of precision, recall, and F1 score. Among the graph layers, max-relative convolution (i.e., the original formulation of ViG) performs best, and relative positional encoding improves predictions across all analyzed settings.

Acknowledgements

Questo percorso di 5 anni è giunto finalmente al termine. Ripensandoci non è stato per niente facile. Partendo dall'inizio, perchè è ripensando al pricipio che ognuno di noi si accorge di quanta strada ha fatto. Il primo anno, un anno di insidie e di difficoltà, di molti dubbi ma anche di altrettante soddisfazioni. Partendo da un test d'ingresso che una volta finito riporta una statistica legata il tempo medio per conseguire una laurea triennale in relazione alla fascia di punteggio ottenuta nel test. Abbastanza scoraggiante nel mio caso: 4 anni e mezzo. Un numero che probabilmente non molti ricordano, ma che io continuo a farlo a distanza di 5 anni.

Quindi partendo da "underdog" il primo giorno di lezione mi imbatto in Gioele. Una persona cinica, le cui ambizioni erano in linea con le mie e con cui ho condiviso tanto durante il primo anno. Dalle giornate di gennaio passate incessantemente a studiare e a fare i test di *Analisi I*; a quei pomeriggi sotto il sole perchè non trovavamo un posto in aula studio a maggio, giugno e luglio per preparare *Fisica I*; fino alle notti in videochiamata ad agosto per preparare *Algebra lineare*. Tutto ciò con la consapevolezza di non partire in vantaggio, ma anzi che era tutto da guadagnare. Ebbene Gio, ora mi sento di dire che non ce la siamo cavata male! Quindi grazie per i momenti di condivisione, sofferenze e difficoltà passate in quell'anno intenso e anche ai momenti di condivisione gli anni dopo nonostante i nostri percorsi diversi.

Il secondo anno, 2019/2020, un anno non facile soprattutto dovuto a motivi a noi conosciuti e che non vorremmo mai rivivere. Nonostante ciò, vorrei ringraziare di cuore una grandissima persona, Alberto, che durante sia il primo semestre in presenza che durante il secondo semestre da remoto, mi ha fatto tenere l'asticella in alto ragionando e confrontandoci nella preparazione dei vari esami.

Al terzo anno, un speciale ringraziamento a Chiara e alle ore in videochiamata durante i lockdown a studiare e preparare esami non banali. Un grazie anche a Giulio e ad Alessandro che in quel anno in particolare mi sono diventati sempre più amici semplicemente confrontandoci e scambiandoci idee che riguardassero l'università e non solo. Inoltre un grazie va anche a Lorenzo, Giovanni, Pasquale, Michele, Chiara, Lisa e ai due Andrea, con i quali ho condiviso un progetto di tesi triennale. È anche grazie al vostro contributo, con i ragionamenti e i confronti avuti

durante i meeting, che sono riuscito a finire il mio percorso triennale ed arrivare alla fine fin ad ora.

Per quanto riguarda questi ultimi due anni più recenti di magistrale, vorrei ringraziare di cuore in primis tutti i magnifici, chi più e chi meno, ragazzi del corso di *Data science* con cui ho avuto il piacere anche di intrattenere una conversazione in aula o anche un messaggio per via telematica per confrontarci su esami e argomenti dei corsi.
Ma partendo dal principio, un grazie enorme a Chiara, Mimmo e Bulfi per le intere giornate di studio durante la sessione del primo semestre del primo anno. Nel particolare, Mimmo e Bulfi grazie ancora per avermi fatto tenere l'asticella alta anche con un pizzico di arroganza nell'affrontare le sfide.
Un grazie al team per il corso di *MLDL*, con il quale siam riusciti a portare a casa in modo chirurgico un progetto non semplice durante il secondo semestre.
Un grazie anche ai ragazzi che durante il caldo di giugno e luglio ci son stati durante le giornate di studio in università e in aula Verdi.
Un grazie in particolare a Pietro che durante la sessione di settembre è stato sempre presente in aula studio per preparare assieme gli esami.
Un grazie di cuore anche a Pino per i momenti di studio e confronto sui progetti al primo semestre del secondo anno, a tutte le giornate di studio e ai weekend per preparare le materie.
Un grazie immenso anche anche ai ragazzi del progetto di *Applied Data Science Project*, Lorenzo, Giovanni, Valerio e Arcangelo, con i quali ho avuto condiviso un esperienza fantastica e li ringrazio per tutto quello fatto durante il progetto.
Un grazie anche ai ragazzi con cui ho studiato le materie date a maggio e a luglio non ancora citati, Jasmine e Sebastiano e soprattutto un grazie a Stefano per quella settimana memorabile a luglio fatta di stress, ma anche alla fine di soddisfazioni.

Un grazie al Professor Garza e a Luca per essere stati sempre disponibili fin dall'inizio per quanto riguarda il lavoro di tesi.

Un grazie vero a Federico, grande amico e coinquilino avuto durante il primo anno di magistrale con cui ho condiviso tanti momenti e discorsi che mi rimarranno sempre impressi.

Infine un ringraziamento speciale ai miei genitori, mio fratello e mia nonna che mi hanno sempre supportato dal principio a cui non sarò mai abbastanza grato per quello che sono per me e quello che hanno rappresentato fin ora in questo percorso.

Grazie ancora,
un grazie a chi c'è stato,
a chi c'è
e a chi ci sarà,
Giovanni.

Table of Contents

List of Tables

List of Figures

Acronyms

DNN
Deep Neural Network

RNN
Recurrent Neural Network

CNN
Convolutional Neural Network

GNN
Graph Neural Network

ViG
Vision Graph Neural Network

GCN
Graph Convolutional Network

GAT
Graph Attention Network

Chapter 1

Introduction

Land cover classification is a significant task in remote sensing, with important implications for applications as different as urban planning, environmental monitoring, and disaster management. This task involves the assignment of pre-defined categories or labels to specific land surface elements in an image, such as water, vegetation, or urban areas. One of the primary difficulties in this task is the complexity of multi-label classification, where each image patch can be associated with multiple class labels. Additionally, the high variability and heterogeneity of remote sensing images increases this complexity.

Traditional convolutional neural networks (CNNs), despite notable successes in image processing tasks, have limitations in effectively capturing irregular and complex objects as they view images as regular grid structures. As a result, a more flexible and expansive approach is needed for complex image recognition tasks like multi-label land cover classification. This encourages the investigation of graph-based neural networks, which have been gaining growing attention within the computer vision community.

Graph Neural Networks (GNNs) offer a high-performance framework for processing structured data. In regards to computer vision, GNNs enable images to be mapped as graphs, encapsulating the relationships between spatial regions. This approach provides a more adaptive way of encoding both local and long-range dependencies within the visual scene, potentially overcoming the limitations of traditional CNNs.

The Vision Graph Neural Network (ViG) is a specific type of GNN developed for computer vision tasks. The network maps images as graphs, with nodes corresponding to image patches or regions, and edges denoting the spatial and semantic relationships between them. The ViG architecture is composed of two basic modules: a Grapher module, which uses graph convolution to gather and manipulate graph information, and a Feed Forward Network (FFN) module, which uses linear layers to transform the features of the nodes.

The purpose of this research is to examine the viability of applying the ViG architecture for multi-label land cover classification. To evaluate different variations of the ViG architecture, we will use the large-scale BigEarthNet Sentinel-2 multispectral dataset and investigate the effectiveness of different message passing layers compared to the original formulation.

Following this introductory chapter, the thesis is structured as follows:

- Chapter 2 provides a literature review on related work in the field of computer vision, graph neural networks, and land cover classification.
- Chapter 3 describes the methodology, including the ViG architecture.
- Chapter 4 presents the dataset, and the evaluation metrics, the experimental setup and results.
- Chapter 5 concludes the thesis and summarizes the key findings.

Chapter 2

Related Work

This chapter offers a thorough introduction to important concepts and advancements in deep learning, specifically focusing on graph neural networks (GNNs) and their implementation in the field of computer vision. The chapter begins with an introduction to computer vision, outlining the main tasks and challenges faced by researchers in this rapidly evolving field. A systematic review of the relevant literature is presented, detailing the remarkable evolution and groundbreaking impact of deep learning on state-of-the-art computer vision models over the past decade. We focus in particular on the architectural innovations of residual networks, explaining how the introduction of residual blocks enabled the training of far deeper networks than previously thought possible. The subsequent section of this chapter delves into the central subject of graph neural networks and describes the principle of neural message passing as a distinguishing feature of GNNs, which enables them to model complex relationships within data. Then, we survey the increasing utilisation of GNNs in computer vision, highlighting their distinct advantages and proven efficacy in a variety of graph-based tasks. Finally, this paper offers a concise overview of the Sentinel-2 satellite and its spectral bands, providing a contextual understanding as we discuss the application of deep learning techniques, including GNNs, in the rapidly evolving field of remote sensing. The overall objective of this chapter is to offer an essential overview of key developments in deep learning, and in particular to explore the revolutionary role and potential of GNNs in shaping computer vision research and applications in the years ahead.

2.1 Computer vision

Computer vision is a special field of artificial intelligence that aims to enable computers to extract meaningful information from visual data, such as photographs and video recordings. This advanced technology allows computers to understand

and interpret their environment and make intelligent decisions. The central tenet of computer vision is to equip computers with the ability to observe and comprehend their surroundings via various sensory devices.

In this context, Figure 2.1 presents some of the most frequent tasks, and a brief description of each is given below:

- **Image and object classification** - This involves the classification of images or three-dimensional objects based on their visually distinctive features. This is a key step in object detection and classification systems. The process typically entails deriving features from the image or object and subsequently classifying it into one of several predefined classes. This task is essential in several areas, such as medical imaging, autonomous driving, and facial recognition.[1]

- **Object detection** - This task involves determining the presence of specific objects in images and locating them with bounding boxes. It is a more advanced task than image classification since it involves not only categorizing the objects but also identifying their location within the image. This is particularly valuable in systems such as surveillance, self-driving cars, and image retrieval.[2]

- **Pose estimation** - The task aims to detect the position and orientation of a person or object, with a specific focus on human pose estimation. This involves predicting the locations of particular body parts such as the elbows and hands. This is highly valuable in applications like augmented reality, where digital objects need to interact convincingly with the real world, and in motion capture for gaming or film production.[3]

- **Image and video generation** - It involves generating new synthetic images or videos by using existing data. This task has various applications in creative domains such as digital art and animation. Generative Adversarial Networks (GANs), a type of deep learning model, have achieved notable success in this field by producing authentic and high-quality media. [4]

- **Denoising** - This procedure involves eliminating extraneous signals from visual data, such as images and videos, to restore the original, undamaged data. It is a necessary process in image restoration and preprocessing, employed in diverse disciplines including medical imaging, astrophotography, and film restoration. [5]

- **Activity recognition** - This task entails the identification of particular movements or actions within video sequences, serving as a valuable method in surveillance, sports analysis, human-computer interaction, and healthcare settings, including patient monitoring and rehabilitation. [6]

- **Semantic segmentation** - This process entails labeling each pixel in an image with a class designation that corresponds to different semantic categories of objects. Such an approach provides a pixel-level, dense understanding of a scene, which is critical for autonomous driving, aerial image analysis, and medical imaging. [7]
- **Instance segmentation** - This task detects and delimits individual object instances, providing precise masks rather than just bounding boxes. This is essential in contexts where distinguished identification of object instances is imperative, such as medical imaging to isolate overlapping cells, or autonomous vehicles to detect pedestrians. [8]
- **Panoptic segmentation** - This is a wide-ranging task that combines semantic and instance segmentation, assigning both class labels and individual identities to each pixel in an image. Such an approach provides a holistic understanding of a scene, which is especially critical in domains like robotics and autonomous driving, where a meticulous comprehension of the surroundings is crucial. [9]

In this work the main topic is image classification, as in particular multi-label classification (as will be discussed in Chapter 4). Given an input image fed to the network, the latter have to associate it into one or more categories.

2.2 Deep learning in computer vision

The area of deep learning, and in particular its application to computer vision, has undergone a significant transformation in recent years. A continuous wave of innovative ideas has propelled the field to make rapid strides. A pivotal force behind this evolution has been the acceptance and honing of Convolutional Neural Networks (CNNs) [11].

Since the introduction of LeNet [12], CNNs have been used as the fundamental structure for many successful visual applications. These applications cover a broad spectrum of tasks, such as image classification, object detection, and semantic segmentation.

Over the last two decades, the architecture of CNNs has evolved at an unprecedented rate. An example of this evolution is the emergence of Residual Networks (ResNets) [13]. The inclusion of skip connections in ResNets enabled the training of much deeper networks. This breakthrough addressed the persistent issues of vanishing and exploding gradients that often inhibit the training of deep networks.

