Graph Network Based Generative Adversarial Network for Hyperspectral Image Classification

Michael Legesse Teferi

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of

Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September, 2022

Adama, Ethiopia

Graph Network based Generative Adversarial Network for Hyperspectral Image Classification

Michael Legesse Teferi

Advisor

Dr. Worku Jifara (Ph.D)

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of

Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September, 2022

Adama, Ethiopia

DECLARATION

I hereby declare that this Master's Thesis entitled "**Graph Network based Generative Adversarial Network for Hyperspectral Image Classification"** is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Michael Legesse		
Name	Signature	Date

RECOMMENDATION

We, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Graph Network based Generative Adversarial Network for Hyperspectral Image Classification**" prepared under my guidance by **Michael Legesse** submitted in partial fulfillment of the requirements for the degree of Master of Science computer science and engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. Worku Jifara (Ph.D.)		
Major Supervisor	Signature	Date
Co-Supervisor	Signature	Date

ACKNOWLEDGEMENT

First and for most I would like give thanks to the Almighty God, My Lord and My Savior for giving me the courage and strength throughout my study and help me accomplish my work from start to the end. I would like to thank Holy Virgin Mary.

Then, I am very grateful to my advisor Dr. Worku Jifara (Ph.D.) who were helping me from the proposal to the completion of this research work, giving me the right directions and guidance, useful comments, valuable support, and supervision.

I would also like to express my deepest thanks all my classmates for their supportiveness throughout my work. Finally, I would like to thank my families who were there in all my ups and down supporting and advising me, and my friends who have been supportive of my work and giving me advice throughout my thesis work

Table of Contents

List of Tables

List of Figures

List of Code Snippets

List of Equations

List of Abbreviations

AA	Average Accuracy
CA-GAN	Collaborative learning and Attention Mechanism based GAN
CGCN	Crystal Graph Convolutional Network
CN	Concatenation
CNN	Convolutional Neural Network
CONV3D	3D Convolutional Layer
EA	Each Accuracy
EWA	Element wise Addition
GAN	Generative Adversarial Network
GAP	Graph Attention Pooling Layer
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GIN	Graph Isometric Network
GNGAN	Graph Network based Generative Adversarial Network (Proposed model)
GNN	Graph Neural Network
HP	Hadamard Product
HSI	Hyperspectral Image
K	Kappa Coefficient
OA	Overall Accuracy
PCA	Principal component Analysis
SSGAN	Semi-supervised Generative Adversarial Network

Abstract

Generative Adversarial Network is one of the emerging deep learning technologies that is used for many applications. Semi-supervised classification is one of the application of GANs by using the discriminator as a classifier. Semi-supervised GANs are used in Hyperspectral image classification to address the limitation of labeled dataset for the training. Most semi-supervised GAN models utilizes Euclidian structure of Hyperspectral image in the discriminator and other models like GCN is used to utilize non-Euclidian structure of Hyperspectral image in form of graph. But most of the semi-supervised GAN models were not address the utilization of graph like structure which contains abundant information compared to Euclidean form. In the proposed solution, GNGAN model are used as semi-supervised classifier by utilizing both Euclidian structure which has joint spatial and spectral features and non-Euclidian structure in the form of graph which has deep spectral features in order to increase the performance of the classification processes. The proposed model uses Convolutional block and Graph network block networks for utilizing Euclidian and non-Euclidean structures of Hyperspectral image respectively. Concatenation mechanism was used as combination mechanism after compared with Hadamard product and Element wise Addition methods to combine the two blocks. The proposed classifier is carried out with two mostly used Hyperspectral image datasets: Salina Valley and Pavia University. The contribution of the proposed work is two which are: it address the long term dependencies of spectral features in non-Euclidian structure using Graph network block and it learns joint spatial spectral features in Euclidian structure using Convolutional block. The obtained results showed that the proposed model provide competitive results compared to other methods: SSGAN and GCN. The GNGAN outperform SSGAN and GCN by 0.5% and 3.66% for Salina Valley dataset and 3.8% and 6.17% for Pavia University dataset respectively using overall accuracy as evaluation matrix.

Key words: *Convolutional block, Graph Convolutional Network, Graph network block Generative Adversarial Network, Hyperspectral image, Semi-supervised learning.*

Chapter One

Introduction

1.1. Background Study

Hyperspectral image (HSI) is a type of spectral image which consists of spatial and spectral information that are used in image spectroscopy. It has 20 to hundreds continuous spectral band at each pixels with narrow wavelength band *(Khan M. et al, 2018)*. Digital camera that capture HSI is called hyperspectral image spectroscope which captures the Spatial-spectral data cube also known as Hypercube *(Arun P. et al, 2020)*. Hyperspectral images have many application in many areas by identifying the composition of materials and processes on the ground based on their spectral signature that has taken from the hyperspectral image *(El-masry G. et al, 2021)*. Identifying minerals in mining *(Sundharsan S, et al, 2019)*, classifying plants with diseases in agriculture *(Hsieh T & Kiang J, 2020)*, and exploring the composition of objects in space exploration *(Guilloteu C. et al, 2020)* some of the application areas where hyperspectral images are used.

The process of image classification is to assign a class either to an entire image, a portion of image or a pixels of an image which is known as pixel wise image classification. In the HSI, image classification is defined as the process of assigning individual pixels to a set of classes and the classified image is known as classification map *(Li C. et al, 2018)*. Based on the availability of labeled HSI, there are 3 types of HSI classification methods which are: Supervised, Unsupervised and Semi- supervised HSI classification *(Rejesh G. & Chaturvedi A., 2019)*. Supervised method uses labeled training datasets whereas semi supervised methods uses both labeled and unlabeled datasets. Unsupervised method uses only unlabeled datasets.

In semi-supervised HSI classification, different schemes are employed to use both labeled and unlabeled training datasets for the classification purpose. Graph based deep learning model *(Aydemir M. & Bilgin G., 2017)* and Generative Adversarial Networks GANs *(Odena A. 2016)* are the common semi-supervised based schemes for HSI classifications.

Generative Adversarial Network GAN is known for tasks such as generating images, image to image translations and text to image translations *(Zhu J. et al, 2017) (Karras T. et al, 2020)*. Recently, GAN is also used for semi-supervised classification purpose where there are small number of labeled data available for training purposes *(Odena A, 2016)*. The discriminator of semi-supervised GAN is used not only for distinguishing generated images as real or fake but also classify real labeled images in to classes by training using both labeled and unlabeled images.

The structure of Hyperspectral image in 3D cube form is known as Euclidian structure *(Wan S. et al, 2019)*. In this form, spatial features can be expressed in 2D plain where pixels are positioned and spectral features can be expressed in 1D vector which can be found in each pixel. Hyperspectral image can also be expressed in non-Euclidian structure in the form of graphs *(Ding Y. et al, 2021), (He S. et al, 2021)*. The graphs can be constructed based on spectral relationship among pixels or relationship among pixels based on their spatial features. In graph form, the relationship among pixels based on their spectral feature can be easily utilized compared with HSI in Euclidian form

Several prior works tries to accommodate both structures by using different deep learning models for HSI classifications *(Liu S. et al, 2021) (Liu Q. et al, 2020)*. In this work, Graph Convolutional Network GCN was employed to utilize non-Euclidian structure of HSI. GCN can be used as node classifier which is makes the GCN model a semi-supervised classifier by converting HSI cubes in single graph or disjoint graphs *(Hong D. et al, 2020).* But node classification via GCN needs large amount of computing power during the computation of adjacency matrix for a single graph. GCN also used as graph classifier acts as supervised classifier.

In prior works which uses Semi-supervised GAN models, utilizes only Euclidian structure of HSI for classification task *(Mehta A. et al, 2020) (Xie W. et al, 2021).* Due to that, misclassification happens for not utilizing non-Euclidian structures. This research tries to address this problem.

In this research, Graph network GAN (GNGAN) model is proposed for utilizing both Euclidian and non-Euclidian structure of HSI during semi-supervised classification. 3D CNN based network called Convolutional block is introduced in the discriminator for utilizing Euclidian structure of HIS whereas GCN based network called Graph network block is introduced to discriminator of Semi-supervised GAN (SSGAN) for utilizing graphs generated from HSI. Concatenation

mechanism is used to integrate Graph network block with Convolutional block in the discriminator of GNGAN.

The experiment results shows that GNGAN improves the classification performance by utilizing both forms compares to SSGAN and GCN models which utilizes only Euclidian structure and non-Euclidian structure respectively.

1.2. Motivation

Hyperspectral image is used in many areas and some of them are beneficial to agriculture, hydrology and environmental applications. these areas needs either high resolution, accurate and real time ground measurements or remote sensing images which contains a lot of information as much as ground measurements. Hyperspectral images can achieve that since it has hundreds of spectral bands which describes the content of objects in the image. The motivation behind this study is to solve issues in Hyperspectral image classification due to the importance of Hyperspectral image in bringing growth and development for Ethiopia in the future.

1.3. Statement of the problem

In many priors works which utilizes both Euclidian and non-Euclidean structure of HSI has better performance compare to other models which utilizes either one of the forms. But these works has drawbacks when it comes to the limited number of labeled dataset for training purposes since they are supervised models which needs enormous amount of training data (*Guo F. et al, 2021*). There are other works that utilizes both structures which conduct semi-supervised classification by using GCN as node classifier. However, the drawback of this models is the computational complexity for computing adjacency matrix for graphs that contains small number of labeled nodes and much larger number of unlabeled nodes, is very high *(Hong D. et al, 2020)*. Therefore, Semi-supervised GAN are preferable when it comes to operating in small number of labeled dataset.

Semi-Supervised GANs are used for Hyperspectral image classification since there is limited availability of labeled Hyperspectral image samples for training purposes due to the expensive cost and time consuming operation of labeling each pixels in Hyperspectral image. But Semi-Supervised GANs utilize only Euclidian structure of Hyperspectral images by extracting joint spatial and spectral features before classification *(Zhu L. et al, 2018) (Feng J. et al, 2020).*

To overcome such challenging problem, "*Graph network based Generative Adversarial Network GNGAN"* model is proposed which utilizes both structures in semi-supervised GAN.

1.4. Research Questions

The following research question will be attempted to address in this research work.

RQ1. What is the effect of 3D convolutional layers as joint spatial spectral feature extraction method in semi supervised GAN as classification model?

RQ2. What is the effect of GCN layers in utilizing non-Euclidian structure of Hyperspectral image features in semi supervised GAN as classification model?

RQ3. What is the effective method to combine spatial spectral joint feature extractor and GCN in semi-supervised GAN?

1.5. Objective of the study

1.5.1. General Objective

The general objective of this study is to design and implement Hyperspectral image classification method using semi-supervised GAN that utilizes both Euclidian and non-Euclidian structure of Hyperspectral image.

1.5.2. Specific Objectives

The specific objectives of the study are listed below.

- To design a graph construction mechanism for Hyperspectral image
- To design a joint spatial spectral feature extraction method for utilizing Euclidean structure of HSI.
- To design Graph Convolution Network layers for utilizing non-Euclidian structure of HSI.
- To design a combination mechanism for combining spatial spectral feature extraction method and GCN layers

1.6. Scope and Limitations

1.6.1. Scope of the study

Based on the available time and resource, the scope of the study contains the following.

- Construct number of graphs from patches of HSI based on spectral relationship among local pixels in patches.
- Implement joint spatial spectral feature extraction mechanism in Euclidian structure of HSI in GNGAN
- Implement deep spectral features extraction mechanism in non-Euclidian structure of HSI in GNGAN.
- Implement a combination method for two feature extraction mechanism in GNGAN.
- Add Batch normalization mechanism for stable training of GNGAN.

1.6.2. Limitation of the study

The limitation faced during the study are listed below.

- The graph construction mechanism doesn't consider variable weight edge in a graph
- In the graph construction process, the spectral similarity among pixels was the only criteria for selecting neighbors for pixels but not spatial and regional similarity were considered.
- The nodes in a graphs consider to be only local because they are all from a single patch.
- The work only focuses on semi-supervised classification and not focused on generation of high quality synthetic hyperspectral image cubes.
- The generator of the GNGAN only generate HSI in the form of Euclidian grid structure and not in graph form.

1.7. Contribution of the study

The contributions of the proposed solution are two which are listed below:

- The proposed model address the long term dependencies of the spectral features in non-Euclidian structure using Graph network block in semi-supervised GAN model.

- The proposed model learns joint spatial and spectral features in Euclidian structure using Convolutional block in semi-supervised GAN model.

1.8. Beneficiaries of the study

This thesis work will benefit many parties greatly that utilize Hyperspectral images with limited amount of labeled data in any application area. Some of these areas are described in details.

- Farming and irrigation land management using remote sensing Hyperspectral image for governmental offices, large agricultural based corporations and research groups.
- Urban planning and management using remote sensing Hyperspectral image of cities and villages for city government office, real estate companies and construction companies.
- Minerals, oil and underground water explorations using remote sensing Hyperspectral image for mining companies, governments and NGOs.
- Object chemical composition study via image spectroscopy using Hyperspectral images for geological companies, researchers and chemical factories.
- Checking the quality of fruit, vegetable and other food related objects by checking the composition of the food using their Hyperspectral images for food processing factories, quality assurance governmental offices and supermarkets.
- Any other beneficiaries who use Hyperspectral image of an object in order to know the composition of the object for any specific application.

1.9. Organization of the Thesis

The organization of this thesis starting from thee next chapter, are listed as follows.

Chapter Two explains the literatures on Hyperspectral image, challenges in Hyperspectral image classifications, and different classification models in details which are used in this work. It also explains related works with their gaps.

Chapter Three explains the details of datasets used in the study, preprocessing methods, evaluation matrices and developmental tools employed in the study are discussed.

Chapter Four explains the architecture of the proposed model in detail, and the training pseudocode of the proposed model.

Chapter Five describes how the proposed solution is implemented, state the working environment and describes how the experiments were arranged.

Chapter Six describe the proposed solution findings, compare the findings with the prior work and analyze the findings in detail.

Chapter Seven concludes the thesis work and give suggestions to the further works

Chapter Two

Literature Review

2.1. Hyperspectral Image Classification

2.1.1. Hyperspectral Image

Digital image is composed of picture elements or pixels which is arranged in 2D rectangular matrix form where each pixels contains intensity information ranges from 0 to 2n in n bit of image. The quality of the image can be determined by Spatial resolution which is the amount of spatial detail in the image or physical dimension represented in a pixel.

Electromagnetic spectrum is range of wavelength or frequencies of electromagnetic radiation. Based on the spectrum information from materials which is known as spectral signature, composition of the material can be identified using the spectrum information. Spectrum information is 1D vector which consists of wavelength of range of electromagnetic spectrum. The quality of the spectral information can be determined by using Spectral resolution which is the ability of holding large number of wavelength band of electromagnetic radiation.

Spectral images is a type of image that consists of both spatial and spectral information in 3D cube where the 2D spatial information consists of pixels in rectangular form and every pixel has 1D spectral information. There are three type of spectral images which are Multispectral image, Hyperspectral image and Ultra spectral image.