The field of deep learning witnessed progress with the advent of MobileNets [2], which introduced optimised lightweight models suitable for mobile devices. By employing depthwise separable convolutions, these models achieved an intricate balance between computational efficiency and performance, thereby reducing the

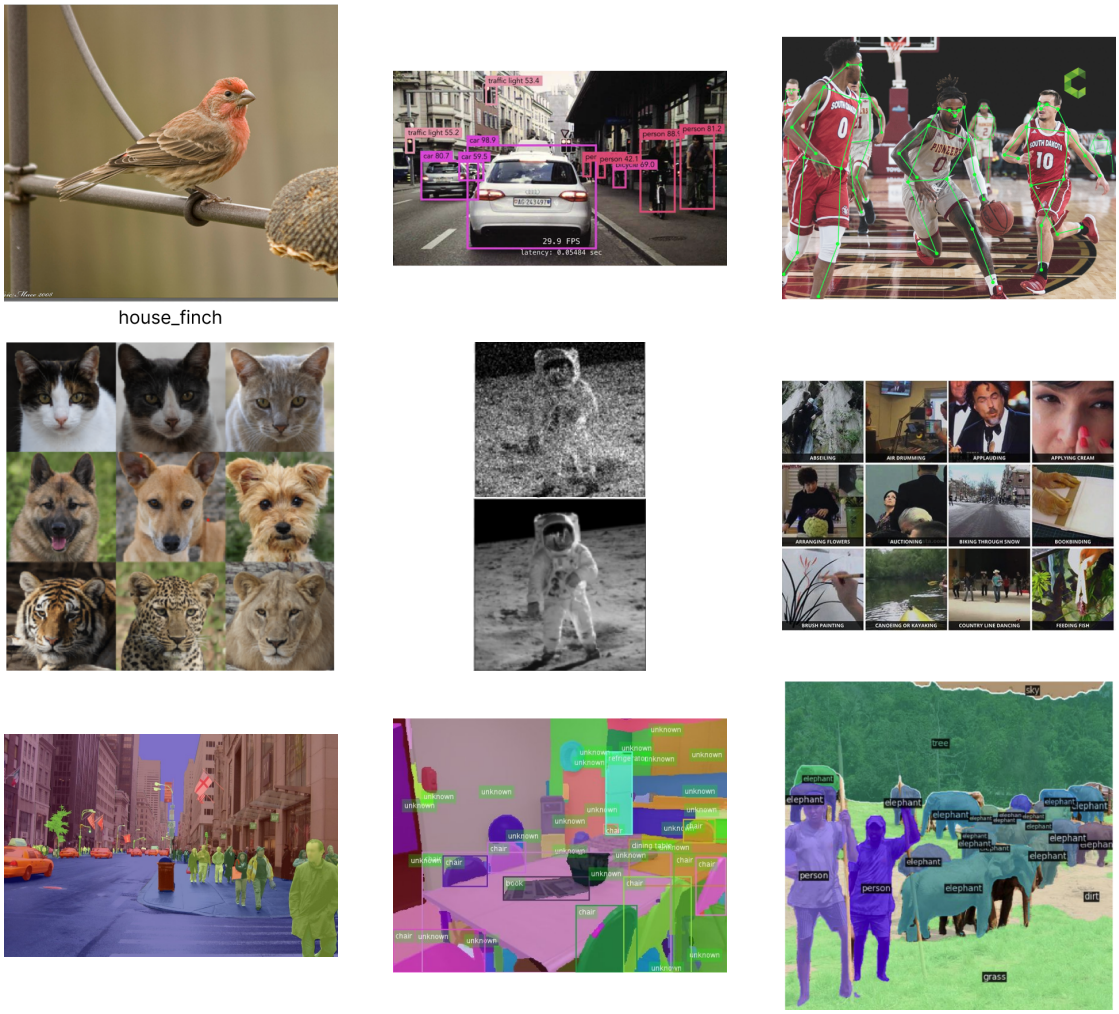

Figure 2.1: Some images related to the following tasks (from left to right and from top to bottom): Image and object classification [1], object detection [2], pose estimation [3], image and video generation [4], image denoising [5], activity recognition [6], semantic segmentation [10], instance segmentation [8], panoptic segmentation [9]

need for computational resources. This development facilitated the use of deep learning models for on-device applications.

The emergence of vision transformers (ViTs) in 2020 marked a significant architectural shift in the field [14]. ViTs adapted the transformer architecture, originally designed for natural language processing, to handle computer vision tasks. As a result, numerous ViT variants have since been developed, each seeking to improve performance through innovative techniques. These included approaches

such as pyramidal hierarchies [15], local attention mechanisms, and improved position encodings to capture spatial image structure more effectively.

MLPs gained attention in computer vision following the success of ViTs [16]. When equipped with specially designed modules [17] [18] [19], MLPs have shown potential for competitive performance in a variety of visual tasks. These range from traditional challenges such as detection and segmentation to more complex problems. This demonstrates the versatility and adaptability of MLP architectures for a wide range of computer vision applications.

2.3 ResNet

Deep residual networks, also known as ResNet, were initially presented by Kaiming He et al. [13] at the 2015 Conference on Computer Vision and Pattern Recognition (CVPR).

Before the introduction of ResNets, due to the challenging nature of the problems and real-world tasks thrown at deep neural networks, it was widely believed that the size of the networks would have to increase if one wanted to achieve high levels of accuracy in these tasks. Without considerable depth, the network could not extract numerous patterns or features for acquiring complex and meaningful input data representations.

The most apparent solution to address this issue was to expand the network's depth to the maximum extent possible, given the resources available for training. Consequently, the authors of the ResNets paper raised a crucial query: "Can one learn superior networks by merely stacking more layers?"

As previously mentioned, according to theoretical principles, an increase in the number of layers within a regular neural network should facilitate the creation of more complex representations, thereby enhancing the learning process and resulting in higher accuracy. However, experiments have shown that as network depth increases, accuracy saturates and then rapidly degrades, but surprisingly this degradation is not caused by overfitting, as shown in the Figure 2.2.

To tackle the issue, which has been formally identified as the *vanishing gradient*, the authors implemented a deep residual learning framework that utilises identity shortcuts that skip one or more layers, referred to as *skip connections*. This enables the gradients to flow directly through the skip connections, thus alleviating the vanishing gradient problem. The central concept is to encourage the learning of residual mappings rather than expecting stacked layers to learn mappings directly.

Mathematically, a single residual block is defined as:

$$\mathbf{y} = \mathcal{F}\left(\mathbf{x}, \{W_i\}\right) + \mathbf{x} \tag{2.1}$$

here x and y are the input and output vectors of the layers considered, the function $F\left(x, \{W_i\}\right)$ is the residual mapping to be learned. Note that in the equation 2.1

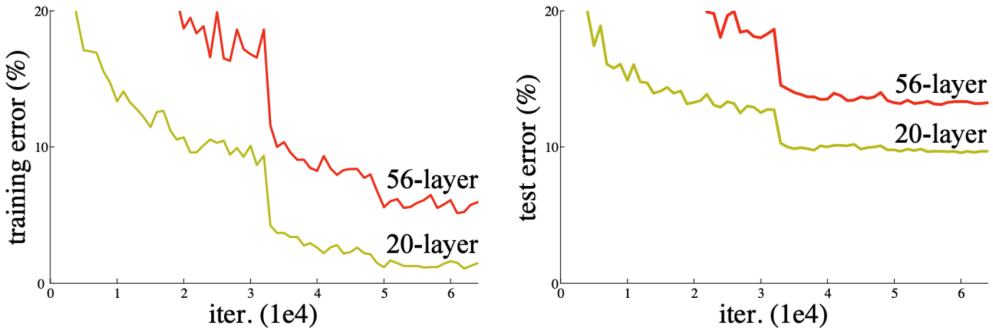

Figure 2.2: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer 'simple' networks. The deeper network exhibits a higher training error, implicating a higher test error. Source [13].

the dimensions of $\mathbf{x}$ and $\mathcal{F}$ must be equal. If it's not the case (e.g., when changing the input/output channels), we can perform a linear projection W_s by the shortcut connections to match the dimensions and the equation can be rewritten as:

$$\mathbf{y} = \mathcal{F}\left(\mathbf{x}, \{W_i\}\right) + W_s\mathbf{x}. \tag{2.2}$$

A graphical representation of the residual block is in Figure 2.3.

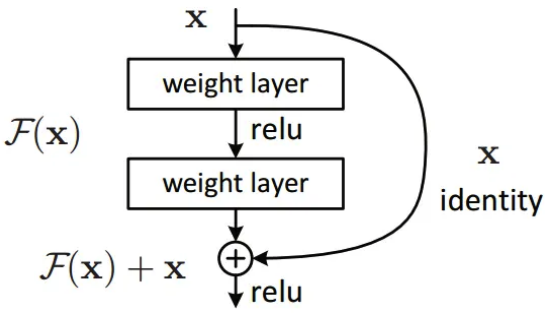

Figure 2.3: A residual learning building block. Source [13].

Moreover, the design of a 34-layer ResNet is shown in Figure 2.4:

2.4 Graph Neural Network

In this section, we examine the fundamental subject of this study, namely, *Graph Neural Networks (GNN)*. GNN serves as a versatile framework for developing Deep Neural Networks (DNN) that function on graph data. The fundamental concept involves generating node representations that rely on the structure of the graph.

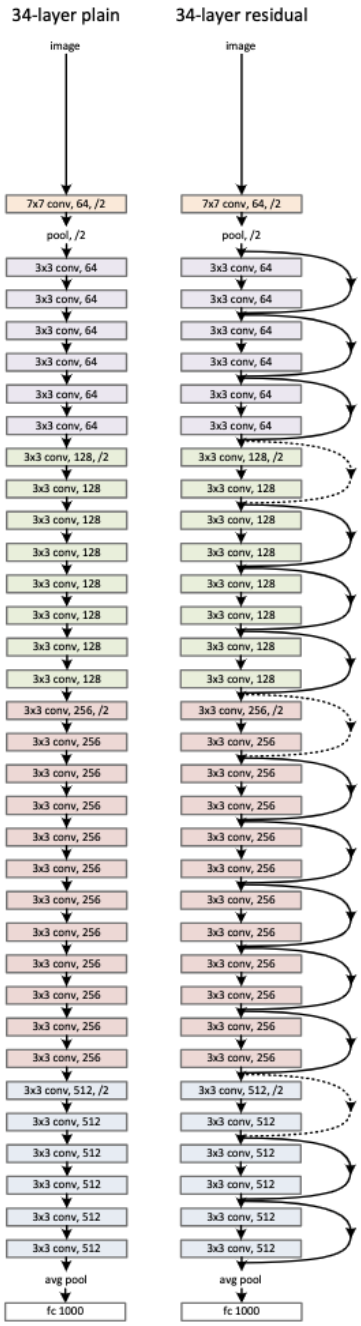

Figure 2.4: Example network architectures of ResNet. **Left**: a plain network with 34 parameter layers. **Right**: a residual network with 34 parameter layers. The dotted shortcuts increase dimensions. Source [13].

However, a difficulty arises when attempting to comprehend and construct intricate encoders for graph-structured data due to the lack of suitability of conventional deep learning techniques. For instance, Recurrent Neural Networks (RNN) are formulated to work with sequential data, such as text, whereas CNNs are tailored for grid-structured inputs like images. Therefore, to design a GNN, a new deep learning structure must be defined.

A possible approach to embedding an entire graph is to use its adjacency matrix as input. For instance, a flattened adjacency matrix could be input to a Multilayer Perceptron (MLP) as follows:

$$\mathbf{z}_{\mathcal{G}} = \text{MLP}(\mathbf{A}[1] \oplus \mathbf{A}[2] \oplus \ldots \oplus \mathbf{A}[|\mathcal{V}|]) \tag{2.3}$$

where $\mathbf{A}[i] \in \mathbf{R}^{|\mathcal{V}|}$ represent a single row of the adjacency matrix and $\oplus$ the vector concatenation.

The key problem with the previously mentioned method is its reliance on the order in which the nodes were arranged to create the adjacency matrix. This impedes the model's permutation invariance, or equivariance, which is vital in GNNs. Generalizing across assorted random sequences enables models to concentrate on the inherent structure of the graph. Mathematically, a desirable function f that receives an adjacency matrix A as input ought to conform to one of two properties:

$$f\left(\mathbf{PAP}^{\top}\right) = f(\mathbf{A})(\text{Permutation Invariance}) \tag{2.4}$$

$$f\left(\mathbf{PAP}^{\top}\right) = \mathbf{P}f(\mathbf{A})(\text{Permutation Equivariance}), \tag{2.5}$$

where $\mathbf{P}$ is a permutation matrix. Permutation invariance implies that the function is independent of the random arrangement of rows/columns in the adjacency matrix, while permutation equivariance implies that the output of f is consistently permuted along with the adjacency matrix.

2.4.1 Neural Message Passing

The distinguishing characteristic of a GNN is that it uses a kind of neural message passing, where vector messages are transferred among nodes and are updated by neural networks [20]. In the following paragraphs we will dive in the details of the neural message passing framework. In particular, how the node embeddings $\mathbf{z}_u, \forall u \in \mathcal{V}$ are generated from an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ along with a set of node features $\mathbf{X} \in \mathbb{R}^{d \times |\mathcal{V}|}$.

Overview of the Framework

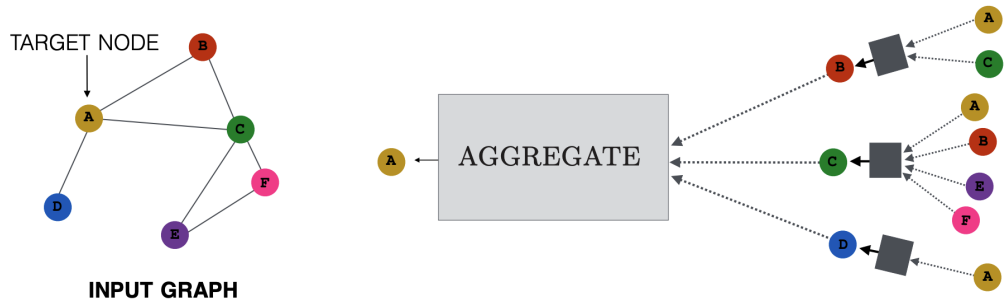

Figure 2.5: An analysis of the process by which a single node aggregates information from its immediate surroundings is carried out. The model focuses primarily on the acquisition of information from node A's neighbors (specifically, nodes B, C, and D). Consequently, the information disseminated by these neighboring nodes is based on the data they have collated from their respective environments, and this process persists accordingly. The figure illustrates a two-layer version of a message passing system. It is worth noting that as the vicinity around the central node expands, the computational diagram of the Graph Neural Network (GNN) shifts towards a tree-like structure. Image source [21]

At each stage of message-passing within a GNN, a *hidden embedding* $\mathbf{h}_u^{(k)}$ belonging to each node $u \in \mathcal{V}$ is updated according to the aggregated information from its own neighbor nodes $N(u)$. In mathematical terms, this update to message passing can be stated as follows:

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{Update}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{Aggregate}^{(k)}\left(\left\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\right\}\right)\right) \\ &= \text{Update}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right).\end{aligned}$$

In this equation, *Update* and *Aggregate* are two differentiable functions and $\mathbf{m}_{\mathcal{N}(u)}$ is the message vector that is aggregated from node u's neighbors nodes $\mathcal{N}(u)$.

Basically, at the kth iteration of the GNN, from the perspective of a single node u, the *Aggregate* function takes as input the embedding of each of its neighbors $\mathcal{N}(u)$ and generates the message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ from the aggregated neighborhood information. The *Update* then merges the message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ with the embedding of the previous step $\mathbf{h}_u^{(k-1)}$ of node u to generate the updated embedding $\mathbf{h}_u^{(k)}$. The starting embeddings at $k = 0$ are fixed to the initial features of all nodes, i.e. $\mathbf{h}_u^{(0)} = \mathbf{x}_u, \forall u \in \mathcal{V}$. After running K iterations of GNN message passing, the result of the last layer can be used to define the embedding of each node, which is the following:

$$\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}.$$

It is important to note that since the *Aggregate* function requires a set as input, GNNs so defined are permutation equivariant by default. The overall Message Passing framework is displayed graphically in Figure 2.5.

Motivations and Intuitions

The fundamental concept that underpins the GNN message-passing framework is easily understandable: during each iteration, every node collects data from its surroundings, and as these iterations proceed, the node embeddings contain more and more knowledge from distant parts of the graph. To clarify further:

- Following the initial iteration (k = 1), each node embedding incorporates knowledge from its direct neighborhood, i.e. it incorporates information about the properties of its direct neighbors that can be reached by a path of length 1 in the graph (1-hop).
- After the second iteration (k = 2), each node embedding includes the embeddings from its 2-hop neighbors.
- In general, after a number of k iterations, every node embedding encompasses information about its k-hop neighborhood.

Node embeddings created by GNN's message passing contain two primary types of data. The first type is *structural* information about the graph. For example, after k iterations, the embedding $\mathbf{h}_u^{(k)}$ of a node u could contain details about the degrees of all nodes within the k-hop proximity of u. This structural knowledge can be extremely valuable for a variety of tasks. When examining the shape of molecules, the degree information can be leveraged to deduce atom types and to recognize different structural patterns, such as toluene rings.

Furthermore, GNN node embeddings include information based on characteristics, also called *feature-based* data. Following k steps of GNN message passing, each node embedding contains information about its k-hop neighborhood and the node embeddings take in data about all qualities in their k-jump region. The feature-aggregation method of GNNs can be compared to the convolutional kernel method of CNNs. But unlike CNNs, which gather feature data from spatial regions of an image, GNNs gather data based on local neighborhoods within the graph.

2.4.2 GCN

In order to generalize the convolution operation from standard signal processing to sigal defined on graphs, in 2016 Thomas N. Kipf and Max Welling proposed the Graph Convolutional Network (GCN) [22]. The authors define the convolved signal

matrix $X' \in \mathbb{R}^{|\mathcal{V}| \times F}$ (where F is the cardinality of the feature maps in output) as follows:

$$X' = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} X \Theta \tag{2.6}$$

where $X \in \mathbb{R}^{|\mathcal{V}| \times d}$ as defined previously, $\hat{A} = A + I$ is the adjacency matrix with inserted self-loops, $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ its diagonal degree matrix and $\Theta \in \mathbb{R}^{d \times F}$ is the matrix of filter parameters. So that the node-wise message passing function is defined as follows:

$$\mathbf{h}_v^{(k)} = \Theta^\top \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}. \tag{2.7}$$

The GCN, as shown in the equation above, employs the Kipf normalization, also known as symmetric normalization, that down-weighting the contributions from high-degree nodes. This is achieved by introducing a normalization factor based on the degrees of the nodes involved in each interaction. Specifically, each edge contribution is divided by the square root of the product of the degrees of its two nodes. In this way, the nodes with a higher degree have a reduced contribution and this ensure a more balanced message aggregation.

2.4.3 GAT

Beyond broad set aggregation methods, GNNs often enhance their aggregation layer through the incorporation of attention mechanisms. This idea was inspired by the work of Bahdanau et al. [23], who proposed the concept of assigning an attention weight or relevance to each neighboring node, thereby modulating its impact during the aggregation process.

The Graph Attention Network (GAT) [24] was the first GNN model to utilize this attention mechanism. The GAT model leverages attention weights to formulate a weighted sum of the neighboring nodes as follows:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v. \tag{2.8}$$

In the equation above, $\mathbf{m}_{\mathcal{N}(u)}$ represents the aggregated information at node u. $\alpha u, v$ is the attention weight assigned to neighbor $v \in \mathcal{N}(u)$ when aggregating information at node u.

The original GAT paper defines the attention weights $\alpha_{u,v}$ as follows:

$$\alpha_{u,v} = \frac{\exp\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v\right]\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}\right]\right)}. \tag{2.9}$$

In the above equation, $\mathbf{a}$ is a trainable attention vector and $\mathbf{W}$ is a trainable weight matrix. The symbol $\oplus$ denotes the concatenation operation, which is applied

to the transformed feature vectors $\mathbf{Wh}_u$ and $\mathbf{Wh}_v$ of the nodes u and v, respectively. The softmax function ensures the attention weights for the neighbors of each node sum to 1, providing a normalized measure of importance of each neighbor's features in the aggregation process. A structural representation of GAT is displayed in Figure 2.6.

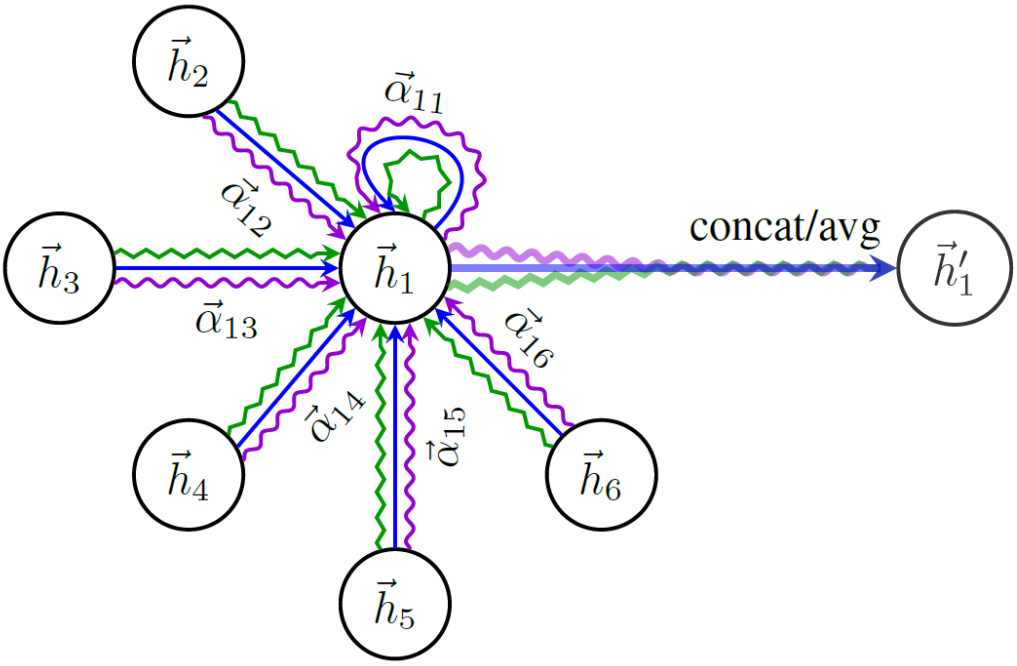

Figure 2.6: Structural representation of GAT.

2.4.4 GNNs and its application in Computer Vision

GNNs have garnered significant attention in recent years owing to their ability to model and interpret graphically structured data. Sources [25] and [26] contain the first references to GNNs. Micheli [27] devised an innovative spatial-based Graph Convolutional Network (GCN) that integrated non-recursive layers. In more recent years, many different types of spatial-based GCNs have been built, as discussed in references [28], [29], [20]. Moreover, Bruna et al. [30] presented a subcategory of GNNs called spectral-based GCNs, which use spectral graph theory for graph convolutions. Numerous improvements and extensions to spectral-based GCNs have been proposed since their introduction, as cited in [31], [32] and [22]. GCNs have been used in various fields, such as social networks, citation networks, and biochemical graphs [33] [34], to study and comprehend the complex relationships present in these graph-structured datasets.

In computer vision, GCNs have been applied to perform tasks such as point cloud classification, as highlighted in references [35], [36], [37], and [38]. Point cloud classification consists of the evaluation of groups of 3D points obtained from LiDAR scans. For this purpose, GCNs have been intensively investigated ([36], [39], [40]). Scene graph generation attempts to transform input images into graphs depicting the relationships between objects. This task commonly employs object detectors in conjunction with GCNs to enhance precision and efficacy (Yang et al., 2018; Xu et al., 2017). GCNs have been applied to handle naturally-occurring graphs that consist of interconnected human joints when recognising how humans carry out actions [41], [42].

Advantages of GNN in Computer Vision

There are several advantages in employing GNNs in computer vision tasks, some of them can be outlined in the following:

- **Graph as a Universal Data Structure**: GNNs exploit the flexibility of graphs as a data structure. Whereas images can be depicted as grids or matrices of pixels, graphs offer a more generalised mapping. In a graph-based model, pixel neighbourhoods can be depicted as nodes interconnected by edges. Grid and sequence representations are simply specific instances of graphs - a grid is a regular graph with fixed connection patterns, and sequences are similar to linear graphs. This flexibility allows GNNs to represent a wide range of data structures, thereby enabling more comprehensive and in-depth analysis.

- **Increased Flexibility over Grids/Sequences**: GNNs offer increased flexibility in modeling irregular shapes, which are often encountered in images that do not comply with fixed grids. Graphs allow the modeling of arbitrary interconnections between pixels or regions, a degree of abstraction not possible with grid-based models. In addition, graph edges can connect non-adjacent pixels/regions, allowing the modeling of long-range dependencies that are beyond the reach of grid representations. This makes GNNs particularly efficient in image analyses where contextual relationships between distant pixels can be critical.

- **Modeling Compositional Structures of Objects**: GNNs can efficiently depict objects that have composite textures. For instance, a plane is composed of several spatially connected parts. Graphs can eloquently represent these part-whole relationships between components. Each part or component can be represented by a node, and edges connecting these nodes can model how these parts assemble into the whole object. This feature provides a more intuitive

and effective way of representing complex objects; in fact, it is more in line with our natural understanding of these objects.

- **Transferability Across Disciplines**. In contrast to domain-specific models, the graph-based structure of GNNs allows techniques developed in one discipline to be adapted to others. Findings made using GNNs for social network analysis or bioinformatics can be retooled to address computer vision challenges. This cross-disciplinary portability allows computer vision researchers to build on advances made outside their field. The flexible architecture of GNNs allows knowledge to be shared across research silos, broadening their potential applications.

2.5 Sentinel-2

The Sentinel-2 mission, part of the European Union's Copernicus program, plays an important role in land monitoring through high-resolution optical imaging. Consisting of two polar-orbiting satellites, this mission provides continuous and frequent observations of the Earth's land and coastal areas. Building on the legacy of missions such as Landsat and SPOT, Sentinel-2 offers improved spectral resolution and more frequent revisit times.

The main aim of Sentinel-2 is to observe alterations in vegetation, soil, and water coverage, with a view to applying the data in agriculture, forestry, and water management. Furthermore, Sentinel-2 is involved in providing emergency response, security, and climate monitoring services, extending beyond terrestrial applications.

Each Sentinel-2 satellite bears a MultiSpectral Instrument (MSI) payload, enclosing 13 spectral bands ranging from visible and near-infrared to shortwave infrared regions. Four of these bands have a spatial resolution of 10 m, six have a resolution of 20 m, and three have a resolution of 60 m. This comprehensive spectral resolution across pivotal land surface wavelengths furnishes intricate environmental information. The shortwave infrared bands offer valuable data on the moisture content of vegetation and the health of plants. The spectral bands are displayed in Table 2.1.