Multispectral image (MSI) has 4 to 20 discrete spectral band at each pixels with medium wavelength band and Ultra spectral image (USI) has thousands continuous spectral band at each pixels with very narrow wavelength band. Hyperspectral image (HSI) has hundreds to thousands continuous spectral band at each pixels with narrow wavelength band. Digital camera that capture HSI is called hyperspectral image spectroscope which captures the Spatial-spectral data cube also known as Hypercube (*Arun P. et al, 2020*).

Hyperspectral image has two basic features which are spatial feature and spectral features. Spatial features are described in Euclidian space in the form of rectangular array just like most image types. Such features are analyzed using techniques operates in spatial domain like convolutions.

The joint spatial and spectral features is also can be expressed in Euclidian structure in 3 dimensional form. Spectral features on the other hand are expressed in non-Euclidian space in geometric structures. Graphs are the most common structures used to express Hyperspectral images in the form of nodes and edges of graphs.

HSI is used in different application areas like medicine, agriculture, military, mining, environmental protection, material science, Space exploration and so on *(Sudharsan S. et al, 2019)(Hsieh T. & Kiang J., 2020)(Guilloteau C., 2020)*. Hyperspectral images are mainly used in remote sensing, which is acquisition of information without making physical contact with the object. Hyperspectral microscopy is another area where HSI is used in microscopy to identify the composition of micro level structures using Spatial-spectral data cube.

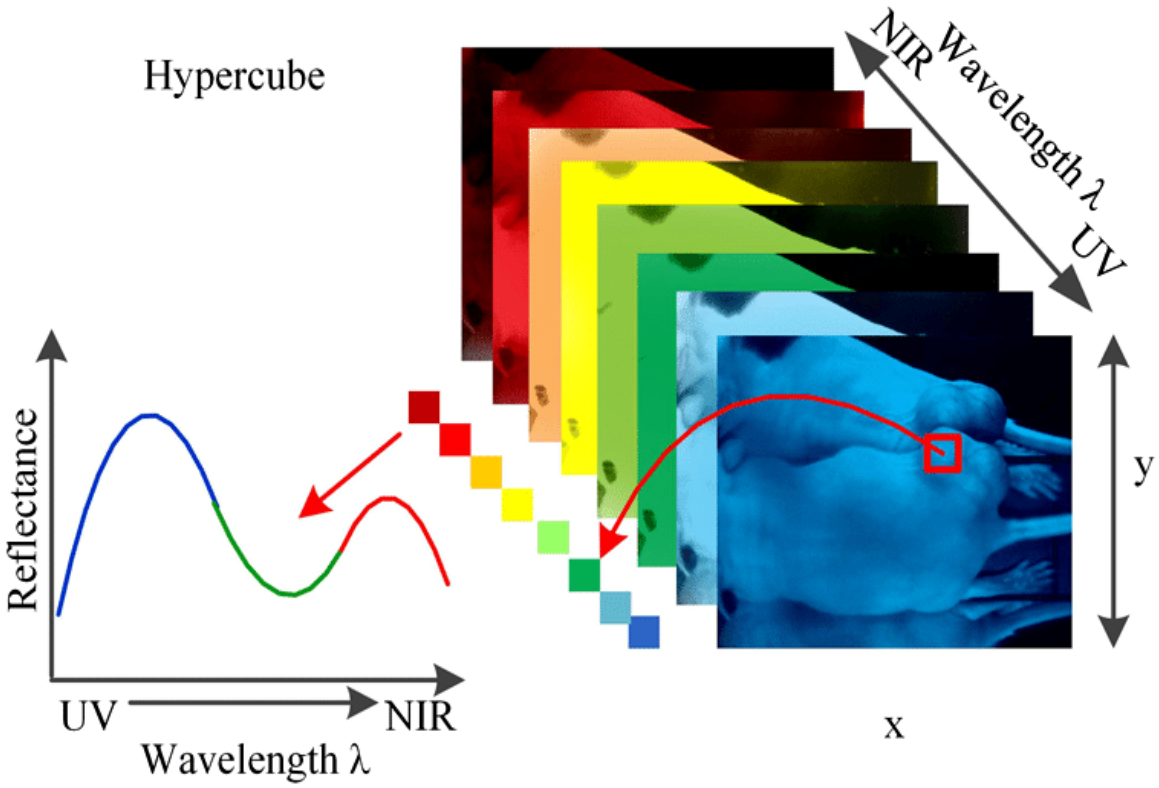


Figure 2.1 Hypercube ***taken from (Sownya V. et al, 2019)***

2.1.2. Challenges in Hyperspectral Image Classification

Hyperspectral Image Classification is pixel wise classifications which allocated pixels to different classes based on their spatial and spectral features. Since HSI classification is based on both spatial and spectral information of the image, series challenges are exhibited during HSI classification *(Lv, W., & Wang, X, 2020)* these challenges are discussed in detail below.

- **High Dimensionality:** HSI is affected with the curse of dimensionality due to having large spatial and spectral information. High dimension HSI data cube can be reduced to low dimension data by either feature extraction or feature selection using different dimension reduction techniques such as Principal Component Analysis PCA, Negative Matrix Factorization and encoding to latent space using Auto encoders *(Menilk Shalu, 2021).* Good dimensionality reduction technique should be used before image classification in order to increase the performance of the hyperspectral image classifiers.
- **Spatial variability of spectral information:** The lack of analyzing spatial context information is another challenge. Many hyperspectral image classification method analyzed either spectral feature only or spatial and spectral features separately. Two different regions in spatial domain which belongs to different classes can show same spectral characteristics whereas two different regions which belongs to the same class can have different spectral characteristics. The joint spatial spectral feature extraction for hyperspectral image classification is one method to solve HSI challenge using different mechanisms like ConvLSTM *(Y Yan et al, 2010*) and three dimensional CNN (*Ahmad M. et al, 2020*).
- **Non-Euclidian structure in HSI data:** The performance of classification model increases when all of the necessary features of the training data is learned by the classification model. Hyperspectral image not only have Euclidian grid like structure like most of image types but also non-Euclidian geometric structure. Most of the time, the irregular geometric structure can be described in the form of graph data structure *(Zhang M. et al, 2021)*. The lack of utilizing irregular structure of data may lead to reduction in the performance of classification models due to misclassifications.
- **Lack of training samples**: The performance of classification method increases with the increase of available training samples whereas the limited availability of training samples

decreases the classification performance with the increase of feature dimension and HSI effect is famously termed as Hughes phenomenon *(Mather P & Koch M, 2011)*. To overcome such problem different methods were proposed for hyperspectral image classification.

2.2. Classification Models

There are many classification model used to classify Hyperspectral image and in recent years, deep learning models were used frequently. Supervised, Unsupervised and Semi-supervised classification methods employed different deep learning models such as CNN, ConvLSTM, GCN, GAN and other models. Out of these models, the proposed model GNSGAN mainly consists of 3D Convolutional layer, 3D Transpose Convolutional layer, GCN, GAN and Semi-supervised GAN. In this section, these components are discussed in detail below.

2.2.1. Convolutional Neural Network (CNN)

CNN is a type of neural network that able to process grid like data such as image and video. This type of neural network can process from one dimensional data to three dimensional data. It consists of three basic layers which are Convolutional layer, pooling layer and Fully connected layer.

Convolutional layer conduct convolution operation to the grid like data with several kernels in order to produce feature maps. Convolution operation is the process of combining two data entities to produce one data entity. Using the convolutional layer, the size of raw data with grid like topology will be reduced to smaller size and important features will be extracted from the raw data. Pooling layer is used to reduce the size of the input data to smaller size whereas the fully connected layer is used to classify the feature extracted data via convolution layer in to class.

There are another type of CNN which is called De-convolutional neural network which is used to generate data with a grid like structure. This type of CNN consists of different layers such as Transpose Convolutional layer and up sampling layer. The input of such network is a latent space which is one dimensional tensor. Transpose convolutional layer perform the opposite operation of Convolutional layer by constructing feature out of a feature map in to larger size tensors.

Hyperspectral image has 3 dimensional tensor structure which is suitable to be processed with CNN for considering the process of joint spatial spectral features. Many CNN based models has been developed for HSI classification and they are mostly used in supervised based classification methods (*Ahmad M, et al, 2022*) since CNN model need a lot of data quantity for training purposes.

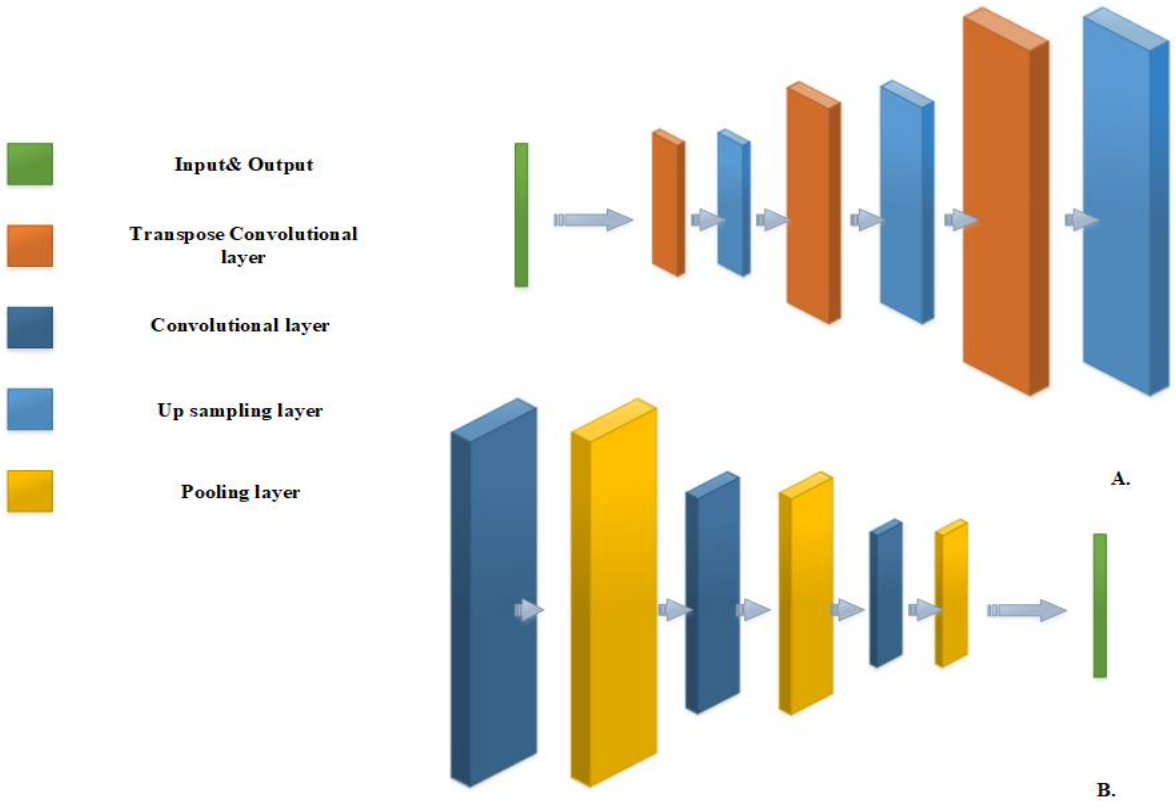


Figure 2.2: CNN: A) Transpose Convolutional and Up sampling layers. B) Convolutional and Pooling layers

2.2.2. Graphs and Graph Neural Network

Graphs are nonlinear data structures that are made up of entities called nodes and their relationship called edges. Many real world data are represented in graph forms like social network, protein to protein interaction network and even pixel relationship network of Hyperspectral image. Nodes in graphs are entities which contains their on node attribute. In Hyperspectral image, each pixel represent a node and the corresponding spectral vector is its node attribute. Edge is the relationship

between two nodes and it can be weighted or not. Graph G can be represented mathematically by the following equation.

$$\boldsymbol{G = (V, E)} \quad \textbf{..... Eq. (2.1)}$$

V is a set of vertices or nodes whereas **E** is a set of edges in the graph. Each node can be describes using their node features and edges in the graph are described using an **Adjacency Matrix (A)** for a single graph.

$$\boldsymbol{A(i,j) = \begin{cases} 1 \; if \; \{v_i, v_j\} \in E \; and \; i \neq j \\ 0 \; otherwise \end{cases}} \quad \textbf{..... Eq. (2.2)}$$

The number of nodes and its neighbors can be described using a Diagonal matrix of the graph **D** and matrixes called Laplacian Matrix **L** and Symmetric normalized Laplacian Matrix $\mathbf{L_{sym}}$ relates to many useful properties of a graph.

$$\boldsymbol{D(i,j) = \begin{cases} d(v_i) \; for \; i = j \\ 0 \; otherwise \end{cases}} \quad \textbf{..... Eq. (2.3)}$$

$$\boldsymbol{L = D - A} \quad \textbf{..... Eq. (2.4)}$$

$$\boldsymbol{Lsym = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}} \quad \textbf{..... Eq. (2.5)}$$

Most deep learning models such as CNN, RNN and GAN takes input data with tensor form which is a linear or Euclidean data structure. Due to this reason, Graph neural network GNN was proposed by (*Scarselli, F et al, 2008*) that takes data with graph data structure. Since that time, several type of GNN models has been proposed. Some of these models are: Graph Convolutional Network (GCN), Graph Recurrent Network (GRU), Graph Attention Network (GAT) and Graph Residual Network (GRN) *(Liu Z. & Zhou J., 2020)*.

2.2.3. Graph Convolutional Network

Graph Convolutional network is one type of Graph neural network which operates convolutional operation on graph domain. There are two approaches in which the GCN conduct convolutional operation in graph domain which are spectral and spatial approaches. In spatial approach, the convolution operation is directly applied on the graph where as in spectral approach, the

convolution operation is applied in spectral domain after the graph is transformed in spectral form using graph Fourier transform.

$$\boldsymbol{F(g * x) = GX} \text{ ..... Eq. (2.6)}$$

The Fourier transform of the convolution of two elements is the multiplication of the Fourier transform of the elements in the spectral domain. Due to that, the convolution of the filter with the node matrix of the graph results with the multiplication of the filter in spectral domain with the node matrix in spectral domain.

In this research, spectral type of GCN was used which was proposed by *(Kipf, T. N., & Welling, M. 2016).* The graph convolution is defined in Fourier transform by computing Eigen decomposition of a graph laplacian **L** matrix. The operation can be defined in the following equations.

$$\mathbf{g(\theta) * X = U g(\theta)(\wedge) U^T X} \text{ ..... Eq. (2.7)}$$

$$\boldsymbol{L = U \wedge U^T} \text{ ..... Eq. (2.8)}$$

U is the matrix of Eigen vector of normalized graph laplacian **L** which was defined in equation 2.5. **X** is the node matrix whereas **g** is the filter. The propagation rule of hidden layer in the GCN can be calculated by the following equation.

$$\boldsymbol{H^{l+1} = \sigma\left(L^l_{sym} H^l W^l + b^l\right)} \text{ ..... Eq. (2.9)}$$

W is the weight matrix, **b** is the bias matrix and **H** is the hidden layer output matrix where **l** is the number of layer.