Sentinel-2 constitutes a significant development in the field of operational land monitoring, providing regular revisits, vast coverage, boosted spatial resolution, and augmented spectral information. The data, which is readily accessible, backs a range of environmental applications and eases the creation of novel data outputs and utilities. In carrying on from previous land observation missions, Sentinel-2 supports crucial endeavors of data collection, catering to both scientific research and practical applications.

Band	Resolution	Central wavelength	Description
B1	60 m	443 nm	Ultra Blue (Coastal and Aerosol)
B2	10 m	490 nm	Blue
B3	10 m	560 nm	Green
B4	10 m	665 nm	Red
B5	20 m	705 nm	Visible and Near Infrared (VNIR)
B6	20 m	740 nm	Visible and Near Infrared (VNIR)
B7	20 m	783 nm	Visible and Near Infrared (VNIR)
B8	10 m	842 nm	Visible and Near Infrared (VNIR)
B8a	20 m	865 nm	Visible and Near Infrared (VNIR)
B9	60 m	940 nm	Short Wave Infrared (SWIR)
B10	60 m	1375 nm	Short Wave Infrared (SWIR)
B11	20 m	1610 nm	Short Wave Infrared (SWIR)
B12	20 m	2190 nm	Short Wave Infrared (SWIR)

Table 2.1: Sentinel-2 spectral bands. Source [43]

Chapter 3

Methodology

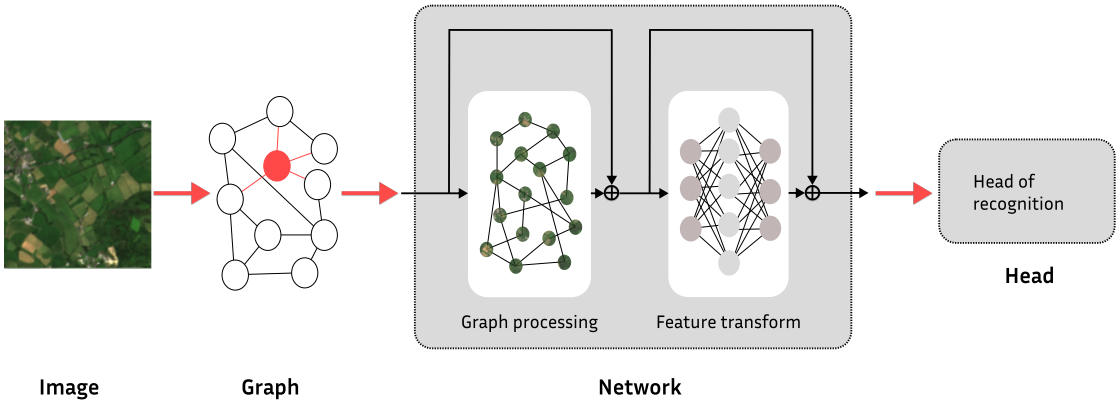

Figure 3.1: The framework of ViG.

This chapter presents a comprehensive study of the methodology utilized in this research. Specifically, we investigate the design of ViG, an innovative strategy that utilizes GNNs to extract graph-level characteristics.

In essence, ViG processes an image as a graph; an image is first divided into equal portions, which are then treated as nodes of a graph. Information is shared and modified among these nodes by means of graph convolution operations. The ViG architecture comprises two key modules: the Grapher module, which gathers and updates graph data, and the feed-forward network (FFN) module, which transforms node features. This chapter explains the construction of graph structures from images, graph-level processing, and the creation of ViG blocks, which are the central components of the ViG network. A detailed explanation of the methodology

is provided, supported by mathematical representations.

In the later sections of this chapter, we discuss the adaptations made to the original structure of ViG for specific applications. These adaptations involve modifying the graph convolution layer, adjusting the number of heads in the multi-head update operation, and incorporating relative positional encoding. These modifications aim to boost the model's capability to manage intricate visual tasks and enhance its predictive capabilities. Ultimately, we present options to the max-relative graph convolution operation, particularly the GCN and the Graph Attention Network (GAT) convolution techniques in their original formulations. These alternative methods are examined as feasible alternatives to the graph convolution operation in the Grapher module in this study.

3.1 Vision Graph Neural Network

Vision Graph Neural Network, in short Vision GNN or ViG, is a novel architecture proposed by Han et al. [44] to represent an image as a graph to extract graph-level features for visual tasks.

3.1.1 Graph Structure of Image

Given an image of dimensions $H \times W \times C$, it is divided into N distinct parts, also called patches. Each patch is transformed into a feature vector, which is represented as $\mathbf{x}_i \in \mathbb{R}^D$.; through this process, a matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ is created, where i ranges from 1 to N and D represents the feature dimension. The transformed features can be viewed as an unordered collection of nodes represented by $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$. Each node v_i is associated with the identification of the K closest neighbors, denoted as $\mathcal{N}(v_i)$. Next, an edge e_{ji} is added, directed from v_j to v_i, for all $v_j \in \mathcal{N}(v_i)$.

This process results in the formation of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ denotes the complete set of edges. The process that leads to this graph construction is denoted as $\mathcal{G} = G(X)$. Having established the image as graph data, it is possible to investigate the application of GNN for the extraction of representative features.

3.1.2 Graph level processing

Once a graph $\mathcal{G}$ is created based on the the features $X \in \mathbb{R}^{N \times D}$, a graph convolutional layer is employed to exchange information between nodes by aggregating features from its neighboring nodes. In particular, graph convolution operates in the following manner:

$$\mathcal{G}' = F(\mathcal{G}, \mathcal{W}) = \text{Update}\left(\text{Aggregate}\left(\mathcal{G}, W_{\text{agg}}\right), W_{\text{update}}\right). \tag{3.1}$$

In this equation, W_{agg} and W_{update} denote the learnable weights tied to the aggregation and update operations correspondingly. To be more specific, the aggregation operation calculates the node's representation by combining the features of its neighboring nodes. Then, the update operation merges the aggregated feature with the existing node representation:

$$\mathbf{x}_i' = h\left(\mathbf{x}_i, g\left(\mathbf{x}_i, \mathcal{N}\left(\mathbf{x}_i\right), W_{\text{agg}}\right), W_{\text{update}}\right). \tag{3.2}$$

In this equation, $\mathcal{N}\left(\mathbf{x}_i\right)$ signifies the set of neighbor nodes of $\mathbf{x}_i$. Following this, the original ViG paper [44] utilizes the max-relative graph convolution [45] for its straightforwardness and efficiency, which is formalized as follows:

$$g(\cdot) = \mathbf{x}_i'' = \left[\mathbf{x}_i, \max\left(\left\{\mathbf{x}_j - \mathbf{x}_i \mid j \in \mathcal{N}\left(\mathbf{x}_i\right)\right\}\right)\right], \tag{3.3}$$

$$h(\cdot) = \mathbf{x}_i' = \mathbf{x}_i'' W_{\text{update}}\,. \tag{3.4}$$

In these equations, the bias term is intentionally left out. In the following sections, the processing at the level of the entire graph will be denoted as $X' = \text{GraphConv}(X)$.

Moreover, a multi-head update operation is introduced in this layer, which is an advanced mechanism for aggregating and transforming node features. In fact, this operation enhances the model's ability to capture diverse image features and spatial relationships. Here a detailed explanation of the process:

- *Splitting into Heads*: The aggregated feature $\mathbf{x}_i''$ is initially decomposed into h heads. This division allows for various smaller vectors to be formed, each capable of learning and concentrating on distinct aspects of the input data. This implies that concerning attention mechanisms, every head can concentrate on different parts of the input sequence.

- *Updating Heads*: These heads are then updated with different weights. Here, $W^1_{\text{update}}, \mathrm{W}^2_{\text{update}}, \cdots, \mathrm{W}^h_{\text{update}}$ are the weight matrices that are specific to each head. This operation is akin to each head learning its own representation of the input data.

- *Parallel Updating*: All heads can be updated in parallel. This is one of the benefits of this approach, as it supports efficient computation.

- *Concatenation*: The updated heads are finally concatenated to form the final output feature vector $\mathbf{x}_i'$. This vector now contains information from all the different heads, and hence, it is expected to have a richer representation of the input data.

We can formalize this operation in the following manner:

$$\mathbf{x}_i' = \Big[\text{head}^1 W^1_{\text{update}} \,,\ \text{head}^2 W^2_{\text{update}} \,, \cdots \,,\ \text{head}^h W^h_{\text{update}} \Big] . \tag{3.5}$$

The incorporation of a multi-head update operation enhances the model's capacity to handle intricate visual tasks. This, in turn, allows the model to operate in multiple representational subspaces simultaneously, resulting in a wider and more versatile range of image features. As a result, the model excels in capturing complex spatial relationships and patterns within the image.

3.1.3 ViG Block

A common issue with earlier Deep GCN architectures is the diminishing expressive power, occurring with an increase in the number of stacked convolutional layers. This problem is referred to as *over-smoothing* [46] [47] and occurs due to the feature vectors' increasing similarity among the nodes in the graph. Consequently, prediction performance is severely degraded. To address this occurrence, the authors of ViG implemented further feature transformations and nonlinear activ

Indeed, to address the aforementioned issue of *over-smoothing*, the authors of ViG propose to introduce a linear layer both preceding and following the graph convolution. The goal of this adjustment is to project the node features into the same domain, thereby promoting feature diversity. Moreover, to avoid the phenomenon of layer collapse, a nonlinear activation function is introduced subsequent to the graph convolution process. The resulting enhanced module is referred to as the Grapher module. Fundamentally, given an input feature $X \in \mathbb{R}^{N \times D}$, the expression for the Grapher module is formulated as follows:

$$Y = \sigma \left(\text{Graph Conv} \left(X W_{\text{in}} \right) \right) W_{\text{out}} + X \tag{3.6}$$

where $Y \in \mathbb{R}^{N \times D}$. Here, W_{in} and W_{out} denote the weights of fully-connected layers, while σ represents the activation function, such as ReLU. Please note that the bias term has been omitted. To further enhance the capacity for feature transformation, a feed-forward network is applied at the node level. Specifically, the FFN module is a straightforward multi-layer perceptron comprising two fully-connected layers:

$$Z = \sigma \left(Y W_1 \right) W_2 + Y \tag{3.7}$$

In this equation, $Z \in \mathbb{R}^{N \times D}$, and W_1 and W_2 represent the weights of the fully-connected layers. It is worth noting that the hidden dimension of the FFN is typically larger than D, the dimension of the node features. Additionally, batch normalization is applied subsequent to every fully-connected layer within both the

Grapher and FFN modules, although this detail is omitted from Equations 3.6 and 3.7 for the sake of brevity.

The fundamental unit of the ViG network, termed the ViG block, is composed of a sequence of Grapher and FFN modules. Given the graph representation of images and the deployment of the ViG block, the comprehensive ViG network architecture is depicted in Figure 3.1.

3.1.4 Network architecture

In computer vision, two primary types of network structure are prevalent: isotropic and pyramid. Each of these architectures has unique characteristics and is suited to different types of tasks.

Isotropic architectures treat all spatial dimensions equally without accounting for variations in scale. As a result, the same processing is applied to all parts of the image regardless of size or level of detail. This architectural approach is employed by models like the Vision Transformer (ViT) [14] and ResMLP [48]. Although this architecture is simpler and more uniform, it may not comprehensively capture the intricate spatial hierarchies that exist in various real-world images.

Conversely, pyramidal architectures are engineered to exploit multiple scales to capture information at different levels of granularity. The architectures process distinct image sections at different scales, instead of uniformly treating spatial dimensions. This enables a better capture of the natural hierarchical structure of images. ResNet [13] and the Pyramid Vision Transformer (PVT) [15] are models that showcase the implementation of the pyramid architecture.

During the development of ViG, both isotropic and pyramid network architectures were tested. Nevertheless, empirical findings have shown that pyramid architectures are typically more effective for visual tasks [15]. This is most likely due to their ability to capture and exploit the hierarchical structure of images, a key feature of many visual challenges.

The developers of ViG have chosen to create four distinct versions of Pyramid ViG, which include information on their structure, layer configuration, and parameters in Table 3.1. Through an analysis of these Pyramid ViG variations, we can gain a deeper understanding of how a variety of design options can impact the effectiveness of pyramid structures in visual tasks.