GCN is used in two ways, the first one is a **node classifier** which classify nodes after training with labeled nodes from single graph. The second way is as **graph classifier** from collection of graphs that has ground truth for each graph. In this research, the GCN is used as a graph classifier.

In this research, GCN was used because it utilize graphs that was constructed from the pixels of Hyperspectral image in a patch as nodes and the relationship between the pixels in the similarity of the spectral signature as edges. GCN can uncover the relationship between a pixel and its neighbors using graph structure which is mentioned as one of the problem mentioned in statement of the problem section of the research.

2.2.4. Generative Adversarial Network (GAN)

Generative Adversarial Networks are deep learning models designed by Ian Goodfellow and his colleagues in 2014 (*Goodfellow I. et al, 2014*). It works by setting two separate neural network models in competition with each other. These two ANN models are Generative and Discriminative networks, the generative network learns to generate fake samples from noise vector and the Discriminative network learns to figure out either the generated sample is authentic or not. The goal of GAN is to generate near authentic samples through competition between the generative network and the discriminative network where the discriminative network cannot distinguish the generated sample as generated ones. If the real and generated samples are images, the GAN is called Deep Convolutional Generative Adversarial Network DCGAN.

In DCGAN, both the generator and discriminator are constructed from Convolutional Neural Network CNN where the generator is composed of transpose Convolutional (De-convolutional) and up sampling layers and the discriminator is composed of Convolutional, Pooling and Fully connected layers. The input of the generator is one dimensional noise vector and the output of the generator is generated image. The input of the discriminator is real image samples and the generated image samples and the output of the discriminator is classification labels whether the image is real or fake (generated). For hyperspectral images, 3D convolutional and deconvolution layer are used to construct discriminator and generator of the GAN. ConvLSTM, Attention Mechanisms, Auto encoders and other deep learning model can be used to construct the generator and discriminator of the GANs.

GANs is hard to train due to three major challenges which are: Mode collapse, Non-Convergence and Vanishing Gradients. Mode collapse happens when the generator generates samples with little representation of populations and it is a result of poor generalization. Non-Convergence in GANs refers to not able to find equilibrium between Generator and Discriminator since they are competing with each other.

Because GANs are made up of two networks, each with its own loss function, they are intrinsically unstable. The gradient vanishing problem can be caused by the Generator (G) loss when the Discriminator (D) can easily discern between real and fake samples. While the G seeks to enhance it, the D seeks to decrease a cross-entropy. When D confidence is strong and it begins to reject the

samples that G generates, the gradient of G disappears. These cause the GAN to become unstable, thus G and D should be in balance.

To mitigate such GAN failures, different solutions were proposed to increase stability of the model, to eliminate gradient vanishing, and mode collapse problems during training. Conditional GAN (*Mirza M & Osindero S, 2014*) were proposed to overcome mode collapse problem by using conditional inputs in the generator and discriminators. Wasserstein GAN (WGAN) (*Arjovsky M. et al, 2017*) were proposed for overcoming instability and vanishing gradient by using a discriminator model that score the realness of the generated image using Wasserstein loss function. Spectral normalization (*Miyato T. et al, 2018*) is weight normalization technique that can stabilize the training of discriminator network by tuning Lipschitz constant which is a hyper parameter. Two time-scale update rule TTUR **(***Heusel M. et al, 2017*) was proposed to overcome non-convergence problem by using two different learning rates for the generator and discriminator. TTUR is also used to make the GAN training faster.

2.2.5. Semi-supervised Generative Adversarial Network

Semi-supervised GAN was proposed by *(Odena A, 2016)* which is used as classifier network for classifying data in to multiple classes. This type of GAN used small amount of labeled data from a large amount of unlabeled data to be trained. The function of the Generator is to generate fake data from the noise input and feed to the Discriminator. The Discriminator on the other hand is used to discriminate the real unlabeled data as real and the generated data as a fake. But the Discriminator also has other function which is to classify the labeled real data in to multiple classes.

The discriminate act like normal discriminate when it was fed fake and unlabeled real data and act like classifier when it is fed labeled real data. The feature extraction part of the discriminator is a shared part for both modes whereas the last part of the discriminator is different for both modes. During supervised mode where the discriminator acts like classifier, the last layer of the discriminator is Softmax layer and during unsupervised mode where the discriminator acts like normal discriminator in normal GAN, the last layer is a customized layer which produce 0 and 1 which represents real and fake. The loss of the supervised discriminator can be given in the following equation. Where y is the target output and y prime is the predicted output.

The lack of large number of labeled Hyperspectral image dataset is a main problem for the classification of Hyperspectral image. Therefore, Semi-supervised GAN has been used recently to mitigate this problem by using small amount of labeled Hyperspectral and large amount of unlabeled Hyperspectral image in order to train the GAN for Hyperspectral image classification (*Zhang F et al, 2020*), (*Zhan Y et al, 2017*), (*Xie W et al, 2021*),

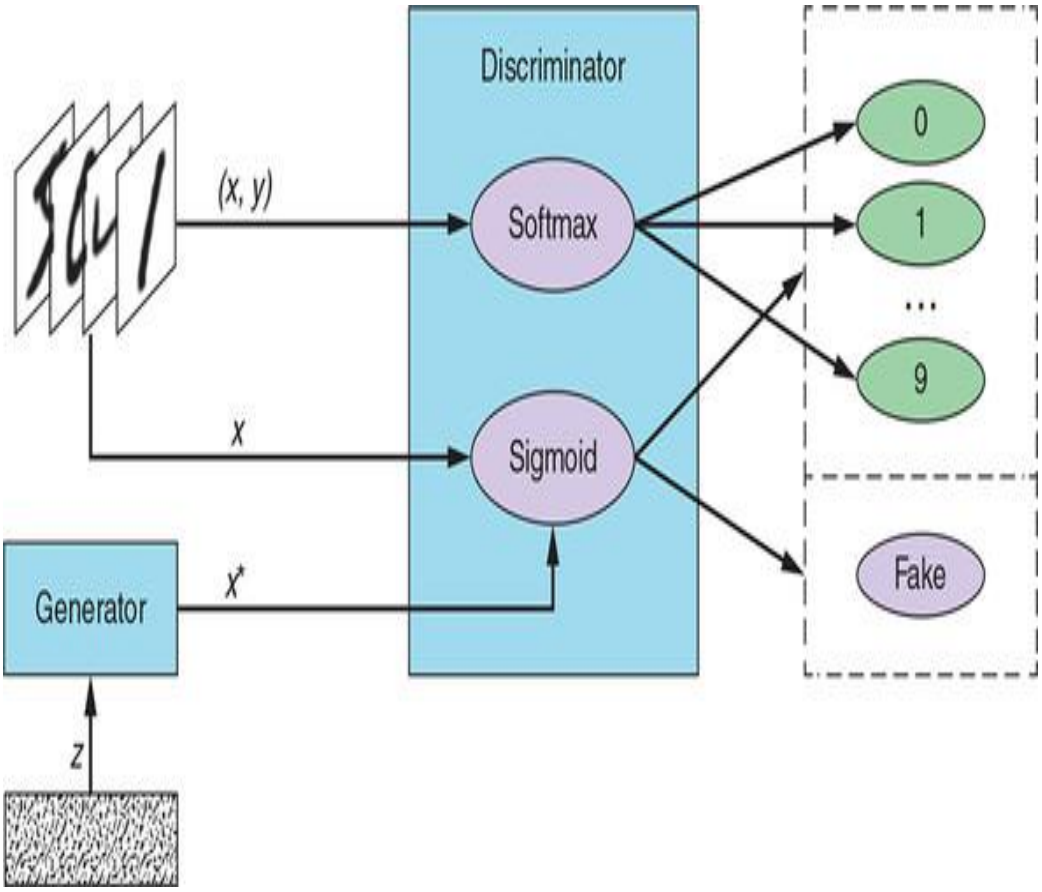


Figure 2.3: Semi-supervised GAN *taken from (Jakub L & Vladimir B, 2019)*

2.3. Related Work

In the previous sections, challenges in Hyperspectral image classification and classification models were discussed. In this section, prior works that address similar problems which are stated in statement of the problem section are discussed below.

The authors of *(Zhu L et al, 2018)* address the issue of limited amount of labeled data in HSI by using semi-supervised 3D GAN. The discriminator of the GAN was constructed using 3D CNN so that the spatial spectral features of Euclidian structure of HSI can be utilized. But, the model lacks the means of addressing the issues of long term spectral dependencies and utilizing non-Euclidean structure of HSI.

The author of *(Feng J. et al, 2020)* proposed CA-GAN which the discriminator and generator not only compute with each other but also collaborate together for generating quality HSI with multiple classes. The generator has a hard attention mechanisms and the discriminator is constructed with 3D CNN and ConvLSTM. The model address the issue of small number of labeled data for training and utilize both long term spectral dependencies and spatial spectral features in Euclidean structure of HSI. But the model doesn't address the utilization of non-Euclidean structure of HIS which contains deep spectral features than in Euclidian structure.

The authors of *(Liang H. et al, 2021)* proposed Adaptive Weight Feature Fusion GAN or AWF^2-GAN which is composed of discriminator with separate feature extractor network for spectral and spatial separately and fuse them using adaptive fusion at the end of the network. The model is used for addressing the limited number of labeled data for the training using semi-supervised learning and utilizing both spectral long term dependencies and spatial spectral feature extraction for Euclidean structure of HSI. But the model doesn't address the utilization of non-Euclidean structure of HIS which contains deep spectral features than in Euclidian structure.

The authors *(Hong D. et al, 2020)* proposed a fusion of 2D CNN network and mini batch GCN for HSI classification. The model able to minimize the time complexity for the adjacency matrix calculation of entire HSI cube during graph generation before patching process. The generated graph then patched in mini batch format for minimizing time complexity. The proposed model address the utilizing of non-Euclidian structure of HSI and the spectral long term dependencies via the mini batch GCN network and spatial feature extraction by the 2D CNN network. But the model lacks to consider joint spatial spectral feature extractions in Euclidian structure format and it lacks the means of addressing the small amount of labeled data for training purposes.

The authors *(Yang P. et al, 2020)* proposed GCN model for semi-supervised HSI classification which addresses the utilization of non-Euclidian structure of HSI and uses graph sampling and

aggregation of graphs for node classification. But the time and space complexity of for computing Adjacency matrix is high and there is no means to utilize Euclidian structure of HSI in the model.

The authors *(Zhang M. et al, 2021)* proposed Offset GCN model for spatial spectral HSI classification which address the utilization of non-Euclidian structure of HSI and the spatial spectral feature extractions and spectral long term dependencies issues. The model also address the small amount of labeled data for the training by using semi-supervised learning via GCN. But the model lacks the means of addressing Euclidian structure of HSI for classification which can bring misclassification.

2.3.1: Summary of Related works

Four type of prior works are considered in the related work section which are Hyperspectral image classification using Semi-supervised GAN models, GCN based models with and without semi-supervised model, and combination model that utilizes both Euclidian and non-Euclidian model without semi-supervised learning. The first type of works utilizes only Euclidian structure of HSI and their gaps is the lack of method for utilizing non-Euclidian structure of HSI which cause misclassifications. The second type of works utilizes only non-Euclidian structure of HSI and their gaps are the lack of utilizing Euclidian structure of HSI and it needs high time complexity for computing adjacency matrix on entire graph for node classification. The third type of works utilizes only non-Euclidian structure of HSI and their gaps are the lack of utilizing Euclidian structure of HSI and it needs large amount of training sample since it is supervised model. The last type of work utilizes both Euclidian and non-Euclidian structure of HSI and their gaps is the need of large amount of training sample since it is supervised model.

Table 2.1. Summary of the related works

Paper	Datasets	Architecture	Evaluation matrices	Gap
(Zhu L.et al, 2018)	▪ Indian Pines ▪ Salina Valley ▪ Kennedy Space Center	3D Generative Adversarial Network	▪ Average accuracy ▪ Overall accuracy ▪ Kappa coefficient	There is no mechanism for utilizing non-Euclidian structures in HSI
(Feng J et al, 2020)	▪ Indian Pines ▪ Pavia University ▪ Washington	CA-GAN or GAN based collaborative learning and Attention mechanism	▪ Average accuracy ▪ Overall accuracy ▪ Kappa coefficient	There is no mechanism for utilizing non-Euclidian structures in HSI
(Liang H. et al, 2021)	▪ Indian Pines ▪ Pavia University	AWF^2-GAN or Adaptive Weight Feature Fusion GAN	▪ Average accuracy ▪ Overall accuracy ▪ Kappa coefficient	There is no mechanism for utilizing non-Euclidian structures in HSI
(Hong D et al, 2020)	▪ Indian Pines ▪ Pavia University ▪ Houston 2013	FuNet-C or Fusion on Concatenation of CNN	▪ Average accuracy ▪ Overall accuracy	▪ Not working well with small amount of training samples.

		with mini batch Graph Convolutional Network	▪ Kappa coefficient	▪ There is no mechanism to extract joint spatial and spectral features in Euclidian structure.
(Zhang M. et al, 2021)	▪ Indian Pines ▪ Pavia University ▪ Salina Valley	(SSOGCN) spectral-spatial offset Graph convolutional network	▪ Average accuracy ▪ Overall accuracy ▪ Kappa coefficient	There is no mechanism for utilizing Euclidian structures in HSI
(Yang P. et al, 2018)	▪ Indian Pines ▪ Pavia University ▪ Kennedy Space Center	GCN with GraphSAGE	▪ Average accuracy ▪ Overall accuracy ▪ Kappa coefficient	▪ There is no mechanism for utilizing Euclidian structures in HSI ▪ High time complexity for computing adjacency matrix

Chapter Three

Research Methodology

3.1. Chapter Overview

In this chapter, the methodology of the research in Graph convolution based GAN model for Hyperspectral image classification consists of detail descriptions of Development tools, Datasets, Data Preprocessing methods and Evaluation matrixes which were used in this research.

3.2. Development tools

Several development tools were used to design and implement the proposed model as well as producing the thesis document. These tools can be categorized as software tools, hardware tools and documentation tools.

- **Hardware tools**

The following Hardware tools were used in this research.

Table 3.1: Hardware tools used in this research

No	Tools	Function
1	GPU	Increase the performance of the computer in order to conduct massive mathematical computing such as training deep learning model

- **Software tools**

The following Libraries and IDE tools were used in this research to implement the proposed model. The programming language used in all implementation is Python 3.7.13 version and basic libraries such as numpy, os, scipy and other were used. The main libraries and IDE tools are described below.