3.1.5 Adaptations

In the previous Subsections 3.1.1, 3.1.2, 3.1.3 and 3.1.4 we outlined the original structure of ViG [44]. Now we want to mention some adjustment to the architecture made that will be further explained in detail Chapter 4. Note that all the considerations are done referring to Pyramid ViG architecture since empirical

Stage	Output size	PyramidViG-Ti	PyramidViG-S	PyramidViG-M	PyramidViG-B
Stem	$\frac{H}{4} \times \frac{W}{4}$	Conv $\times$ 3	Conv $\times$ 3	Conv $\times$ 3	Conv $\times$ 3
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix} D=48 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=80 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=96 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=128 \\ E=4 \\ K=9 \end{bmatrix} \times 2$
Downsample	$\frac{H}{8} \times \frac{W}{8}$	Conv	Conv	Conv	Conv
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	$\begin{bmatrix} D=96 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=160 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=192 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=256 \\ E=4 \\ K=9 \end{bmatrix} \times 2$
Downsample	$\frac{H}{16} \times \frac{W}{16}$	Conv	Conv	Conv	Conv
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} D=240 \\ E=4 \\ K=9 \end{bmatrix} \times 6$	$\begin{bmatrix} D=400 \\ E=4 \\ K=9 \end{bmatrix} \times 6$	$\begin{bmatrix} D=384 \\ E=4 \\ K=9 \end{bmatrix} \times 16$	$\begin{bmatrix} D=512 \\ E=4 \\ K=9 \end{bmatrix} \times 18$
Downsample	$\frac{H}{32} \times \frac{W}{32}$	Conv	Conv	Conv	Conv
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	$\begin{bmatrix} D=384 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=640 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=768 \\ E=4 \\ K=9 \end{bmatrix} \times 2$	$\begin{bmatrix} D=1024 \\ E=4 \\ K=9 \end{bmatrix} \times 2$
Head	1×1	Pooling & MLP	Pooling & MLP	Pooling & MLP	Pooling & MLP
Parameters (M)		10.7	27.3	51.7	92.6
FLOPs (B)		1.7	4.6	8.9	16.8

Table 3.1: Pyramid ViG series parameters. D is the feature dimension, E is the hidden dimension ratio in FFN, K is the number of neighbours in GCN, $H \times W$ is the input image size. 'Ti' stands for tiny, 'S' for small, 'M' for medium, and 'B' for base. Source [44]

evidences and the results demonstrated the higher effectiveness of this structure for image classification task.

Reduction factor

The original pyramid ViG was tailored for 224×224 images (*e.g.* ImageNet [1]) so that at each step after the downsample a sufficient number of nodes are present, which was 9 in the original formulation (see in Table 3.1 the parameter K). However, when working with smaller images, such as the 120×120 images in the BigEarthNet dataset [49], precautions had to be taken to maintain a consistent number of neighbors in the message passing layer.

While it might seem straightforward to just upsample the input images to the size of 224×224, there are several reasons we decided against this approach.

Firstly, in the context of our application, upsampling would mean artificially increasing the resolution of the images from the BigEarthNet dataset, which was originally 120x120. This procedure can introduce interpolation artifacts and distort the original image content. This could lead to misleading or incorrect

information being fed into our model. Additionally, upsampling does not provide extra information to the images; it merely enlarges the existing pixels, which could lead to the model overfitting to these features rather than learning meaningful ones.

Furthermore, applying upsampling to each image in the dataset to 224x224 would considerably boost the computational complexity and memory demands on the model. This could restrict the scalability of our method, rendering it less efficient and potentially impractical for the extensive datasets we are examining in this study.

In light of this, we elected to adopt a versatile and more efficient approach, which entailed adapting the pyramid ViG architecture to suit smaller images. By halving each dimension of the image after every downsampling stage with a constant reduce factor, whilst preserving the number of graph convolution layers and the same number of node neighbors, we were able to maintain the original image resolution and prevent interpolation artifacts from being introduced.

Furthermore, by maintaining the same number of neighboring nodes throughout the network, this expansion of the receptive field effectively enhances the model's capability to capture both local and global graph patterns. This results in a more resilient representation of the graph structure, ultimately improving the model's discriminative power.

Multi-head update

Moreover, due to the limited resources available to perform the experiments in this work, which will be detailed in Chapter 4, we have made the decision to set the number of heads in the multi-head update operation of the graph convolution to 2. However, it is important to note that increasing this number results in improved prediction performance. The reason behind this improvement lies in the fact that by increasing the number of heads, the model's parameters also increase, enabling the Grapher module to process node features in different subspaces. This allows for a more comprehensive exploration of the graph structure and enhances the model's ability to capture and leverage diverse information from the nodes, leading to enhanced predictive capabilities.

Furthermore, the introduction of multiple heads in the graph convolutional layer introduces parallelism and allows for simultaneous processing of node features from different perspectives. Each head focuses on a different aspect or viewpoint of the graph, enabling the model to capture diverse and complementary information. By incorporating multiple heads, the model gains the ability to learn and aggregate features at various levels of abstraction.

However, it is important to strike a balance when determining the number of heads. While increasing the number of heads can enhance performance, there

is a trade-off with computational complexity and resource requirements. As the number of heads increases, the overall model size and computational cost also grow. Therefore, the decision to set the number of heads to 2 in this work takes into consideration the available resources while still benefiting from the performance gains achieved through multi-head operations.

3.2 Positional Encoding

Positional encoding is a well-established method for incorporating into Transformer models [50] information about the relative position of words in a sequence. As the transformer architecture lacks recurrence or convolution, it has no inherent awareness of word order. Positional encoding is used to provide further information on the position of each word, in order to help the model use the sequence order to improve accuracy. The authors of the study limit their attention to fixed (non-trainable) positional encoding [51].In particular, the authors use sine and cosine functions of different frequency to calculate the a value for each input vector in the following manner:

$$PE_{(pos,2i)} = \sin\Big(\ pos\ /10000^{2i/d_{\text{model}}}\Big)$$
$$PE_{(pos,2i+1)} = \cos\Big(\ pos\ /10000^{2i/d_{\text{model}}}\Big)$$

where pos is the position and i is the dimension; note that each dimension of the positional encoding corresponds to a sinusoid and that the positional encodings are added to the input embedding so they have to be of size d_{model} .

A very powerful visualization is provided by [52] and is displayed in Figure 3.2.

3.2.1 Positional and Relative Positional Encoding in ViG

In the ViG model we can exploit the above mentioned technique to represent the position information of the nodes. In particular, given the positional encoding vector $\mathbf{e}_i \in \mathbb{R}^D$, where D is the feature node dimension, for each node:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{e}_i \tag{3.8}$$

In the context of Pyramid ViG, we tried to further improve our model by incorporating the concept of relative positional encoding, a mechanism that was originally introduced in the Swin Transformer [53]. The relative positional encoding plays an essential role in capturing the spatial relationships between different elements in an input sequence, which is particularly beneficial in our case, where the spatial arrangement of nodes can carry significant information.

Let's consider two arbitrary nodes, denoted as i and j, in the graph. We can calculate the relative positional encoding between these two nodes by computing

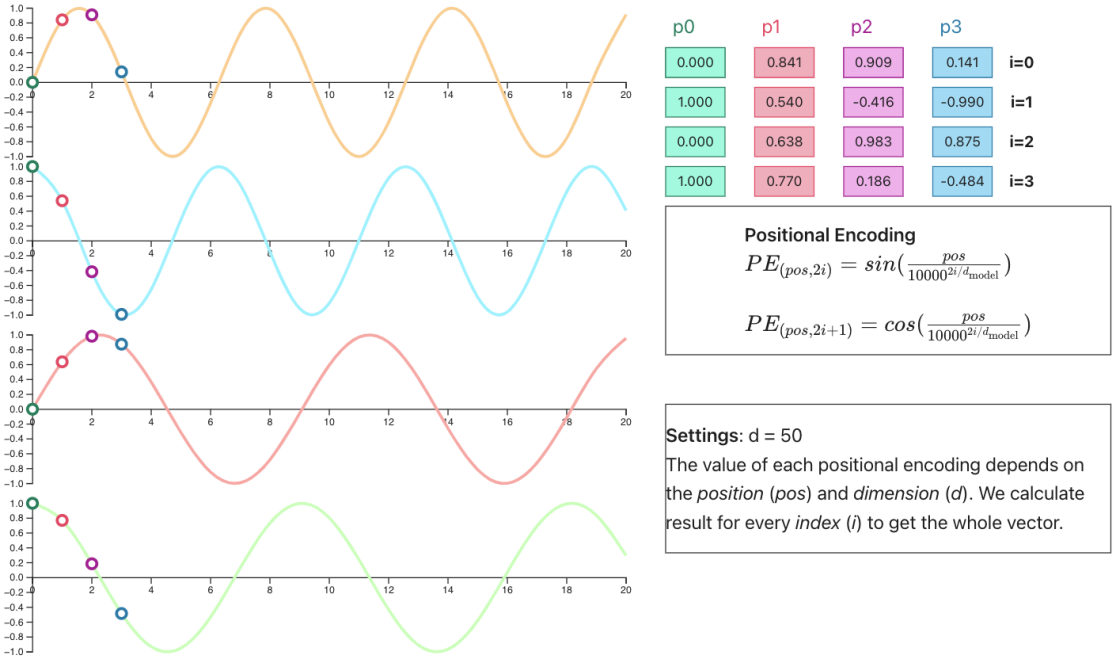

Figure 3.2: Positional encoding visualization. Source [52].

the dot product of their individual positional encodings, $\mathbf{e}_i^T \mathbf{e}_j$. The result of this computation is a scalar value that encapsulates the spatial relationship between nodes i and j.

This scalar value is then used to modify the feature distance matrix at each step of the graph convolutional layer, which is responsible for constructing the graph. Specifically, the relative positional matrix obtained from the previous step is added to the feature distance matrix. This addition operation essentially infuses the spatial information captured by the relative positional encoding into the feature distance matrix, thereby enriching the graph with additional spatial context.

It's important to note that the relative positional encoding is a static matrix, i.e., its values are not updated during the training process. This is because it represents the inherent spatial relationships between different nodes in the graph, which are independent of the specific task that the GNN is designed to perform.

Given the pyramid architecture of our model, the graph undergoes a downsampling process at every step. This effectively reduces the spatial resolution of the graph, thereby altering the spatial relationships between different nodes. Consequently, the dimension of the relative positional matrix may vary at each step of the downsampling process. This variation is accounted for in our model, ensuring that the relative positional encoding remains accurate and meaningful throughout the entire graph convolution process.

3.3 Graph Convolution Layer

In accordance with the details presented in Section 3.1.2, the network under study initially employed the max-relative graph convolution as its primary operation. This technique, while effective in certain contexts, has its limitations especially when dealing with more complex graph structures. Motivated by the need for more flexible and adaptive graph processing techniques, this work introduces two alternative graph convolution methods: the Graph Convolutional Network (GCN) and the Graph Attention Network (GAT) convolutions, as originally formulated in [22] and [24], respectively.

3.3.1 GCN and GAT convolution

The GCN convolution, as detailed in Section 2.4.2 of Chapter 2, is characterized by two main features. Firstly, it utilizes a self-loop update mechanism. This innovative approach helps to prevent overfitting by ensuring that a node's information remains distinct from its neighbours' information during the aggregation process. Secondly, GCN uses a symmetric normalized aggregation technique which tempers the influence of nodes with a high degree of connections. This ensures a balanced aggregation process, preventing highly connected nodes from monopolizing the information flow.

In contrast, the GAT convolution, discussed in detail in Section 2.4.3, employs an attention mechanism to aggregate information. This mechanism assigns a unique attention score to each neighbour node, which effectively quantifies its importance in the context of the given task. This allows the network to focus more on significant nodes and reduce attention on less relevant ones, leading to more efficient and targeted information processing.

Moreover, the GAT approach offers the advantage of supporting bipartite message passing operations. This functionality enables the model to process pairs of node features—specifically, source and target nodes—as input. This introduces a broader spectrum of learnable associations, permitting the network to learn more effective node-level attention coefficients. This process accentuates the importance of context and structure in the graph, resulting in a more nuanced model that can adapt to the intricacies of the graph data.

Chapter 4

Experiments

In this chapter, we provide an in-depth overview of the exact procedures, specific details, and comprehensive results related to the experiments conducted in our research. Our primary resource was the BigEarthNet dataset, leveraging its Sentinel-2 bands. This multispectral dataset is a robust collection of high-resolution satellite images, offering a rich variety of spectral features that proved instrumental in our study.

We present a detailed description of our experimental setup, focusing on the tools used, the metrics applied, and the specifics of the process. The primarily libraries used are Pytorch, Pytorch Geometric and Lightning. The main key performance indicator of this work are precision, recall, and F1-score. We justify the choice of these metrics within the context of our specific research objectives.

Then we dedicate an important portion of the chapter to a detailed discussion of the results obtained by comparing the several settings, considering only the RGB or the whole Sentinel-2 bands. In particular, we displayed some plot regarding the training performances considering the training loss and the loss and F1 score on validation per each epochs. Then, we discuss the results on test split. In the end, we have also a discussion about the models' trainable parameters and a qualitative comparison with model predictions along with the images.

4.1 BigEarthNet S-2

In this work, we employed BigEarthNet, which is a multispectral large scale dataset to train, validate and test the several settings of ViG comparing them also with ResNet-101 model. In the specifics, BigEarthNet is a multi-label image classification dataset, where the true label of each image in the archive is associated to one or more classes that are provided from the CORINE Land Cover (CLC) database. Moreover, considering the Sentinel-2's spectral bands in Table 2.1, the total 590,326

image patches have the following resolution: 120×120 for $10m$ bands, 60×60 for $20m$ bands and 20×20 for $10m$ bands. Note that since BigEarthNet is a Sentinel-2 L2A dataset, the 10th band of the one displayed in Table 2.1 is not included.

For the purposes of our research, we employed Level-3 of the CLC nomenclature, implying that each image is associated with one to twelve class labels out of a possible 43. Upon examining the dataset using this particular label nomenclature, it becomes apparent that the data distribution is imbalanced, as can be observed from Table 4.1. For instance, broader labels such as *Mixed forest* and *Pastures* are associated with a greater number of images, while more specific labels like *Airport* are less represented. In general, 95% of the images are associated with at most five labels. Several examples of images from the dataset, along with their respective labels, can be seen in Figure 4.5 and 4.10.

4.2 Experimental Setup

In the following we will describe the metrics, libraries, tools and settings of our work.

4.2.1 Problem statement and approach

Given the complexity of multi-label classification problems, it is imperative to discuss the methodological approach for addressing these tasks. In such scenarios, the number of nodes in the final layer of our model should correspond to the total quantity of labels. For a conventional classification task, where labels are mutually exclusive, the softmax function is typically employed to convert the raw output of the final layer, also referred to as logits, into probabilities. These probabilities, which collectively sum to one, are then used to identify the class with the highest confidence as the predicted outcome.

However, this reasoning is not directly applicable to multi-label classification tasks due to the variable nature of label associations. For instance, in the BigEarthNet dataset, the true label count for each image can range from one to twelve within the full set of potential classes. Therefore, the traditional approach in such situations involves applying a sigmoid function to each of the output logits, mapping each input to a value between 0 and 1. A predetermined probability threshold is then used to classify those values exceeding this limit as the predicted classes. In our research, we established the threshold value at 0.5, that considering the characteristics of a sigmoid function, this decision implies that classes with strictly positive logits are taken into account.

Class name	**Number of images**
Mixed forest	217,119
Coniferous forest	211,703
Non-irrigated arable land	196,695
Transitional woodland/shrub	173,506
Broad-leaved forest	150,944
Land principally occupied by agriculture, with significant areas of natural vegetation	147,095
Complex cultivation patterns	107,786
Pastures	103,554
Water bodies	83,811
Sea and ocean	81,612
Discontinuous urban fabric	69,872
Agro-forestry areas	30,674
Peatbogs	23,207
Permanently irrigated land	13,589
Industrial or commercial units	12,895
Natural grassland	12,835
Olive groves	12,538
Sclerophyllous vegetation	11,241
Continuous urban fabric	10,784
Water courses	10,572
Vineyards	9,567
Annual crops associated with permanent crops	7,022
Inland marshes	6,236
Moors and heathland	5,890
Sport and leisure facilities	5,353
Fruit trees and berry plantations	4,754
Mineral extraction sites	4,618
Rice fields	3,793
Road and rail networks and associated land	3,384
Bare rock	3,277
Green urban areas	1,786
Beaches, dunes, sands	1,578
Sparsely vegetated areas	1,563
Salt marshes	1,562
Coastal lagoons	1,498
Construction sites	1,174
Estuaries	1,086
Intertidal flats	1,003
Airports	979
Dump sites	959
Port areas	509
Salines	424
Burnt areas	328

Table 4.1: Class distribution of BigEarthNet S-2 Level-3 CLC

Figure 4.1: Discontinuous urban fabric, Non-irrigated arable land, Complex cultivation patterns, Land principally occupied by agriculture, with significant areas of natural vegetation.

Figure 4.2: Pastures, Moors and heathland, Peatbogs.

Figure 4.3: Construction sites, Non-irrigated arable land, Pastures, Coniferous forest, Inland marshes, Water courses.

Figure 4.4: Discontinuous urban fabric, Pastures, Broad-leaved forest, Coniferous forest, Mixed forest.

Figure 4.5: Some images along with the labels of BigEarthNet dataset.

Figure 4.6: Pastures, Land principally occupied by agriculture, with significant areas of natural vegetation, Coniferous forest, Transitional woodland/shrub.

Figure 4.7: Pastures, Coniferous forest, Moors and heathland, Transitional woodland/shrub.

Figure 4.8: Non-irrigated arable land, Pastures.

Figure 4.9: Burnt areas, Peatbogs.

Figure 4.10: Some images along with the labels of BigEarthNet dataset.

4.2.2 Metrics

In order to evaluate the experiments carried out in this work, we considered the following metrics: precision, recall, and F1 score. Specifically, we considered micro-aggregation in order to obtain a comprehensive aggregation of these evaluation metrics. Micro-aggregation, a technique that aggregates the contributions of all classes, allows us to compute a more holistic average metric.

The metrics are formally defined as follows:

$$\textit{micro-precision} = \frac{\sum \mathrm{TP}}{(\sum \mathrm{TP} + \sum \mathrm{FP})}, \tag{4.1}$$

$$\textit{micro-recall} = \frac{\sum \mathrm{TP}}{(\sum \mathrm{TP} + \sum \mathrm{FN})}, \tag{4.2}$$

$$\textit{micro-F1} = 2 \cdot \frac{\textit{micro-precision} \cdot \textit{micro-recall}}{(\textit{micro-precision} + \textit{micro-recall})}. \tag{4.3}$$

In these equations, the terms TP, FP, and FN represent true positive, false positive, and false negative values, respectively. These terms are cumulatively summed across all the classes. The micro-aggregation technique ensures equitable treatment of all instances, irrespective of the class they emerge from. Thus, it aggregates the contributions of all instances to compute a comprehensive average metric, rather than calculating metrics for each class in isolation. It is essential to underline that this particular method of aggregation is particularly suited to our objective, given the class imbalance in the BigEarthNet dataset, as discussed in the previous Section (4.1).

4.2.3 Libraries and Tools

The entirety of this work's code is written in Python, specifically making use of the renowned deep learning framework, PyTorch [54]. In addition to PyTorch, we also utilized several libraries that have been built on top of it, these include PyTorch Lightning [55], PyTorch Geometric [56], and TorchGeo [57]. PyTorch Lightning, an open-source library, proves to be highly beneficial as it allows us to organize the code in a manner that enhances clarity and promotes reproducibility. Furthermore, it enables the model to be hardware agnostic and capitalizes on its integrated callback system. On the other hand, PyTorch Geometric is specifically designed to ease the development of deep learning models based on a variety of geometric structures, such as graphs. Recalling the structure of ViG, the Grapher module handle the graph level processing through the message passing layer. In particular, this module was reimplemented from the original formulation with the built in Class of message passing in Pytorch Geometric library. Moreover, both

GCN and GAT convolutions layers have been also imported from it to perfom our investigation. TorchGeo has provided the BigEarthNet dataset for this study. As we utilized Sentinel-2 images, and the bands possess different spatial resolutions, this library efficiently upsamples all of them to a resolution of 10 meters. We also made use of the torchvision package that is part of PyTorch to import the ResNet101 model and also the scikit learn library for its built-in evaluation metrics. The most relevant libraries required to replicate the experiments are displayed in Table 4.2 along with their versions.

During the course of our study, the tool that proved to be exceptionally useful for monitoring our experiments was Comet.ml. It allowed us to track and compare our experiments, even in live mode, by providing an array of visualization tools. Lastly, all of our experiments were conducted using an Nvidia Tesla V100 GPU with 32GB.

Package	**Version**
Pytorch	2.0.1
Pytorch Geometric	2.3.1
Pytorch Lightning	2.0.2
TorchGeo	0.4.1
Comet ML	3.33.3
Scikit Learn	1.2.2

Table 4.2: The most relevant libraries along with their versions.

4.2.4 Settings

In Table 4.3 the parameters of the training process are displayed. We trained the models, both ViG with several configurations and ResNet-101, for a maximum of 100 epochs with a batch size of 4. The optimizer chosen is Adam with 10^{-3} as learning rate, without weight decay. A step learning rate scheduler was chosen to reduce it of a factor 10 every 10 epochs. Finally, since we are in a multi-label classification task we chose a binary cross entropy with logits as loss function.

Parameter	Value
Epochs	100
Batch size	4
Optimizer	Adam
Learning rate	0.001
Scheduler	StepLR
Loss function	BCE with logits loss

Table 4.3: Training parameters along with their values.

4.3 Experimental Results

In this section, we present the results of experiments evaluating three convolutional layer architectures - GCN, GAT, and Max-Relative convolution - incorporated into ViG. We assessed the ViG model with these different convolutional layers and with or without relative positional encoding, using all the Sentinel-2 bands imagery from BigEarthNet. For comparison, we evaluated also a not pre-trained ResNet-101 model on this dataset to establish baseline benchmark performance.

In addition to that, we decided to conduct further tests using RGB bands. These experiments were designed to serve as a benchmark test, helping us to understand the value of the extra information provided by the full Sentinel-2 spectrum. The same experimental setup and evaluation metrics were used for both the RGB and full spectrum experiments, ensuring a fair comparison.

We trained, validated, and tested the models using the official split of the BigEarthNet. Since we did not set a fixed seed in our experiments, we computed the *mean* and *standard deviation* for each model configuration based on three separate runs. As outlined in Section 4.2.2, we evaluated the models by micro-averaging the precision, recall, and F1 scores across all classes.

In the following, for each scenario, we will first discuss the training performances of ViG's variants, commenting the trends of the training and validation's losse as well as the validation's f1 score during our experiments. To ensure clear interpretation, only a single run for each model configuration is displayed on the plots. Additionally, the validation are performed every 2 epochs of the training. Then, we will evaluate the results using the the mean and standard deviation the of the metrics derived from the test set over three distinct runs. The seed was changed for each run, which facilitated a comprehensive comparison of various model configurations.

4.3.1 RGB bands

Training performances of ViG's variants

Considering the ViG's variants, in Figures 4.11, 4.12 and 4.13 are displayed the validation loss, f1 score and the training loss respectively. First of all, we can notice that the training loss is decreasing over the epochs reaching a stable value around 0.12. Considering the validation loss, which is computed every 2 epochs of training, also in this case considering all the ViG's configurations it has a decreasing trend overall, witnessing that our models are able to generalize and are not overfitting. Then, considering the f1 score on validation we can see that overall it has an increasing trend, reaching a value of approximately 0.60 with the best model configuration.

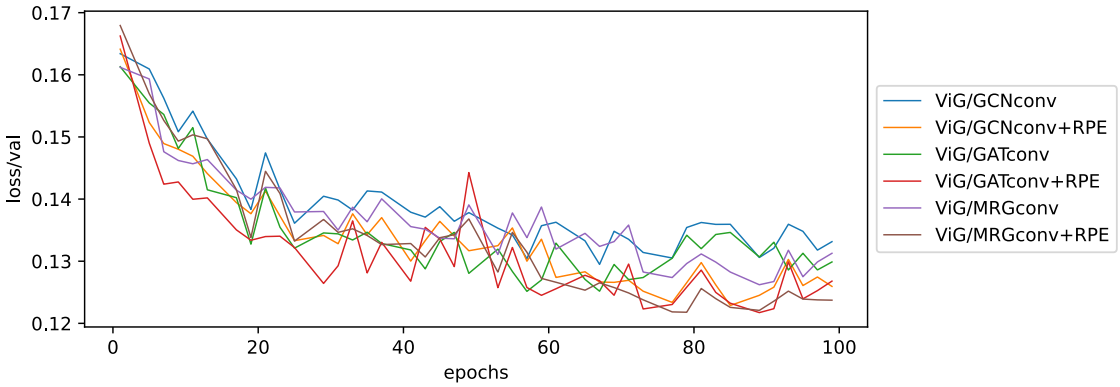

Figure 4.11: RGB validation loss during the epochs.

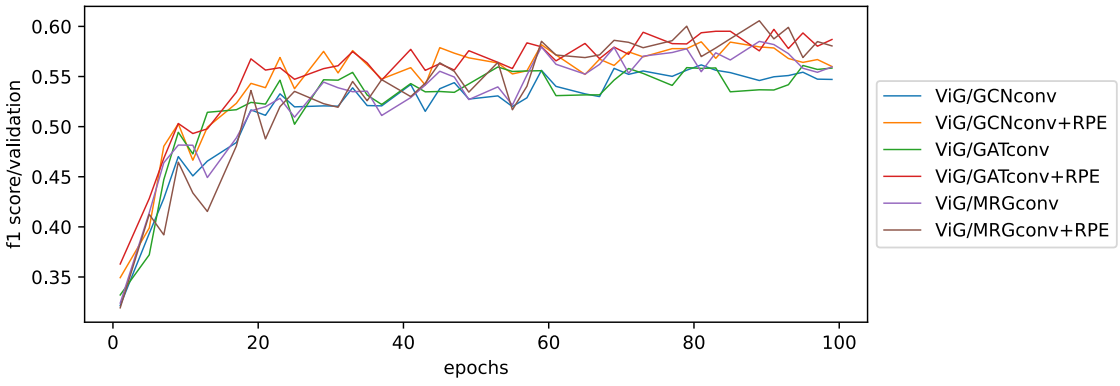

Figure 4.12: RGB F1 score on validation during the epochs.