Table 3.2: Software Tools

No	Tools	Version	Function
Library and Framework			
1	Tensorflow	2.8.2	Deep learning framework for defining and training deep learning models
2	Keras		Library for deep learning models that is used along with Tensorflow.
3	Spectral python	0.21	Python module used for reading, displaying, manipulating, and classifying hyperspectral imagery.
4	Spektral	1.0	Library for graph deep learning based on Keras and Tensorflow
IDE and platform			
5	Jupyter notebook	6.3.0	One of IDE for python which are widely used for AI development.
6	Google Colaboratory	--	Product from Google Research used for writing and executing python code and provide GPU and TPU computing resource for deep learning model training.

3.3. Dataset

In this research, two datasets were used for training and testing the proposed model which are Salina Valley and Pavia University[1]. Although the dataset were produced in 1990's, they are standard Hyperspectral image dataset and were used in most Hyperspectral image based researches. Each dataset consists of one 3D Hyperspectral image cube that contains large amount of pixels along with spectral signatures and 2D ground truth for each pixel.

[1] Salina Valley and Pavia University dataset found at https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes

3.3.1. Salina Valley

The 224-band AVIRIS sensor was used to gather the Salinas Valley data set over the Salinas Valley in California. There are 16 ground-truth classes in this 512 row by 217 column data collection. There are 204 spectral bands maintained with 111,104 pixels after the water absorption bands and noise-affected bands are removed, and 54,129 foreground pixels are left over after the background data is removed. The ground truth matrix measures 512 by 217 pixels.

Table 3.3: Salina Valley

No	Classes	Number of samples (Pixels)
1	Broccoli green weeds 1	2009
2	Broccoli green weeds 2	3726
3	Fallow	1976
4	Fallow rough plow	1394
5	Fallow smooth	2678
6	Stubble	3959
7	Celery	3579
8	Grapes untrained	11271
9	Soil vineyard develop	6203
10	Corn senesced green weeds	3278
11	Lettuce romaine 4wk	1068
12	Lettuce romaine 5wk	1924
13	Lettuce romaine 6wk	916
14	Lettuce romaine 7wk	1070
15	Vineyard untrained	7268
16	Vineyard vertical trellis	1807
Total		54,129

3.3.2. Pavia University

The University of Pavia data set was collected over University of Pavia, Northern Italy, by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor. After deleting many noise-corrupted bands, there are 103 spectral bands in the spectrum range of 0.43 to 0.86 m. The image is 610 340 pixels in size, has a spatial resolution

of 1.3 m/p, contains 207,400 pixels, and has 42,776 foreground pixels after background data has been removed.

Table 3.4 Pavia University

No	Class	Number of samples (pixels)
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Trees	3064
5	Painted metal sheets	1354
6	Bare soil	5029
7	Bitumen	1330
8	Self-blocking bricks	3682
9	Shadows	947
Total		42,776

3.4. Data Preprocessing

In this research, four data preprocessing methods were employed to make the raw dataset in to two forms which are patched multiple Hypercube and patched Graph structure based representation of the Hypercube. The four data preprocessing methods are: Dimensionality reduction technique using Principal Component Analysis (PCA), Patching, Graph construction and Data balancing. .

3.4.1. Principal Component Analysis

Dimensional reduction is the transformation of high dimensional data to low dimensional data by reducing the data attribute. There are several dimensional reduction techniques employed in many research works and Principal Component Analysis (PCA) is one of them. In this research, the spectral bands were reduced to 50 in all three datasets using the PCA.

Table 3.5: PCA operation on all datasets

No	Dataset	Size before PCA	Size after PCA
1	Salina Valley	512 x 217 x 204	512 x 217 x 50
2	Pavia University	610 x 140 x 103	610 x 140 x 50

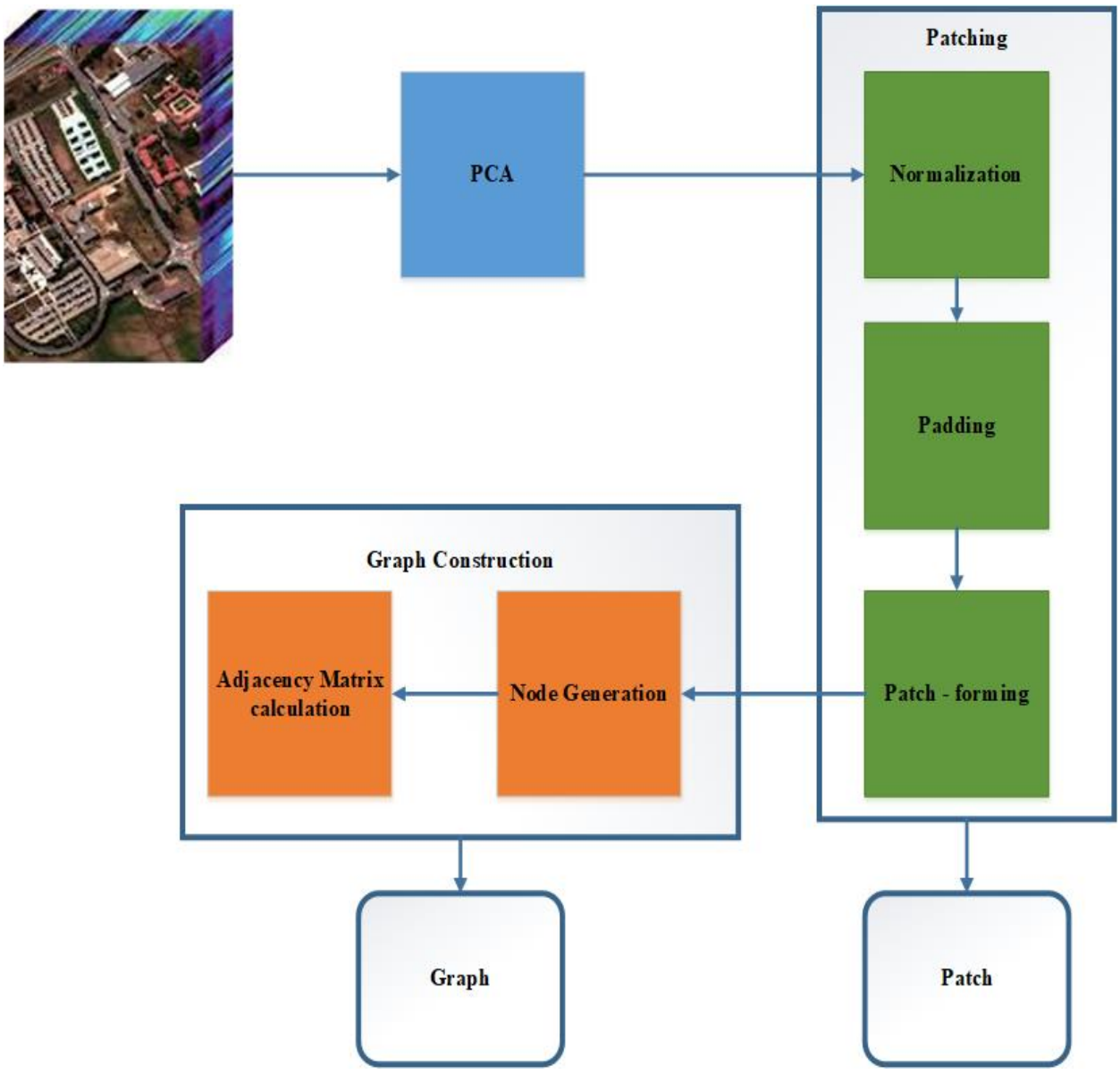


Figure 3.1: Data preprocessing methods

3.4.2. Patching

Patching is a process of converting one large image in to multiple smaller size patches that is a portion of the original image. In this research, all three datasets were converted in to multiple patches with the size of **7 x 7 x 50** each and the length and width of a patch is known as **window size**. The patching process takes place in three consecutive steps which are: **Normalization, Padding and Patch forming**. The first step is Normalization which is used to normalize the value

of each band in the multiple pixels between 0 and 1 for shortening the training time of the model. In this step, the size of the original Hypercube was not changed.

The second step is to add padding around the Hypercube with the same margin in the four sides. The margin is calculated as:

$$\textbf{Margin} = \frac{(\boldsymbol{WindowSize-1})}{\mathbf{2}} \quad \textbf{..... Eq. (3.1)}$$

The main purpose of the padding is to form a patch with the same size for all the pixels in the Hypercube.

The last step is to form the patch from the Hypercube. Each patch consists of 49 pixels and from those pixels, one pixel in the middle served as the main pixel whereas the rest 48 pixels served as neighbors for the main pixel. The label assigned to a patch is the same with the label assigned to the middle pixel. Any middle pixel may have a minimum of 15 real pixel neighbors if it found at the boundary of the hypercube. The number of patches produced during this step is the same with the number of pixels in the Hypercube.

Table 3.6: Size of Patches for all datasets

No	Dataset	No of pixels	Size of Patches	Size of label matrix
1	Salina Valley	54,129	54129 x 7 x 7 x 50	54129 x 1
2	Pavia University	42,776	42776 x 7 x 7 x 50	42776 x 1

3.4.3. Graph Construction

The link between two items is represented by a graph, an abstract notation. A graph is made up of vertices, also known as nodes, or nodes that are connected to one another by edges. The majority of the time, hyperspectral images are utilized in the classification process as Tensors, but more recently, hyperspectral images are employed as Graphs. Two processes, Graph node creation and Adjacency Matrix calculation, were used to convert patches of Hyperspectral picture into Graph form.

- **Graph node Construction**

  In this stage, each pixel in a patch was assigned as a node in a graph which consists of all 49 pixels as nodes. Each node contains node features with the size of 50.

- **Adjacency Matrix Calculation**

  In this stage, 15 nearest neighbor nodes to the central node were selected using Manhattans distance as a measuring tool to calculate the adjacency matrix based on their spectral signature values. Including the central node, there are 16 nodes which are interconnected with each other using Edges with the weight of 1 for simplicity purpose. The size of adjacency matrix for each graph is **49 x 49**. The Manhattans distance is given in the following equations.

$$D_{m,n} = \|E_m - E_n\|^1 \text{..... Eq. (3.2)}$$

Table 3.7: Node matrix and Adjacency matrix size in all datasets

	Salina valley	Pavia University
Node size	54129 x 49 x 50	42776 x 49 x 50
Adjacency Matrix size	54129 x 49 x 49	42776 x 49 x 49

3.5. Evaluation Matrices

In this research, three evaluation matrices were used to evaluate the proposed model using test data. These matrices are Overall accuracy (OA), Average accuracy (AA) and Kappa Coefficient (K). They are mostly used in Hyperspectral .image classification researches and all of them are derived from confusion matrix.

- **Overall Accuracy:** It is calculated by summing correctly classified values and divided by total number of values.

$$\text{OA} = \frac{\sum_{i=0}^{n} C_{i,i}}{\sum_{i=0}^{n}\sum_{j=0}^{n} C_{i,j}} \text{ ..... Eq. (3.3)}$$

- **Average Accuracy and Each Accuracy:** It is calculated by summing the accuracy of each class and divided by the number of class

$$\text{AA} = \frac{\sum_{i=0}^{n} C_{i,i}}{n} \text{ .....Eq. (3.4)}$$

Each class has its own accuracy and average accuracy is their mean.

- **Kappa Coefficient:** It is a measurement to determine agreement between classification values and truth values.

$$\mathbf{K} = \frac{M\sum_{i=0}^{n} C_{i,i} - \sum_{i=0}^{n}(Gi*Hi)}{M^2 - \sum_{i=0}^{n}(Gi*Hi)} \quad \textbf{.....Eq. (3.5)}$$

M Total number of classified value compared to truth value

G Total number of classified value belong to class i

H Total number of truth value belongs to class i

Chapter Four

Proposed Model Architecture

4.1. Chapter Overview

In this chapter, the details structure of the proposed model which is Graph Network based Generative Adversarial Network is described in detail. The proposed model is a semi-supervised classifier model with supervised and unsupervised mode and contains three basic parts which are Generator, Discriminator in supervised mode and Discriminator in unsupervised mode. The two Discriminator networks share common 3D Convolutional block which is trained in both mode. All the concepts mention above are discussed in this chapter below. It contains the overall proposed model structure, Generator network structure, Discriminator network structure and finally Training Pseudocode is discussed in this chapter.

4.2. Graph Network Based Generative Adversarial Network

Graph Network based Generative Adversarial Network (GNGAN) is the proposed model which able to be used in a situation where there is small number of labeled data and it also able to capture not only Euclidian structures but also non-Euclidian structure of different local regions in HSI data for the classification purpose.

The GNGAN is able to train with both labeled and unlabeled datasets in supervised and unsupervised mode respectively. It has one generator which generates HSI cubes in Euclidian form by taking noise vector as input. It has a discriminator which produce two outputs based on the mode in which the discriminator operates in. If the discriminator operates in unsupervised mode, it takes two inputs which are: generated HSI cubes and real unlabeled HSI patches and distinguish them as real or fake like discriminator in vanilla GAN do. In this mode, the layers of the discriminator trained on the features of real and generated HSI cubes are in Euclidian form.

If the discriminator operates in supervised mode, it takes two inputs which are: labeled HSI cubes in Euclidian form and labeled graphs in non-Euclidian form. By takes these inputs, the discriminator classify each inputs to a corresponding class. There are parts of the discriminator

which operates in both unsupervised and supervised modes since both use HSI in Euclidian form and there are parts of the discriminator which only operates in supervised mode.

The part of discriminator which operates in both modes is called Convolutional block. This block is composed of 3D Convolutional layers, Global Average pooling layer and Batchnormalization layers along with Leaky Relu as activation layer. This block learns HSI features in Euclidian form from both modes which avoid the need for large number of labeled HSI samples. Convolutional block is mainly used as joint spatial spectral feature extraction mechanism for utilizing Euclidian structure of HSI. The output of the Convolutional block is applied to number of fully connected layers with sigmoid activation function at the last layer during unsupervised mode. The architecture, hyper parameters, and the reason for selecting such layers in the block are discussed in Convolutional block section.

Part of discriminator which operates in only in supervised mode is called Graph network block. Graph block is composed of GCN layers, and Global Attention Pooling layer. This block learns HSI features in non-Euclidian structure in the form of graphs. Graph network block is used as spectral feature extraction mechanism in graph form for the GNGAN. The output of the Graph network block combines with the output of convolutional block with concatenation mechanism and their combination applied to fully connected or dense layers with Softmax activation at the last layer in supervised mode. The architecture, hyper parameters and reason for selecting such layers and parameters will be discussed in Graph network block section.

4.3. Generator Network

The main purpose of the generator in the GNGAN is for training the Convolutional block by providing generated HSI spatial and spectral features. In other GAN models, the generator is the main part of the GAN and the discriminator is used just to train the generator and discarded after the training is finished. But in this model, the discriminator is the important part and generator is used for training purposes and after the training is over, it will be discarded.