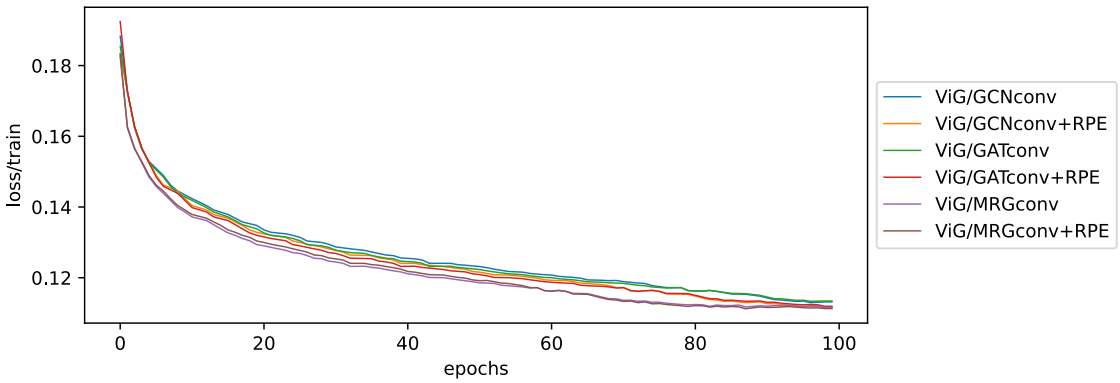

Figure 4.13: RGB training loss during the epochs.

Results

Model configuration	F1 score		Precision		Recall	
	Mean	std	Mean	std	Mean	std
ResNet-101	0.5752	0.175	0.7901	0.108	0.4499	0.043
ViG/GCNconv	0.5721	0.091	0.7881	0.134	0.4491	0.059
ViG/GCNconv+RPE	0.5977	0.101	0.7531	0.027	**0.4954**	0.073
ViG/GATconv	0.5688	0.109	0.7714	0.035	0.4505	0.096
ViG/GATconv+RPE	0.5985	0.127	0.7733	0.062	0.4883	0.088
ViG/MRGconv	0.5868	0.114	0.7896	0.038	0.4669	0.095
ViG/MRGconv+RPE	**0.6093**	0.126	**0.8049**	0.040	0.4901	0.091

Table 4.4: RGB results of each of the configuration on BigEarthNet S-2 test set. MREconv stands for Max-Relative Graph convolution, RPE stands for relative positional encoding and the mean.

The results derived from using the Red, Green, and Blue bands from the Sentinel-2 bands of BigEarthNet are displayed in Table 4.4. Upon analyzing the results, it is evident that each configuration of the ViG model surpasses the performance of the benchmark model, ResNet-101. This superior performance is recorded across all three evaluation metrics: F1 score, precision, and recall.

Diving deeper into the performance of the three graph convolutional layers, a distinct pattern emerges. The Max-Relative Graph convolution (MRGconv), which is the convolutional layer used in the original formulation of the ViG model, consistently outperforms the other two layers, GCN and GAT. Specifically, the

improvement in the F1 score when using MRGconv is approximately 3% compared to GCN and roughly 2% when compared to GAT.

This pattern of superior performance by the MRGconv layer persists when relative positional encoding is enabled. With this additional feature, the F1 score of the ViG model with the MRGconv layer increases to an average of 0.6093 across the three runs. This configuration also achieves the highest precision among all the model configurations evaluated. However, the highest recall is observed when the GCN convolutional layer is used with positional bias enabled.

Overall, the implementation of relative positional encoding contributes to superior performance, particularly in terms of the F1 score, across all three graph convolutional layers examined.

4.3.2 Multi spectral bands

Training performances of ViG's variants

Considering the Figures 4.14, 4.15 and 4.16 we can see the validation loss, f1 score and the training loss respectively, in this scenario. In particular, we can make similar considerations to the ones made previously for the RGB scenario. Also in this case, both training and validation's losses have decreasing trends over the several model configurations witnessing the ability to generalize without overfitting in the training process. The F1 score on validation plot, on the other hand, have an overall increasing trend, reaching value of approximately 0.70 for the best model configuration.

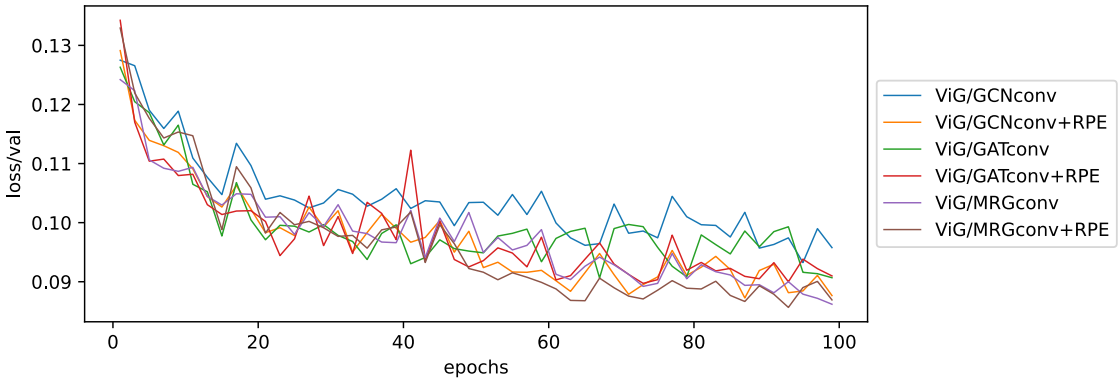

Figure 4.14: Multi-band validation loss during the epochs.

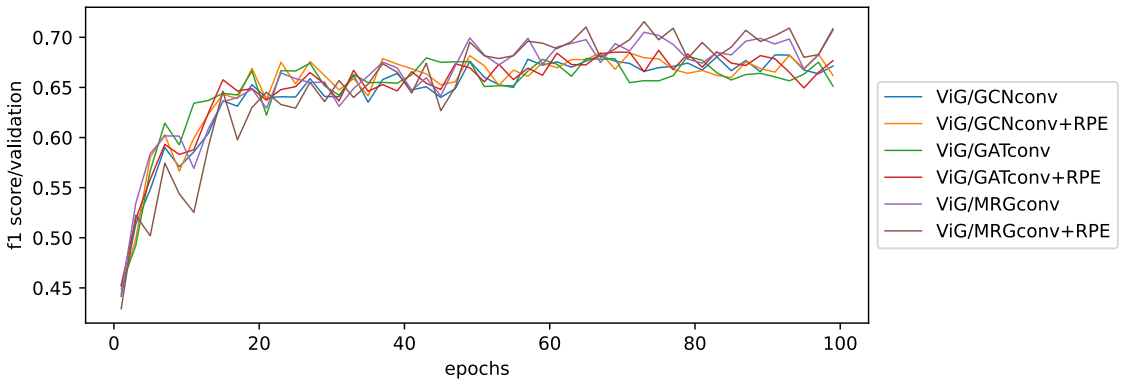

Figure 4.15: Multi-band F1 score on validation during the epochs.

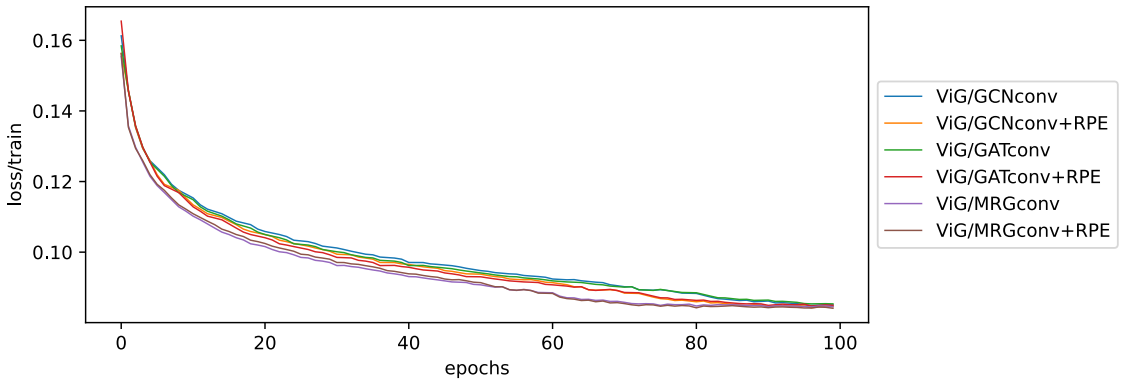

Figure 4.16: Multi-band training loss during the epochs.

Results

Model configuration	F1 score		Precision		Recall	
	Mean	std	Mean	std	Mean	std
ResNet-101	0.6656	0.026	0.8147	0.082	0.5625	0.112
ViG/GCNconv	0.6769	0.046	0.8023	0.099	0.5855	0.129
ViG/GCNconv+RPE	0.6879	0.056	0.7785	0.141	0.6162	0.072
ViG/GATconv	0.6840	0.045	**0.8173**	0.098	0.5881	0.067
ViG/GATconv+RPE	0.6898	0.104	0.8045	0.033	0.6038	0.086
ViG/MRGconv	0.70465	0.121	0.8120	0.048	0.6223	0.099
ViG/MRGconv+RPE	**0.7136**	0.124	0.8151	0.051	**0.6408**	0.082

Table 4.5: Multibands results of each of the configuration on BigEarthNet S-2 test set. MREconv stands for Max-Relative Graph convolution, RPE stands for relative positional encoding.

Switching to the analysis to all the bands of Sentinel-2 within the BigEarthNet, the results of which are also displayed in Table 4.5, we replicate the exact experiments that were conducted in the RGB scenario.

As we could expect, the trends observed from the RGB scenario are largely replicated here. The ViG model maintains its superior performance over the baseline model, ResNet-101, demonstrating its consistent effectiveness. However, it is important to note that considering all the Sentinel-2 bands, we have observed an increase in performance, across all the settings, by approximately 10% in terms of the F1 score. This increase is attributed to the major feature set available in the input data with respect to the previous setting.

Analogous to the RGB scenario, the MRGconv continues to achieve the highest F1 scores among the three convolutional layers. This efficient performance is sustained even when relative positional encoding is enabled. Specifically, the configuration considering the MRGconv, coupled with positional encoding, yields the highest F1 score and recall among all the model configurations, with respective values of 0.7136 and 0.6408. We can also notice that the highest precision in this context is achieved when the GAT convolutional layer is used, specifically in the absence of the positional bias.

In summary, the employment of relative positional encoding continues to yield improved results across all three graph convolutional layers. Among these, the MRGconv stands out as the most effective, affirming its consistency across varying experimental conditions.

Model's parameters

Figure 4.17 shows a quantitative scatter plot illustrating the correlation between the performance of each model setting and their total number of parameters. The horizontal axis represents the total number of trainable parameters, while the vertical axis represents performance in terms of F1 score in a multi-band scenario. Notably, this axis uses a logarithmic scale due to the significant difference in the number of parameters between the ResNet-101 and ViG models, with the former having one order of magnitude higher than the latter.

We can notice that observing the plot the ViG's variants with the relative positional encoding enable are not displayed since the positional bias introduce only static non trainable parameters. Specifically, the configuration with the maximum relative convolution layer has approximately 2.1 millions parameters, all of which are trainable. In contrast, relative positional coding introduces approximately 3 millions parameters, none of which are trainable due to the static nature of the encoding.

Overall, the plot highlights the superiority of the ViG model over the baseline ResNet-101 model evaluated in this work. Despite having significantly fewer

parameters, the ViG model outperforms ResNet-101 in terms of the performance indicator.

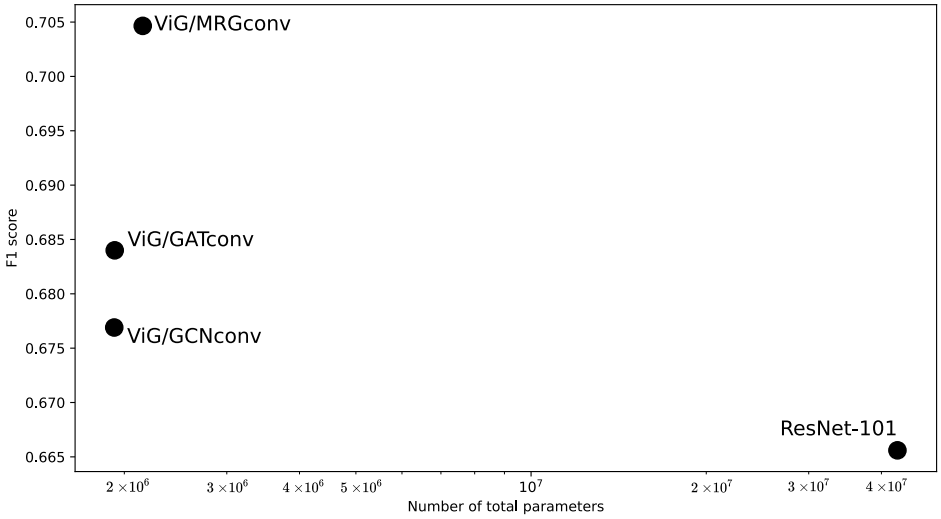

Figure 4.17: Scatter plot with axis-X representing the total number of trainable parameters of each model configurations and axis-Y the F1 score in multi-band scenario.

4.3.3 Qualitative comparison

To provide a more qualitative comparison between the model variants, we further examined the multi-label predictions associated with some of the images from the test split using the ResNet-101 model and the ViG variant using the MRG convolution with positional bias enabled, both in the multi-band scenario.