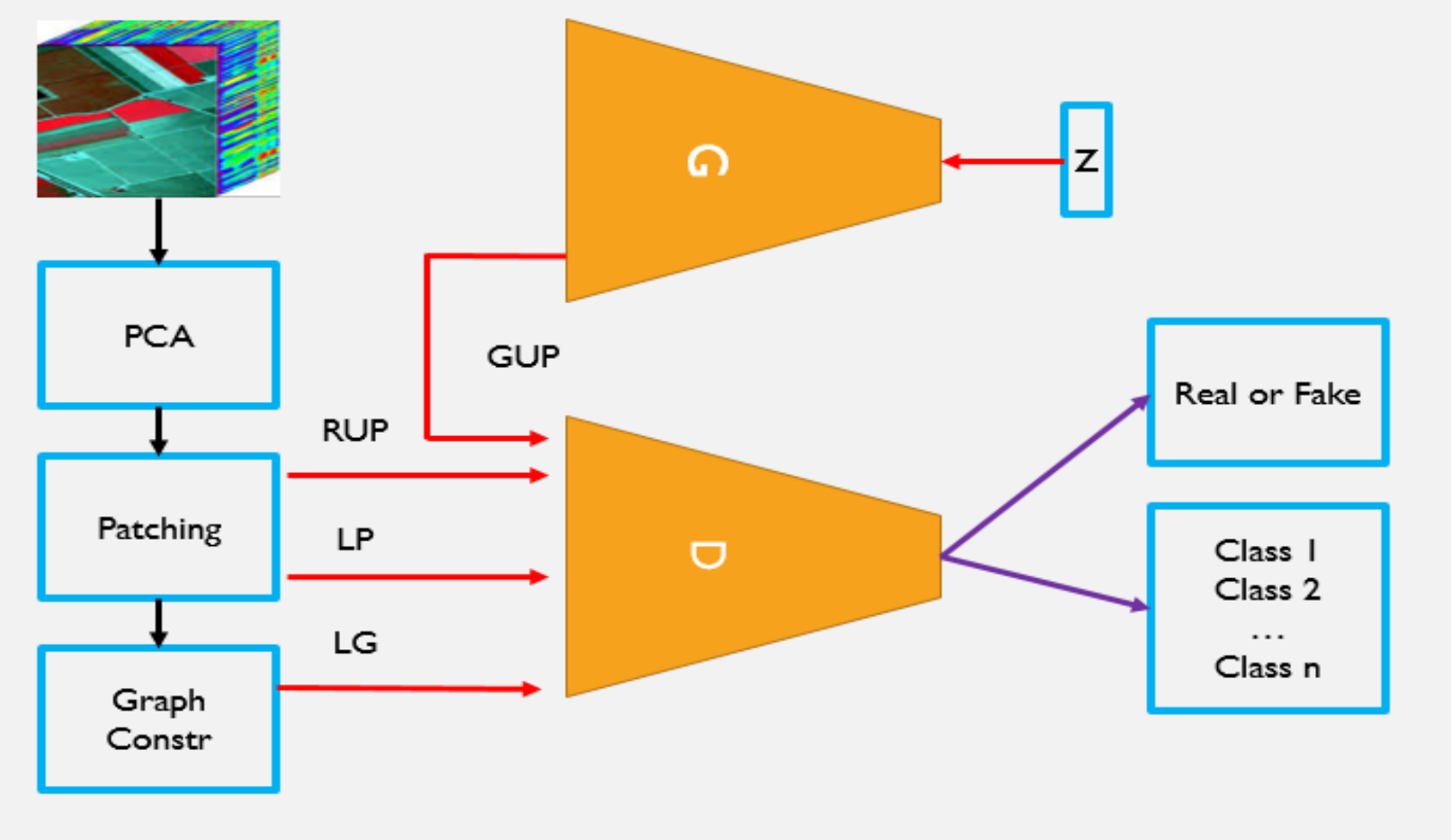


Figure 4.1: Block Diagram of the proposed Model GNGAN

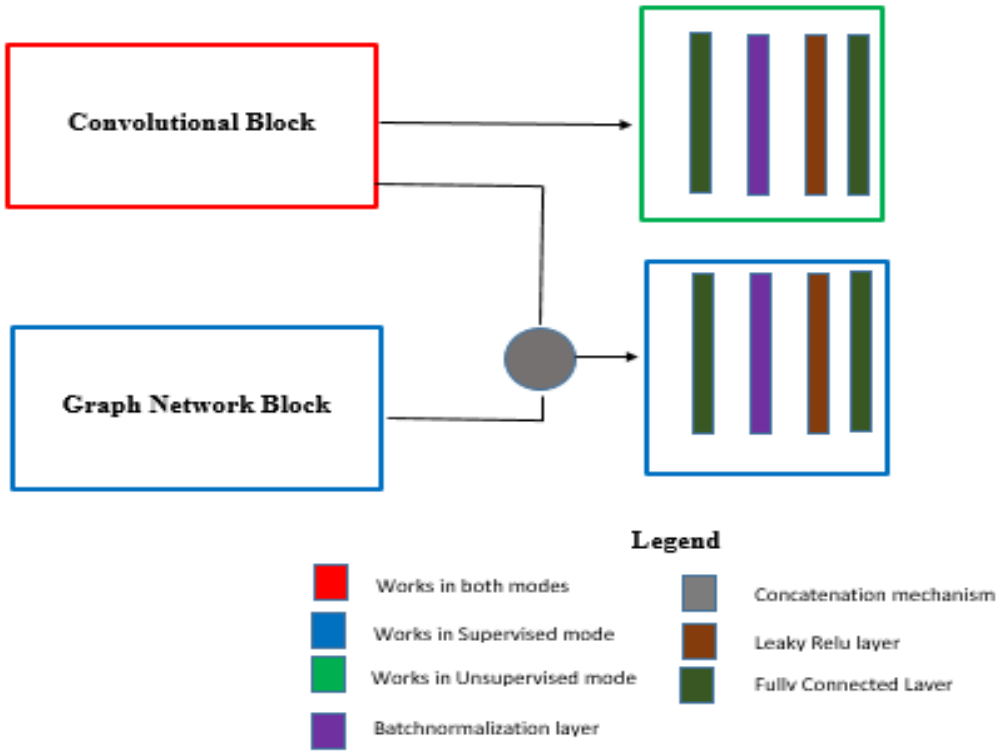


Figure 4.2: Architecture of Discriminator of GNGAN

The generator in the GNGAN consists of 14 layers with the input noise size of (100) to the output HSI size of (7, 7, 50) hypercube. The layers in the generator varies from 3D transpose Convolutional layers to the dense layers. In order to regulate the generator model, batch normalization layers are also used. The following table summarizes the structure of the generator.

Table 4.1: Layers of the Generator

Layers	Hyper parameter	Input size	Output size
Input layer		100	-
Dense	Units = 128	100	128
Batch norm		128	128
Leaky Relu act	Alpha = 0.01	128	128
Reshape		128	(1, 1, 2, 64)
Conv transpose	Filter = 32, kernels = (3, 3, 7), stride = 1	(1, 1, 2, 64)	(3, 3, 8, 32)
Batch norm		(3, 3, 8, 32)	(3, 3, 8, 32)
Leaky Relu act	Alpha = 0.01	(3, 3, 8, 32)	(3, 3, 8, 32)
Conv transpose	Filter = 16, kernels = (3, 3, 9), stride = 1	(3, 3, 8, 32)	(5, 5, 16, 16)
Batch norm		(5, 5, 16, 16)	(5, 5, 16, 16)
Leaky Relu act	Alpha = 0.01	(5, 5, 16, 16)	(5, 5, 16, 16)
Conv transpose	Filter = 8, kernels = (3, 3, 15), stride = 1	(5, 5, 16, 16)	(7, 7, 30, 8)
Batch norm		(7, 7, 30, 8)	(7, 7, 30, 8)
Leaky Relu act	Alpha = 0.01	(7, 7, 30, 8)	(7, 7, 30, 8)
Conv transpose	Filter = 1, kernels = (3, 3, 21), stride = 1, activation = sigmoid	(7, 7, 30, 8)	(7, 7, 50, 1)

Transpose Convolutional layers are used to construct feature maps with greater spatial and spectral dimension from smaller input feature maps. The hyper parameters of the transpose convolutional layers were fixed in order to match the Convolutional block in reverse way. The last layer of the generator which is transpose convolutional layer, has an activation function of sigmoid in order to limit the values of elements in the generated tensor in range between 0 and 1. This is necessary

because the elements in HSI patches are also limited in range between 0 and 1 during preprocessing using normalization technique in patching process.

The Batchnormalization and Leaky Relu layers was used in order to stabilize the generator during training. These layers were added after trying to train GNGAN and the loss of the generator and others become "*nan*". Leaky Relu activation was used due to two reason which are: to able Batchnormalization layer operates before activation function was applied to the output of transpose convolutional layers and compared to Relu activation, the GNGAN training was more stabilized.

The input of the generator network is a noise vector that generated using normal random distribution between 0 and 1 in the size of (100) vector. The equation of the normal distribution is given below.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad \text{..... Eq. (4.1)}$$

4.4. Discriminator Network

The discriminator of GNGAN consists of Convolutional block, Graph network block, Concatenation mechanism, and fully connected layer for each corresponding mode. In this section each components architecture and hyper parameter selection criteria are discussed in detail

4.4.1. Convolutional block

The Convolutional block takes real labeled and unlabeled HSI patches and generated HSI cube as input with the size of (7, 7, 50) since it is trained in both supervised and unsupervised modes. The output of Convolutional block is a single column tensor with the size of (128). This block is composed of 3D convolutional layers, Global Average Pooling layer and Batchnormalization layer along with Leaky Relu as activation layer. The following table summarize the details of the layer of the convolutional block

Table 4.2. layers in Convolutional block

Layers	Hyper Parameter	Input	Output
Input		(7, 7, 50, 1)	
Conv 3D	Filters = 8, kernel = (1, 1, 21), stride= 1	(7, 7, 50, 1)	(7, 7, 30, 8)

Batch norm		(7, 7, 30, 8)	(7, 7, 30, 8)
Leaky Relu	alpha=0.01	(7, 7, 30, 8)	(7, 7, 30, 8)
Conv 3D	Filters = 16, kernel= (1, 1, 15), stride= 1	(7, 7, 30, 8)	(5, 5, 16, 16)
Batch norm		(5, 5, 16, 16)	(5, 5, 16, 16)
Leaky Relu	alpha=0.01	(5, 5, 16, 16)	(5, 5, 16, 16)
Conv 3D	Filters = 32, kernel= (1, 1, 9), stride= 1	(5, 5, 16, 16)	(3, 3, 8, 32)
Batch norm		(3, 3, 8, 32)	(3, 3, 8, 32)
Leaky Relu	alpha=0.01	(3, 3, 8, 32)	(3, 3, 8, 32)
Conv 3D	Filters = 64, kernel= (1, 1, 7), stride= 1	(3, 3, 8, 32)	(1, 1, 2, 64)
Batch norm		(1, 1, 2, 64)	(1, 1, 2, 64)
Leaky Relu	alpha=0.01	(1, 1, 2, 64)	(1, 1, 2, 64)
Reshape		(1, 1, 2, 64)	(1, 128)
Glob pool		(1, 128)	(128)

The Convolutional block was used as joint spatial spectral feature extractor which utilizes both spatial and spectral features together in Euclidian structure form. There are different layers can be employed to achieve such task such as 3D convolutional layer which learn local short term volumetric data such as HSI cubes. The kernel, filters and stride hyper parameters in 3D convolutional layers were selected after many trial and considerations.

The Global Average Pooling layer was selected from Global Max Pooling and Global Min Pooling because it considers the contribution of overall spatial spectral regions during squeezing the dimension unlike other global pooling layer types. The Batchnormalization and Leaky Relu layers was used in order to stabilize the discriminator during training. These layers were added after trying to train GNGAN and the loss of the generator and discriminator in unsupervised mode become "*nan*". Leaky Relu activation was used due to two reason which are: to able Batchnormalization layer operates before activation function was applied to the output of 3D convolutional layers and compared to Relu activation, the GNGAN training was more stabilized.

4.4.2. Graph network block

The Graph network block takes labeled graphs as input which represents non-Euclidian structures of HSI and the graph is represented by node matrix in the shape of (49, 50) and adjacency matrix in the shape of (49, 49). The output of Graph network block is single column tensor with the shape (128). The Graph network block consists of spectral GCN layers with Relu as activation function and Global Attention Pooling layer. Laplacian matrix (LM) was computed from the adjacency matrix since the GCN layers takes it as input instead of adjacency matrix. The architecture of the Graph network block is summarized in the table below.

Table 4.3: layers of Graph network

Layers	Hyper parameter	Input	Output
GCN	Channels = 64, activation = Relu	Node = 49 x 50, LM = 49 x 49	Node = 49 x 64
GCN	Channels = 96, activation = Relu	Node = 49 x 64, LM = 49 x 49	Node = 49 x 96
GCN	Channels = 128, activation = Relu	Node = 49 x 96, LM = 49 x 49	Node = 49 x 128
GAP	Channels = 128	Node = 49 x 128	(128)

There are many type of GCN layers which operates using different methods. In this study, spectral GCN layer was used because it can operates both as node classifier and graph classifier. It can also operates on batch of graphs, single large graphs and disjoint graphs. The graph generated from HSI patches during preprocessing, are used in batch operations and can only be classified as a graph not in node level.

This GCN network were proposed by *(Kipf, T. N., & Welling, M. 2016)* is suitable for this study compared to networks like Graph Isomorphism Network (GIN) or Crystal Graph Convolutional network (CGCN) which were proposed by (*Xie T & Grossman J, 2018*) and (*Xu K et al, 2018*) respectively. Since the Graph network block doesn't participate in the generator training directly, the use of Leaky Relu as activation function is not necessary. Relu was used as activation function for all 3 GCN layer. The hyper parameters of GCN and GAP was selected after several trials and consideration which result best results during testing of GNGAN model.

The Global Attention pooling layer was proposed by *(Li Y. et al, 2015)* which is used to flatten the node matrix to a single size vector in order to produce the same size tensor to that of the output of

the convolutional block for concatenation purposes. Global Attention pooling layer was selected compared from other Global pooling layers which operate on the graphs such as Global average pooling layer, Global sum pooling layer and Global max pooling layer because it improves the ability to express the features during squeezing of the node dimension better than the other global pooling layers (*Yang Y et al, 2021*).

The following figure shows the architecture of Graph network block of GNGAN.

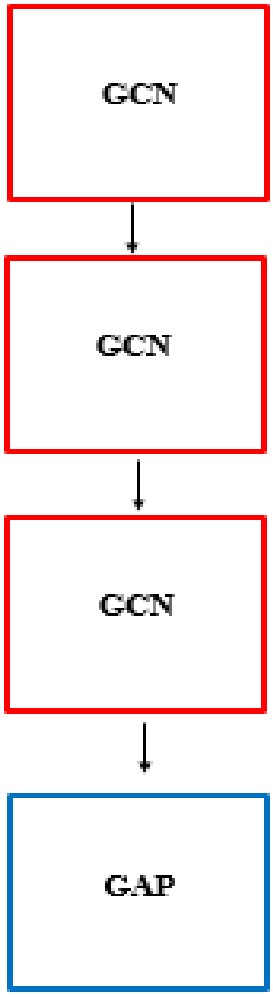


Figure 4.3: Architecture of Graph Network Block

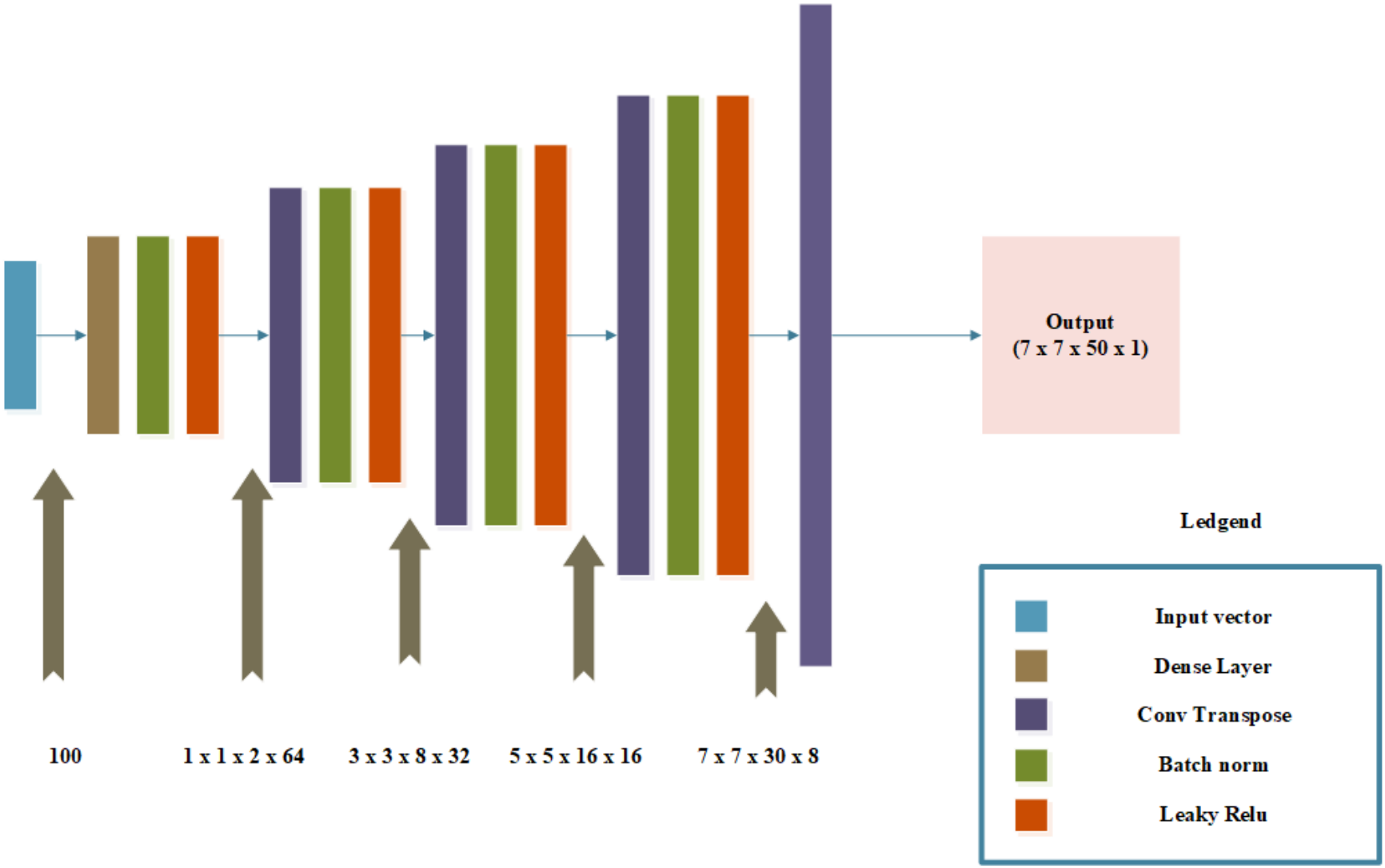


Figure 4.4: Architecture of Generator Network

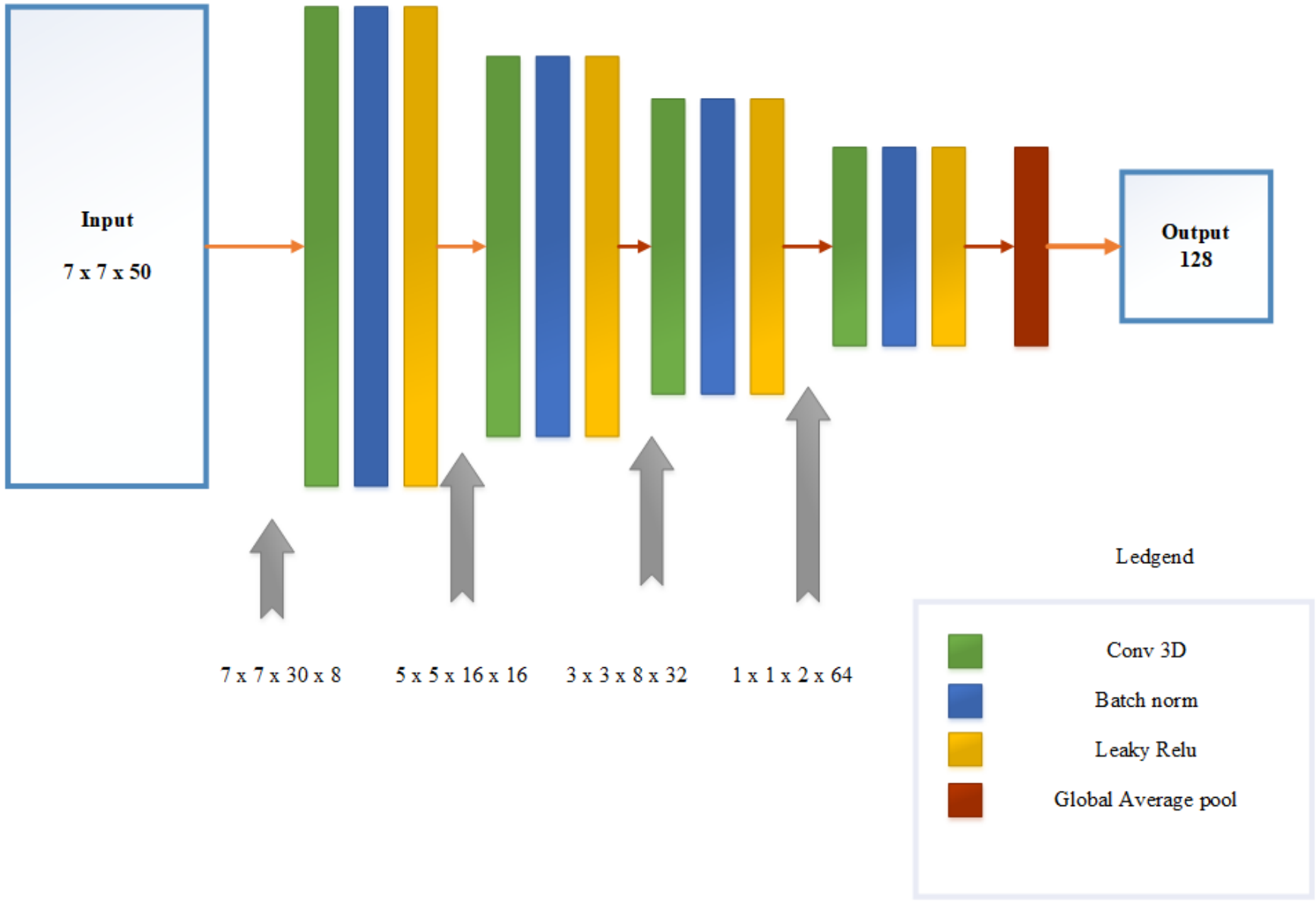


Figure 4.5: Architecture of Convolutional Block

4.5. Discriminator in Supervised mode

When the discriminator of GNGAN operates in supervised mode, both Convolutional block and Graph network block operate together and their output are concatenated to give an output with the shape of (256). Concatenation (CN) was selected as combination mechanism other than other methods such as Hadamard products (HP) and Element wise Addition (EA) after experiment was conducted using all three methods and CN performs better than the two combination methods. This experiment and its result is described in chapter six in detail.

Let **Gn** be an array contains elements of vector with the size of (128) which represent the output of Graph network block, and let **Cv** be an array contains elements of vector with size of (128) which represents the output of Convolutional block. The following equation represents the result array after combination by the three methods.

$$\boldsymbol{F_{CN} = F\{Gn \cup Cv\}} \quad \textbf{..... Eq. (4.2)}$$

$$\boldsymbol{F_{HP}(i) = Gn(i) * Cv(i)} \quad \textbf{..... Eq. (4.3)}$$

$$\boldsymbol{F_{EA}(i) = Gn(i) + Cv(i)} \quad \textbf{..... Eq. (4.4)}$$

The output of CN was applied to fully connected layers which contains two fully connected or dense layers, one Batchnormalization layer and one leaky relu layer. The Batchnormalization and leaky relu layers are applied to the first fully connected layer which have 64 units. The last fully connected layer has Softmax as activation function with units equal to 16 or 9 when Salina Valley dataset and Pavia University dataset are used respectively.

Softmax activation function was used in order to predict multinomial probability in size of 16 or 9 based on which dataset was used. The units 16 or 9 in last fully connected layer corresponds to the number of classes in Salina Valley and Pavia University dataset. The Batchnormalization and leaky relu layers were used in order to stabilize the discriminator.

The loss function used in the discriminator during supervised mode, is categorical cross entropy since the operation is multi class classification. The loss function was selected because it produce

a probable match for each class which makes it best for multiclass classification. The equation of categorical cross entropy is given below.

$$L_{sup} = -\sum \hat{y} \log(D_{sub}(X, G, L)) \quad \text{..... Eq. (4.5)}$$

Where $\mathbf{D_{sub}}$ is the discriminator in supervised mode and **X**, **G** and **L** are HSI patch, node matrix and Laplacian Matrix respectively**. Y prime** is the actual ground truth.

The following table summarizes the hyper parameters and the shapes of input and output of the Discriminator in supervised mode excluding Graph network block and Convolutional block.

Table 4.4: Layers of discriminator in supervised mode

Layers	Hyper parameter	Input	Output
Concatenation		Input 1 = 128 Input 2 = 128	256
Dense	Unit = 64	256	64
Batch norm		64	64
Leaky Relu	Alpha = 0.01	64	64
Dense	Unit = 16 for Salina Valley, activation = Softmax Unit = 9 for Pavia University	64	16 or 9

4.6. Discriminator in Unsupervised mode

When the discriminator of GNGAN operates in unsupervised mode, Convolutional block operates .but not Graph network block. The output of the convolutional block was applied to the two fully connected layers and Batchnormalization layer with leaky relu as activation layer. The Batchnormalization and leaky relu layers are used for stability of the discriminator and applied after first fully connected layer.

The last fully connected layer has a unit equals to 1 since the discriminator distinguish the patches as real or fake. The activation function is sigmoid in order to produce the output either as 0 or 1. The loss function employed for the discriminator in unsupervised mode is binary cross entropy

because of the two possible outputs. The equation of the loss for both discriminator and generator in unsupervised mode is given in the equation below.

$$L(G,D) = E[log(D(x)] + E[log(1 - D(G(z)))] \text{ ..... Eq. (4.6)}$$

Where **G** is Generator, **D** is Discriminator and **Z** is noise vector. **E** stands for entropy. The following table summarize the architecture of the discriminator excluding the convolutional block.

Table 4.5: Layers of discriminator in unsupervised mode

Layer	Hyper parameter	Input	Output
Dense	Units = 64	128	64
Batch norm		64	64
Leaky Relu	Alpha = 0.01	64	64
Dense	Units = 1, activation = sigmoid	64	1

4.7. Training Pseudocode

The following is the training pseudocode of the proposed model GNGAN. In this algorithm, $\mathbf{L_{duns}}$, $\mathbf{L_{dsup}}$ and $\mathbf{L_{gen}}$ represent loss of unsupervised discriminator, supervised discriminator and Generator respectively. Where **L**, $\mathbf{D_{uns}}$, $\mathbf{D_{sub}}$ and **G** represents Loss function, unsupervised Discriminator, Supervised Discriminator and Generator models. The word **"*yes*"** represent the discriminator distinguish a sample as real and a word **"*no*"** represent the discriminator distinguish a sample as fake. The letter **"*k*"** represent the number of classes of labeled datasets. The training sample for Salina Valley and Pavia University are **1600** and **900.** The batch size for Salina valley and Pavia University are **32** and **18**. The number of batches in one epoch for both datasets is **50** which can be calculated by dividing training samples by batch size.

1 *Apply PCA*
2 *Patch the image in to smaller units*
3 *Generate graph structures from patch*
4 *For 200 epochs*
5 *For 50 no of batch: Train D_{uns}, with patches*
6 *L_{duns} = 0.5*(L{D_{uns} (real, yes)} + L{ D_{uns} (generated, no)})*
7 *For 50 no of batch: Train G*
8 *L_{gen} = L{D_{uns} (G(z), yes)}*
9 *For 50 no of batch: Train D_{sub}*
10 ***Take patches in Conv3D block***
11 ***Take graphs in GCN block***
12 ***Combine the output of Conv3D and GCN block***
13 ***Feed to Dense layers with Softmax activation***
14 ***L_{dsup} = L{ D_{sub} (output, k)}***

The section where the written in bold is the added specific contribution of GNGAN model to semi-supervised GAN.

Chapter Five

Implementation Details of Proposed Model

5.1. Chapter overview

In this chapter, the implementation of Graph Network based Generative Adversarial Network model are discussed. The working environments, Experimental arrangements and implementation of different parts of the proposed model are discussed in this chapter.

5.2. Working Environment

During training and testing the proposed model with the datasets, the following hardware specifications and software packages were used.

Hardware:

1. **Laptop**

- Processor: Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz 2.20 GHz
- RAM: 8.00 GB
- Operating system: Windows 10 pro 64 bit

2. **Google collaborator**

- GPU: Tesla T4
- RAM: 12.00 GB

Software:

- Jupyter notebook
- Tensorflow framework and Keras library
- Spektral library for GCN and GAP layers
- Spectral library for visualization of Hyperspectral image cubes and ground truth
- Common python libraries like numpy, scipy and matplotlib

5.3. Preprocessing methods implementation

The two datasets were subjected to the preprocessing method before fed to the proposed model for training and testing purposes. There are three preprocessing methods which are mentioned in chapter three in details. These methods were PCA, Patching and graph Construction but we will discuss the graph construction and patching methods in this section.