In particular, considering Figure 4.6, we can see 5 different Sentinel-2 images with the associated true multi-label and the several labels assigned by 2 models taken into consideration. In particular, analyzing one image per time, we can see that the first one is associated with 2 labels, *Non-irrigated arable land* and *Pastures*, that recalling from the class distribution in Section 4.1 are quite common. The ResNet-101 model predicts the two labels correctly but in addition, it assigns another label, while the ViG model predicts them correctly. Considering the second image our baseline model assigns correctly the two common labels, *Pastures* and *Sea and ocean*, but also in this case assigns other two labels to the image which are similar to the true classes; in particular *Complex cultivation patterns* are similar to *Pastures*, while *Water bodies* are similar to *Sea and ocean*. Considering the third image, the ResNet model in this case wasn't able to assign a rare label to the image,

Test Image	True Multi-Label	ResNet-101	ViG/MRGconv+RPE
	Non-irrigated arable land, Pastures	Non-irrigated arable land, Pastures, Annual crops associated with permanent crops	Non-irrigated arable land, Pastures
	Discontinuous urban fabric, Pastures, Sea and ocean	Complex cultivation patterns, Pastures, Sea and ocean, Water bodies	Discontinuous urban fabric, Pastures, Sea and ocean
	Discontinuous urban fabric, Sport and leisure facilities, Pastures	Discontinuous urban fabric, Pastures	Discontinuous urban fabric, Sport and leisure facilities, Pastures
	Discontinuous urban fabric, Pastures, Coniferous forest, Moors and heathland, Peatbogs, Sea and ocean	Discontinuous urban fabric, Pastures, Coniferous forest, Natural grassland, Sea and ocean	Discontinuous urban fabric, Pastures, Coniferous forest, Moors and heathland, Peatbogs, Sea and ocean
	Sea and ocean	Sea and ocean, Water bodies, Sparsely vegetated areas	Sea and ocean

Table 4.6: Example of BigEarthNet Sentinel-2 images with the true multi-labels and the multi-labels assigned by ResNet-101 and ViG/MRGconv with relative positional encoding.

i.e. Sport and leisure facilities. Moving forward, the fourth image is a particularly challenging one since the number of true labels is 6 in total. The baseline model also in this case does not perform well as the ViG model; in fact, it was not able to predict them all by also replacing the *Moors and hearthland* class with the *Natual grassland*. Finally, the last image presents an ambiguous image since it is only associated with the label *Sea and ocean*, while clearly by observing it there is a

noticeable green land. The ViG model, surprisingly, was able to assign only the single correct label, while ResNet-101 mistakenly assigned other two unrelated labels, which are *Water bodies* and *Sparsely vegetated areas*.

Chapter 5

Conclusions

This research aimed to explore the application of Vision Graph Neural Networks (ViG), a novel deep learning architecture, for multi-label land cover classification using remote sensing imagery. Specifically, we leveraged the BigEarthNet Sentinel-2 dataset and examined the performance of different graph convolutional layers within the ViG model.

We hypothesized that by incorporating different types of graph convolutional layers into the ViG architecture and by enabling or disabling relative positional encoding, we could improve the model's performance on the multi-label classification task. Furthermore, we conjectured that the use of all Sentinel-2 spectral bands would render superior results compared to using only the RGB bands.

Through systematic experiments, we validated that the ViG model outperformed ResNet-101 across all evaluation metrics. Among the graph convolutional layers, the max-relative convolution from the original ViG formulation achieved the best results. Enabling relative positional encoding consistently improved performance, especially the F1 score, for all three graph convolutional layers tested. Using the full Sentinel-2 bands provided superior accuracy compared to just RGB, demonstrating the value of the enhanced spectral information.

The results of our research validate the efficacy of the Pyramid ViG model for multi-label land cover classification tasks. The results presented over the several setting taken in consideration by varying the convolutional layers and the significant improvement brought by relative positional encoding underscore the potential of this approach for remote sensing applications. Moreover, our findings highlight the importance of using all available spectral bands in Sentinel-2 data to achieve superior performance.

While our research has made significant strides, several avenues for further exploration exist. In future works, we plan to:

- Test the ViG model on additional remote sensing datasets of varying types

and resolutions to enhance its generalizability.

- Investigate the performance of different types of graph convolutional layers within the ViG architecture, which could expand our understanding of the model's potential.
- Explore variations of relative positional encoding to better understand their impact.
- Extend the application of the ViG model beyond land cover classification to assess its versatility in other remote sensing tasks or entirely different domains.

In conclusion, our research provides valuable insights into Vision Graph Neural Networks and their application in remote sensing. Through systematic experimentation, we established the effectiveness of the ViG model in outperforming traditional models like ResNet-101. Our results also underscored the importance of the choice of convolutional layer and the introduction of relative positional encoding. We believe our findings pave the way for further exploration and optimization of Vision Graph Neural Networks, potentially unlocking new advancements in the field of remote sensing and beyond.

Bibliography

[1] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV] (cit. on pp. 5, 7, 25).

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV] (cit. on pp. 5–7).

[3] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2019. arXiv: 1812.08008 [cs.CV] (cit. on pp. 5, 7).

[4] Minguk Kang, Woohyeon Shim, Minsu Cho, and Jaesik Park. *Rebooting ACGAN: Auxiliary Classifier GANs with Stable Training*. 2021. arXiv: 2111.01118 [cs.CV] (cit. on pp. 5, 7).

[5] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. *Uformer: A General U-Shaped Transformer for Image Restoration*. 2021. arXiv: 2106.03106 [cs.CV] (cit. on pp. 5, 7).

[6] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. *MoViNets: Mobile Video Networks for Efficient Video Recognition*. 2021. arXiv: 2103.11511 [cs.CV] (cit. on pp. 5, 7).

[7] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. *BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation*. 2020. arXiv: 2004.02147 [cs.CV] (cit. on p. 6).

[8] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. *PolyTransform: Deep Polygon Transformer for Instance Segmentation*. 2021. arXiv: 1912.02801 [cs.CV] (cit. on pp. 6, 7).

[9] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, Ping Luo, and Tong Lu. *Panoptic SegFormer: Delving Deeper into Panoptic Segmentation with Transformers*. 2022. arXiv: 2109.03814 [cs.CV] (cit. on pp. 6, 7).

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV] (cit. on p. 7).

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386 (cit. on p. 6).

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 6).

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV] (cit. on pp. 6, 8–10, 24).

[14] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV] (cit. on pp. 7, 24).

[15] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. *Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions*. 2021. arXiv: 2102.12122 [cs.CV] (cit. on pp. 8, 24).

[16] Ilya Tolstikhin et al. *MLP-Mixer: An all-MLP Architecture for Vision*. 2021. arXiv: 2105.01601 [cs.CV] (cit. on p. 8).

[17] Shoufa Chen, Enze Xie, Chongjian Ge, Runjian Chen, Ding Liang, and Ping Luo. *CycleMLP: A MLP-like Architecture for Dense Prediction*. 2022. arXiv: 2107.10224 [cs.CV] (cit. on p. 8).

[18] Dongze Lian, Zehao Yu, Xing Sun, and Shenghua Gao. *AS-MLP: An Axial Shifted MLP Architecture for Vision*. 2022. arXiv: 2107.08391 [cs.CV] (cit. on p. 8).

[19] Jianyuan Guo, Yehui Tang, Kai Han, Xinghao Chen, Han Wu, Chao Xu, Chang Xu, and Yunhe Wang. *Hire-MLP: Vision MLP via Hierarchical Rearrangement*. 2021. arXiv: 2108.13341 [cs.CV] (cit. on p. 8).

[20] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG] (cit. on pp. 11, 15).

[21] William L. Hamilton. «Graph Representation Learning». In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.3 (), pp. 1–159 (cit. on p. 12).

[22] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG] (cit. on pp. 13, 15, 29).

[23] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL] (cit. on p. 14).

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML] (cit. on pp. 14, 29).

[25] M. Gori, G. Monfardini, and F. Scarselli. «A new model for learning in graph domains». In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. 2005, 729–734 vol. 2. DOI: 10.1109/IJCNN.2005.1555942 (cit. on p. 15).

[26] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. «The Graph Neural Network Model». In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605 (cit. on p. 15).

[27] Alessio Micheli. «Neural Network for Graphs: A Contextual Constructive Approach». In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 498–511. DOI: 10.1109/TNN.2008.2010350 (cit. on p. 15).

[28] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. *Learning Convolutional Neural Networks for Graphs*. 2016. arXiv: 1605.05273 [cs.LG] (cit. on p. 15).

[29] James Atwood and Don Towsley. *Diffusion-Convolutional Neural Networks*. 2016. arXiv: 1511.02136 [cs.LG] (cit. on p. 15).

[30] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. *Spectral Networks and Locally Connected Networks on Graphs*. 2014. arXiv: 1312.6203 [cs.LG] (cit. on p. 15).

[31] Mikael Henaff, Joan Bruna, and Yann LeCun. *Deep Convolutional Networks on Graph-Structured Data*. 2015. arXiv: 1506.05163 [cs.LG] (cit. on p. 15).

[32] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. 2017. arXiv: 1606.09375 [cs.LG] (cit. on p. 15).

[33] Nikil Wale, Ian A. Watson, and George Karypis. «Comparison of descriptor spaces for chemical compound retrieval and classification». In: *Knowledge and Information Systems* 14 (2006), pp. 347–375. URL: https://api.semanticscholar.org/CorpusID:2596211 (cit. on p. 15).

[34] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI] (cit. on p. 15).

[35] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. *Scene Graph Generation by Iterative Message Passing*. 2017. arXiv: 1701.02426 [cs.CV] (cit. on p. 16).

[36] Loic Landrieu and Martin Simonovsky. *Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs*. 2018. arXiv: 1711.09869 [cs.CV] (cit. on p. 16).

[37] Yongcheng Jing, Yining Mao, Yiding Yang, Yibing Zhan, Mingli Song, Xinchao Wang, and Dacheng Tao. *Learning Graph Neural Networks for Image Style Transfer*. 2023. arXiv: 2207.11681 [cs.CV] (cit. on p. 16).

[38] Runzhong Wang, Junchi Yan, and Xiaokang Yang. «Learning Combinatorial Embedding Networks for Deep Graph Matching». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00315. URL: https://doi.org/10.1109%2Ficcv.2019.00315 (cit. on p. 16).

[39] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. *Dynamic Graph CNN for Learning on Point Clouds*. 2019. arXiv: 1801.07829 [cs.CV] (cit. on p. 16).

[40] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. «Distilling Knowledge From Graph Convolutional Networks». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020 (cit. on p. 16).

[41] Sijie Yan, Yuanjun Xiong, and Dahua Lin. *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition*. 2018. arXiv: 1801.07455 [cs.CV] (cit. on p. 16).

[42] Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. *Structural-RNN: Deep Learning on Spatio-Temporal Graphs*. 2016. arXiv: 1511.05298 [cs.CV] (cit. on p. 16).

[43] *Sentinel 2 bands combinations*. urldate: 2023-10-18. URL: https://gisgeography.com/sentinel-2-bands-combinations/ (cit. on p. 18).

[44] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. *Vision GNN: An Image is Worth Graph of Nodes*. 2022. arXiv: 2206.00272 [cs.CV] (cit. on pp. 21, 22, 24, 25).

[45] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. *DeepGCNs: Can GCNs Go as Deep as CNNs?* 2019. arXiv: 1904.03751 [cs.CV] (cit. on p. 22).

[46] Kenta Oono and Taiji Suzuki. *Graph Neural Networks Exponentially Lose Expressive Power for Node Classification*. 2021. arXiv: 1905.10947 [cs.LG] (cit. on p. 23).

[47] Qimai Li, Zhichao Han, and Xiao-Ming Wu. *Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning*. 2018. arXiv: 1801.07606 [cs.LG] (cit. on p. 23).

[48] Hugo Touvron et al. *ResMLP: Feedforward networks for image classification with data-efficient training*. 2021. arXiv: 2105.03404 [cs.CV] (cit. on p. 24).

[49] Gencer Sumbul, Marcela Charfuelan, Begum Demir, and Volker Markl. «Bigearthnet: A Large-Scale Benchmark Archive for Remote Sensing Image Understanding». In: *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, July 2019. DOI: 10.1109/igarss.2019.8900532. URL: https://doi.org/10.1109%2Figarss.2019.8900532 (cit. on p. 25).

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL] (cit. on p. 27).

[51] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. *Convolutional Sequence to Sequence Learning*. 2017. arXiv: 1705.03122 [cs.CL] (cit. on p. 27).

[52] *Understanding Positional Encoding in Transformers*. urldate: 2021-05-10. URL: https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers#positional-encoding-visualization (cit. on pp. 27, 28).

[53] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV] (cit. on p. 27).

[54] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 36).

[55] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. DOI: 10.5281/zenodo.3828935. URL: https://github.com/Lightning-AI/lightning (cit. on p. 36).

[56] Matthias Fey and Jan Eric Lenssen. *Fast Graph Representation Learning with PyTorch Geometric*. May 2019. URL: https://github.com/pyg-team/pytorch_geometric (cit. on p. 36).

[57] Adam J. Stewart, Caleb Robinson, Isaac A. Corley, Anthony Ortiz, Juan M. Lavista Ferres, and Arindam Banerjee. «TorchGeo: Deep Learning With Geospatial Data». In: *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '22. Seattle, Washington: Association for Computing Machinery, Nov. 2022, pp. 1–12. DOI: 10.1145/3557915.3560953. URL: https://dl.acm.org/doi/10.1145/3557915.3560953 (cit. on p. 36).