Since there is one Hypercube in each dataset, the cube should be divided in to multiple patches for the processing. The patching processing were done using the following python code.

```
def padWithZeros(X, margin=2):
    newX = np.zeros((X.shape[0] + 2 * margin, X.shape[1] + 2* margin, X.shape[2]))
    x_offset = margin
    y_offset = margin
    newX[x_offset:X.shape[0] + x_offset, y_offset:X.shape[1] + y_offset, :] = X
    return newX
def createImageCubes(X, y, windowSize=5, removeZeroLabels = True):
    margin = int((windowSize - 1) / 2)
    zeroPaddedX = padWithZeros(X, margin=margin)
    # split patches
    patchesData = np.zeros((X.shape[0] * X.shape[1], windowSize, windowSize, X.shape[2]))
    patchesLabels = np.zeros((X.shape[0] * X.shape[1]))
    patchIndex = 0
    for r in range(margin, zeroPaddedX.shape[0] - margin):
        for c in range(margin, zeroPaddedX.shape[1] - margin):
            patch = zeroPaddedX[r - margin:r + margin + 1, c - margin:c + margin + 1]
            patchesData[patchIndex, :, :, :] = patch
            patchesLabels[patchIndex] = y[r-margin, c-margin]
            patchIndex = patchIndex + 1
    if removeZeroLabels:
        patchesData = patchesData[patchesLabels>0,:,:,:]
        patchesLabels = patchesLabels[patchesLabels>0]
        patchesLabels -= 1
    return patchesData, patchesLabels
```

Code snippet 5.1: Patching

These two functions are for padding the hypercube with the size of margin on all sides and patching it to multiple parts afterwards. The window size was 7 and the value of the margin was 3. After the hypercube was patched, the second function removes patches with zero labels which are belongs to the background part of the hypercube. The remaining cubes are all part of the foreground pixels. Each pixel has a corresponding patch with 48 other pixel neighbors that is why the padding had used before the patching was done for the pixels located at the boundary.

The second method was the Graph construction methods with two steps which are: node generation and adjacency matrix calculation. The node generation is a step that generate a graph nodes out of a patch using the pixels presented in it. The node generation step is shown in the following code.

```
def node(X):
    Xprime = np.reshape(X, (X.shape[0], X.shape[1]*X.shape[2], X.shape[3]))
    return Xprime
```

Code snippet 5.2: node generation

In the above code, x is the patch hyper cubes with the size of (number of samples, 7, 7, 50) array and the function on the code make the size to (number of samples, 49, 50) array where 49 = 7x7. This indicates there are 49 nodes or pixels in a graph with 50 node features or spectral signature for each nodes.

The second step is adjacency matrix calculation which has three steps which are calculating the mean of the spectral signature of every node, calculate the Manhattan distance between the mean of the central node and the rest of the nodes and find 15 nearest neighbors, and the final step is connect central node with 15 selected neighbors connected with each other by setting their corresponding adjacency cell value to 1 and 0 for the other connections. The following 3 code snippets shows the three steps respectively.

```
def Graphsum(G):
    Gsum = np.zeros((G.shape[0], G.shape[1], 1))
    for i in range(G.shape[0]):
        for j in range(G.shape[1]):
            summ = 0
            for k in range(G.shape[2]):
                summ = summ + G[i][j][k]
            Gsum[i][j][0] = summ/50
    return Gsum
```

Code snippet 5.3: computing means of spectral signature of every node

In the above code, the input G is the node matrix with the size of (number of graphs, 49, 50) and the output size is (number of graphs, 49, 1) where the last dimension indicates the 50 node features are added together and divided by 50 to give the mean value.

```
def neighval(Gsum):
    neilist = []
    for i in range(Gsum.shape[0]):
        dictlist = {}
        elelist = []
        for j in range(Gsum.shape[1]):
            if j == 24: # 24 is the index of the central node, from 49 nodes
                        # 0 to 48 index, 24 is the middle one
                continue
            else:
                # manhatten distance calculation
                dictlist[j] = math.fabs(Gsum[i][j][0] - Gsum[i][24][0])
        # sort by their manhatten distance in assending order
        dictlist = dict(sorted(dictlist.items(), key=lambda item: item[1]))
        new_list = list(dictlist.items())
        # select 15 nearest neighbors
        sorted_list = new_list[0:14]

        neilist.append(sorted_list)

    return neilist
```

Code snippet 5.4: Manhattan distance calculation of nearest neighbors

The function in the code takes an array of nodes in a graph with mean value of node features and select the middle or central node with index 24 to calculate Manhattan distance with other 48 nodes to find nearest neighbors. 15 nearest neighbors were selected and arranged in a list.

```
def neilist(Gsum):
    nl = neighval(Gsum)
    nlist = []
    for i in range(len(nl)):
        mlist = [24]
        for j in range(15):
            mlist.append(nl[i][j][0])
        nlist.append(mlist)
    return nlist

def Adjmat(Gsum):
    N = neilist(Gsum)
    A = np.zeros((Gsum.shape[0], Gsum.shape[1], Gsum.shape[1]))
    for i in range(len(N)):
        for j in range(16):
            for k in range(16):
                A[i][N[i][j]][N[i][k]] = 1
    return A
```

Code snippet 5.5: Setting values to adjacency matrix

5.4. Generator model implementation

The generator model is used to train the shared convolutional block in unsupervised mode to increase its performance when it is used in supervised discriminator for the classification task. The architecture details of the generator were covered in chapter four and it was implemented using Keras layers which are: ConvTranspose3D, Leaky Relu, Batchnormalization and Dense layers. The following code snippet shows the implementation details.

```
def Generator(latent_dim = 100):
   inputlayer = Input(latent_dim)
   denselayer1 = Dense(units=128)(inputlayer)
   denselayer1 = BatchNormalization()(denselayer1)
   denselayer1 = LeakyReLU(alpha=0.01)(denselayer1)
  # 1x1x2x64
   image2d = Reshape((1, 1, 2, 64))(denselayer1)
  # 3x3x8x32
   convtrans1 = Conv3DTranspose(filters=32, kernel_size=(3,3,7), strides=(1,1,1), padding='valid')(image2d)
   convtrans1 = BatchNormalization()(convtrans1)
   convtrans1 = LeakyReLU(alpha=0.01)(convtrans1)
  # 5x5x16x16
   convtrans2 = Conv3DTranspose(filters=16, kernel_size=(3,3,9), strides=(1,1,1), padding='valid')(convtrans1)
   convtrans2 = BatchNormalization()(convtrans2)
   convtrans2 = LeakyReLU(alpha=0.01)(convtrans2)
  # 7x7x30x8
   convtrans3 = Conv3DTranspose(filters=8, kernel_size=(3,3,15), strides=(1,1,1), padding='valid')(convtrans2)
   convtrans3 = BatchNormalization()(convtrans3)
   convtrans3 = LeakyReLU(alpha=0.01)(convtrans3)
  # 7x7x50x1
   convtrans4 = Conv3DTranspose(filters=1, kernel_size=(1,1,21), strides=(1,1,1), padding='valid', activation='sigmoid')(convtrans3)

   model = Model(inputs=inputlayer, outputs=convtrans4)

   return model
```

Code snippet 5.6: Generator implementation details

The input of the generator is a noise vector with the size of (100) and each element in the vector has a value between 0 and 1.

5.5. Discriminator model implementation

5.5.1. Convolutional block implementation

The convolutional block is a shared model that made up of 3D convolutional layers and global average pooling layer. It is used to utilize spatial spectral features in regular grid structure of HSI cube. The following code shows the implementation details of convolutional block

```
def discriminator(TrX):
  inputshape = (TrX.shape[1], TrX.shape[2], TrX.shape[3], 1)
  inputlayer = Input(inputshape)
  # 7x7x30x8
  conv3D1 = Conv3D(filters=8, kernel_size=(1, 1, 21))(inputlayer)
  conv3D1 = BatchNormalization()(conv3D1)
  conv3D1 = LeakyReLU(alpha=0.01)(conv3D1)
  # 5x5x16x16
  conv3D2 = Conv3D(filters=16, kernel_size=(3, 3, 15))(conv3D1)
  conv3D2 = BatchNormalization()(conv3D2)
  conv3D2 = LeakyReLU(alpha=0.01)(conv3D2)
  # 3x3x8x32
  conv3D3 = Conv3D(filters=32, kernel_size=(3, 3, 9))(conv3D2)
  conv3D3 = BatchNormalization()(conv3D3)
  conv3D3 = LeakyReLU(alpha=0.01)(conv3D3)
  # 1x1x2x64
  conv3D4 = Conv3D(filters=64, kernel_size=(3, 3, 7))(conv3D3)
  conv3D4 = BatchNormalization()(conv3D4)
  conv3D4 = LeakyReLU(alpha=0.01)(conv3D4)
  # 1x1x128
  outputlayer  = Reshape((1,1,128))(conv3D4)
  outputlayer = GlobalAveragePooling2D()(outputlayer)
  model = Model(inputs = inputlayer, outputs = outputlayer)
  return model
```

Code snippet 5.7: Convolutional block implementation details

The input of the convolutional block is generated, unlabeled real and labeled real Hypercube in the shape of (7, 7, 50) for each patch.

5.5.2. Graph block implementation

The graph block is used to utilize irregular geometric structure in a graph form and spectral long term dependencies. The block consists of GCN layers and Global Attention Pooling layer. It receives two inputs which are node matrix and laplacian matrix and produce a tensor output with the shape of (128). The implementation details are described in the following code.

```
def Graphnetwork(TrG, TrL):
    gin = Input(shape = (TrG.shape[1], TrG.shape[2])) # (49, 50)
    ain = Input(shape = (TrL.shape[1], TrL.shape[2])) # (49, 49)
    #(49, 64)
    gcn1 = GCNConv(64, activation = 'relu')([gin, ain])
    #(49, 96)
    gcn2 = GCNConv(96, activation = 'relu')([gcn1, ain])
    #(49, 128)
    gcn3 = GCNConv(128, activation = 'relu')([gcn2, ain])
    #(128)
    outputlayer = GlobalAttentionPool(128)(gcn3)
    model = Model(inputs = [gin, ain], outputs = outputlayer)
    return model
```

Code snippet 5.8: Graph network block

One of the inputs is a laplacian matrix and the calculation of the laplacian matrix from the adjacency matrix is given in the following code using Spektral library.

```
def normlaplace(A):
    L = np.zeros((A.shape))
    for i in range(A.shape[0]):
        L[i] = gcn_filter(A[i])
    return L
```

Code snippet 5.9: computing laplacian matrix out of adjacency matrix

5.5.3. Unsupervised discriminator implementation

The unsupervised discriminator means discriminator in unsupervised mode which is composed of convolutional block and other small layers like dense layers and batch normalization layers. The code implementation is given below.

```
def discriminator_unsup(disc):
  model = tf.keras.models.Sequential()
  model.add(disc)
  model.add(Flatten())
  model.add(Dense(240, activation = 'relu'))
  model.add(Dense(1, activation = 'sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.0002, beta_1=0.5))
  return model
```

Code snippet 5.10: Unsupervised discriminator implementation

The input of the function in "*discriminator_unsup*" is a convolutional block model which was connected sequentially by using "*Keras.models.Sequentials ()"* function. During training, the generator and unsupervised discriminator should be combined to form a GAN in order train the generator. In this combined model, the unsupervised discriminator is not trained but only the generator does. The following code described the combined model.

```
def define_gan(gen_model, disc_unsup):

  disc_unsup.trainable = False # make unsup. discriminator not trainable
  gan_output = disc_unsup(gen_model.output) #Gen. output is the input to disc.
  model = Model(gen_model.input, gan_output)
  model.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.0002, beta_1=0.5))
  return model
```

Code snippet 5.11: Combined GAN model for Generator training

5.5.4. Supervised discriminator implementation

The supervised discriminator means discriminator in supervised mode which is used as a multi class classifier of Hyperspectral image using both Euclidian structure and non-Euclidian structure of Hyperspectral image features. The learning rate used by this model is 0.002. The following code describes the implementation details of supervised discriminator.

```
def discriminator_sup(disc, gcn, Cno):
  in1 = Input(shape = (disc.get_layer(index = 14).output.shape[1]))
  in2 = Input(shape = (gcn.get_layer(index = 5).output.shape[1]))
  di = concatenate([in1, in2],name="concatenated_layer")
  flatlayer = Flatten()(di)
  dense_layer1 = Dense(units=64)(flatlayer)
  dense_layer1 = BatchNormalization()(dense_layer1)
  dense_layer1 = LeakyReLU(alpha=0.01)(dense_layer1)
  dense_layer2 = Dense(units=Cno, activation = 'softmax')(dense_layer1)


  model = Model(inputs = [in1, in2], outputs = dense_layer2)
  model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.002, beta_1=0.5), metrics= ['accuracy'])
  return model
```

Code snippet 5.12: Supervised discriminator implementation

The variable **Cno** is the number of class which is used as unit value for the last dense layer. These makes the model is compatible for datasets with different number of classes. The code takes convolutional block model and graph network block model as inputs and concatenate them together.

5.6. Experiment arrangements

In this research, four different experiments were conducted using both datasets. The first experiment was conducted for comparing the proposed model GNGAN with SSGAN which only utilizes HSI in Euclidian form and GCN which only utilizes HSI in non-Euclidian form. The SSGAN has the same structure with GNGAN except it lacks Graph network block and GCN model has the same structure with Graph network block in addition fully connected layers and Softmax layer.

The second experiment was to show the impact of number of epochs on the proposed model during training by comparing the test results with different epoch values. The third experiment was to show the impact of number of training samples on the proposed model since it is a semi-supervised classification model. The final experiment was to compare different combination mechanisms for combine the outputs of convolutional block and Graph network block in GNGAN. These mechanisms were concatenation, element wise addition and Hadamard product.

Table 5.1: Summary of the Experiment arrangement

No	Experiment	Purpose
1	Comparison among GNGAN, SSGAN and GCN models	Impact of Convolutional block and Graph network block
2	Comparison among GNGAN trained with different epochs	Impact of epoch
3	Comparison among GNGAN trained with different amount of data	Impact of no of training data
4	Comparison among GNGAN with Concatenation, Hadamard product and element wise addition mechanism for combining Convolutional block and graph network block	To find effective method for combination mechanism **RQ3**

In each experiments, there are two tests for Salina Valley and Pavia University dataset each. Before the experiments, Overfitting prevention method were considered due to the variation in numbers elements among different classes in each dataset. Therefore, data balancing were made before the experimentation. The following table shows the details of the data balancing.

Table 5.2: Data balancing for Pavia University dataset

No	Class name	No of elements before balancing	No of elements after balancing	Train Test ratio	
				Train	Test
1	Asphalt	6631	947	100	847
2	Meadows	18649	947	100	847
3	Gravel	2099	947	100	847
4	Trees	3064	947	100	847
5	Painted metal sheets	1354	947	100	847
6	Bare soil	5029	947	100	847
7	Bitumen	1330	947	100	847
8	Self-blocking bricks	3682	947	100	847
9	Shadows	**947**	947	100	847
Total		42776	8523	900	7623

The number of elements in each class in Pavia University dataset after balancing is 947 because the class with the smallest number of elements is "*Shadow"* class with the number elements of 947. In Salina Valley, the smallest element is 916 and the class is "*Lettuce Romaine 7wk*". In order to show the proposed model works fine with small number labeled elements, the train to test ratio is approximately 10 to 90. Different ratios were considered in third experiments.

Table 5.3: Data balancing for Salina Valley dataset

No	Class name	No of elements before balancing	No of elements after balancing	Train Test ratio	
				Train	Test
1	Broccoli green weeds 1	2009	916	100	816
2	Broccoli green weeds 2	3726	916	100	816
3	Fallow	1976	916	100	816
4	Fallow rough plow	1394	916	100	816
5	Fallow smooth	2678	916	100	816

6	Stubble	3959	916	100	816
7	Celery	3579	916	100	816
8	Grapes untrained	11271	916	100	816
9	Soil vineyard develop	6203	916	100	816
10	Corn senesced green weeds	3278	916	100	816
11	Lettuce romaine 4wk	1068	916	100	816
12	Lettuce romaine 5wk	1924	916	100	816
13	Lettuce romaine 6wk	**916**	916	100	816
14	Lettuce romaine 7wk	1070	916	100	816
15	Vineyard untrained	7268	916	100	816
16	Vineyard vertical trellis	1807	916	100	816
Total		54129	14656	1600	13056

Chapter Six

Result and Discussion

6.1. Chapter overview

In this chapter, the result of the experiments are reported and discussed. The proposed method outcome is compared with two models SSGAN and GCN using three evaluation matrices which are: overall accuracy, Average and Each accuracy and kappa coefficient. In this section, the result of the experiments and the analysis of the result are discussed in detail by attempting to answer the research questions.

6.2. Training the proposed model

The proposed model and the two other models were trained under the same environments like epoch, batch size, and learning rate. All the model were used Adam optimizer with the parameter of learning rate and first exponential decay for first momentum estimate or beta one. The models GNGAN and SSGAN are semi-supervised GAN based model and they have two different learning rate for the discriminator in supervised mode and unsupervised mode. In this experiment, the value of the discriminator and generator learning rate was selected after many trial and error during training of GNGAN. The following table summarizes the parameters used for training all the models.

Table 6.1: Parameters for Model training

Models	GNGAN	SSGAN	GCN
No of epoch	200	200	200
Batch size	32 for SV & 18 for PU	32 for SV & 18 for PU	32 for SV & 18 for PU
Optimizer	Adam	Adam	Adam
Learning rate	SD for 0.002 and G & UD 0.0002	SD for 0.002 and G & UD 0.0002	0.002

The loss for the supervised discriminator of the proposed model GNGAN for both Salina Valley and Pavia University are shown in the figures below.

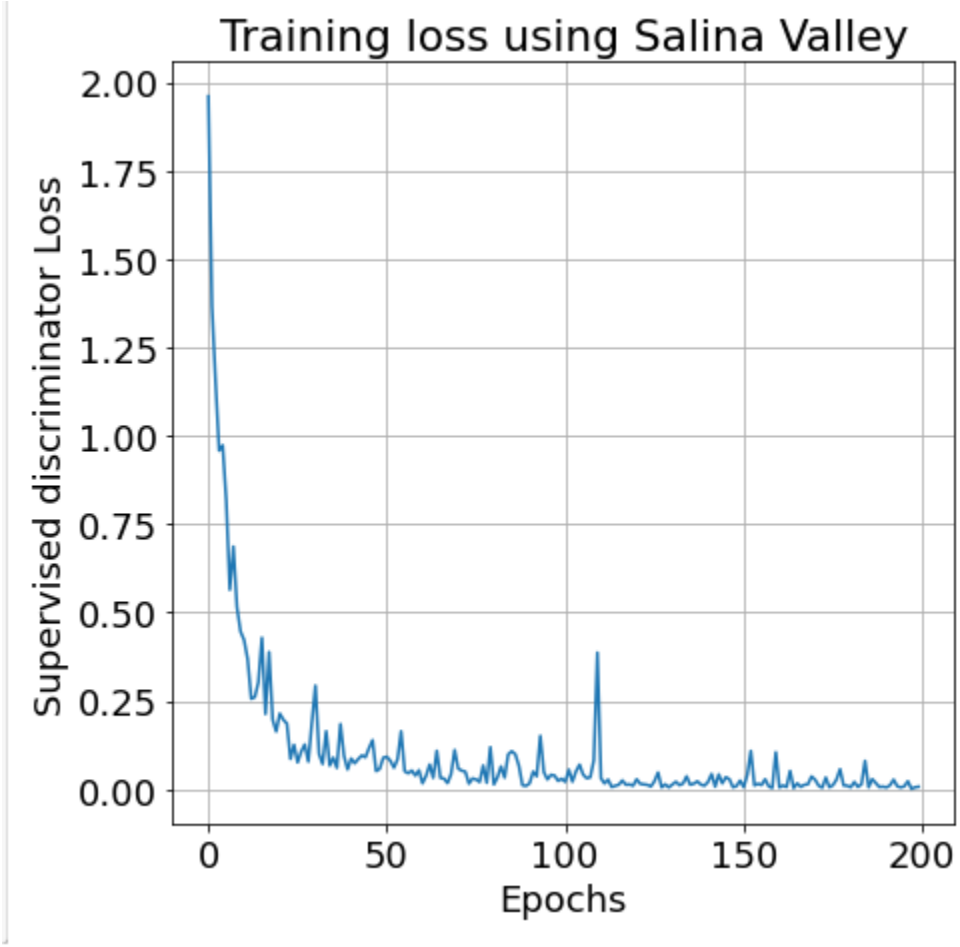


Figure 6.1: Supervised discriminator of GNGAN's loss during training using Salina valley dataset

In the figures, the loss decay is not smooth due to the effect of generator and unsupervised discriminator during training. The main emphasis of the training procedure is to minimize the loss of the supervised discriminator but not the generator and discriminator. Categorical cross entropy loss function were used for supervised discriminator whereas Binary cross entropy loss function was used for the generator and unsupervised discriminator.

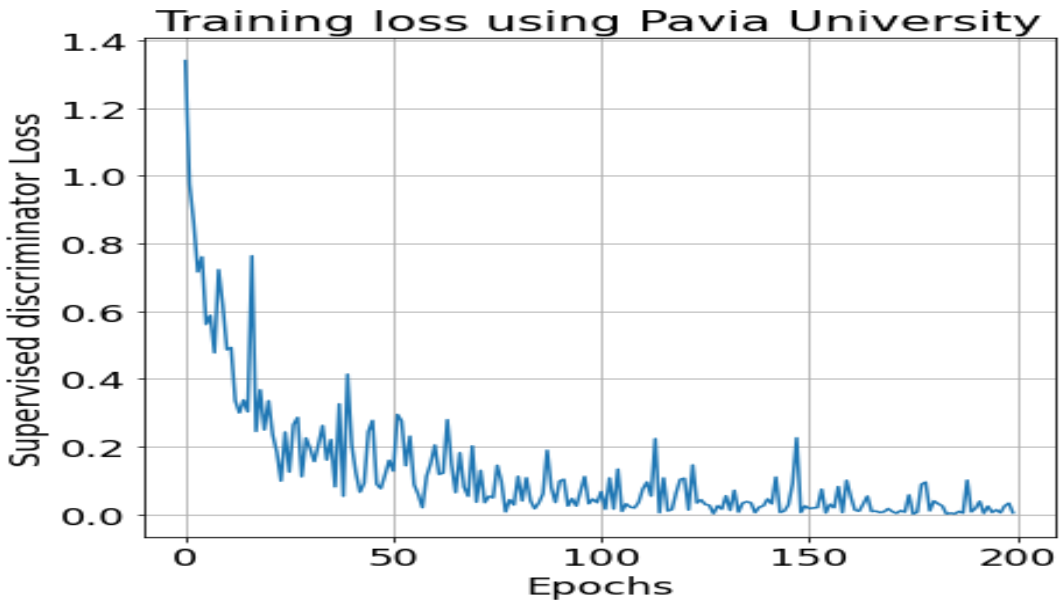


Figure 6.2: Supervised discriminator of GNGAN's loss during training using Pavia University dataset

The loss of the generator and unsupervised discriminator of GNGAN is shown below for both Salina Valley and Pavia University datasets.

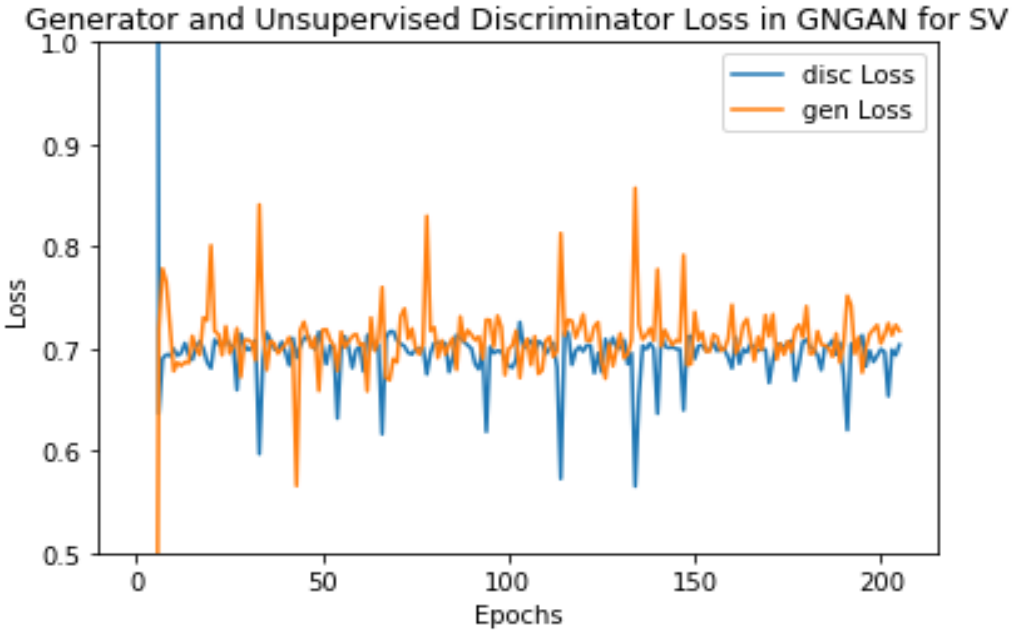


Figure 6.3: Generator and Unsupervised discriminator of GNGAN's loss for SV

Figure 6.4: Generator and Unsupervised discriminator of GNGAN's loss for PU

6.3. Results for GNGAN model

The results of evaluating the GNGAN model and the other models using testing data are described in the form of Confusion Matrices, using evaluation matrices and comparing 2D chart of ground truth with predicted labels.

6.3.1. Confusion Matrices for GNGAN

Confusion matrices of GNGAN are drawn in this section for both datasets whereas the confusion matrices of the other two models are drawn in Appendix A section.

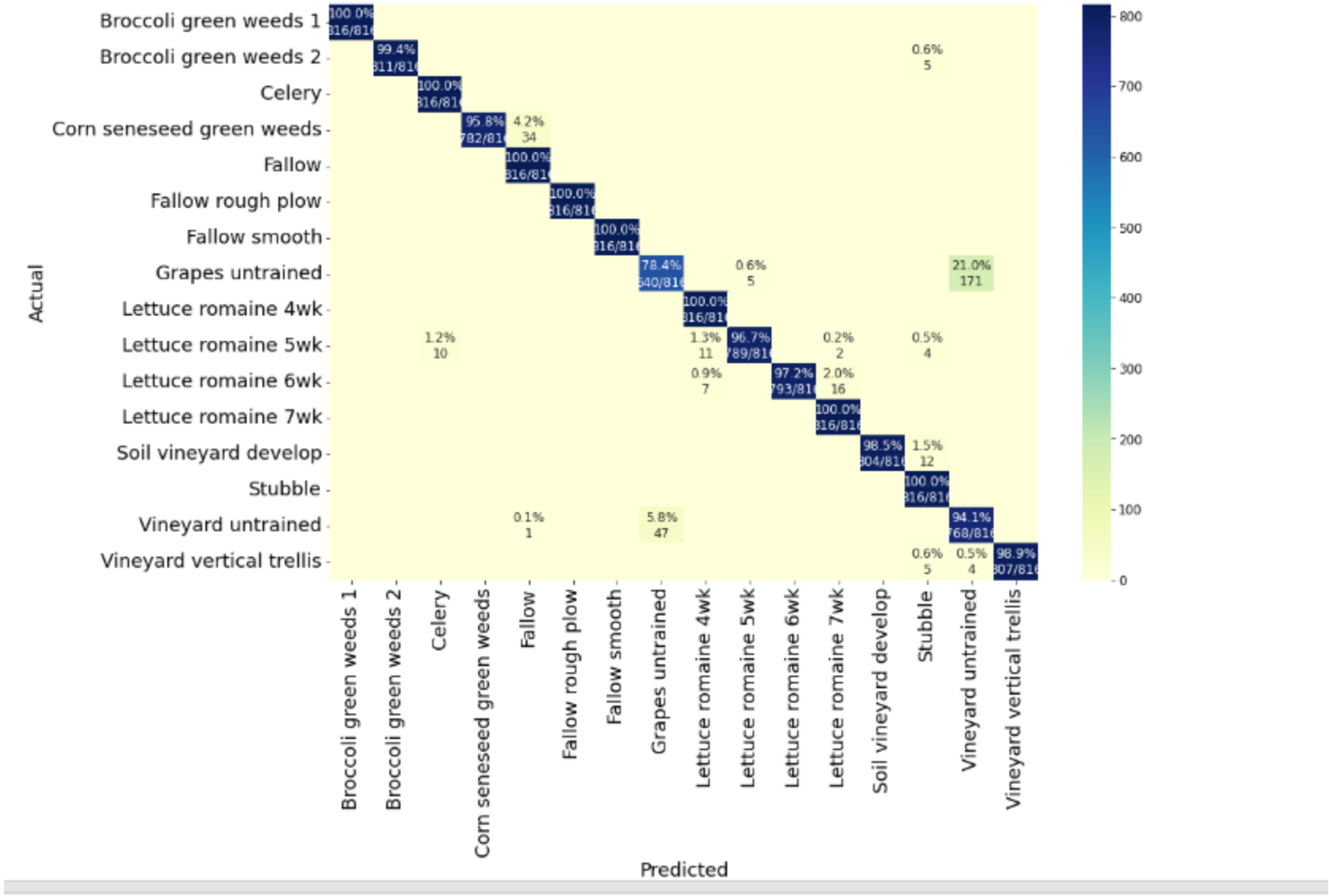


Figure 6.5: Confusion Matrix of GNGAN for Salina Valley dataset

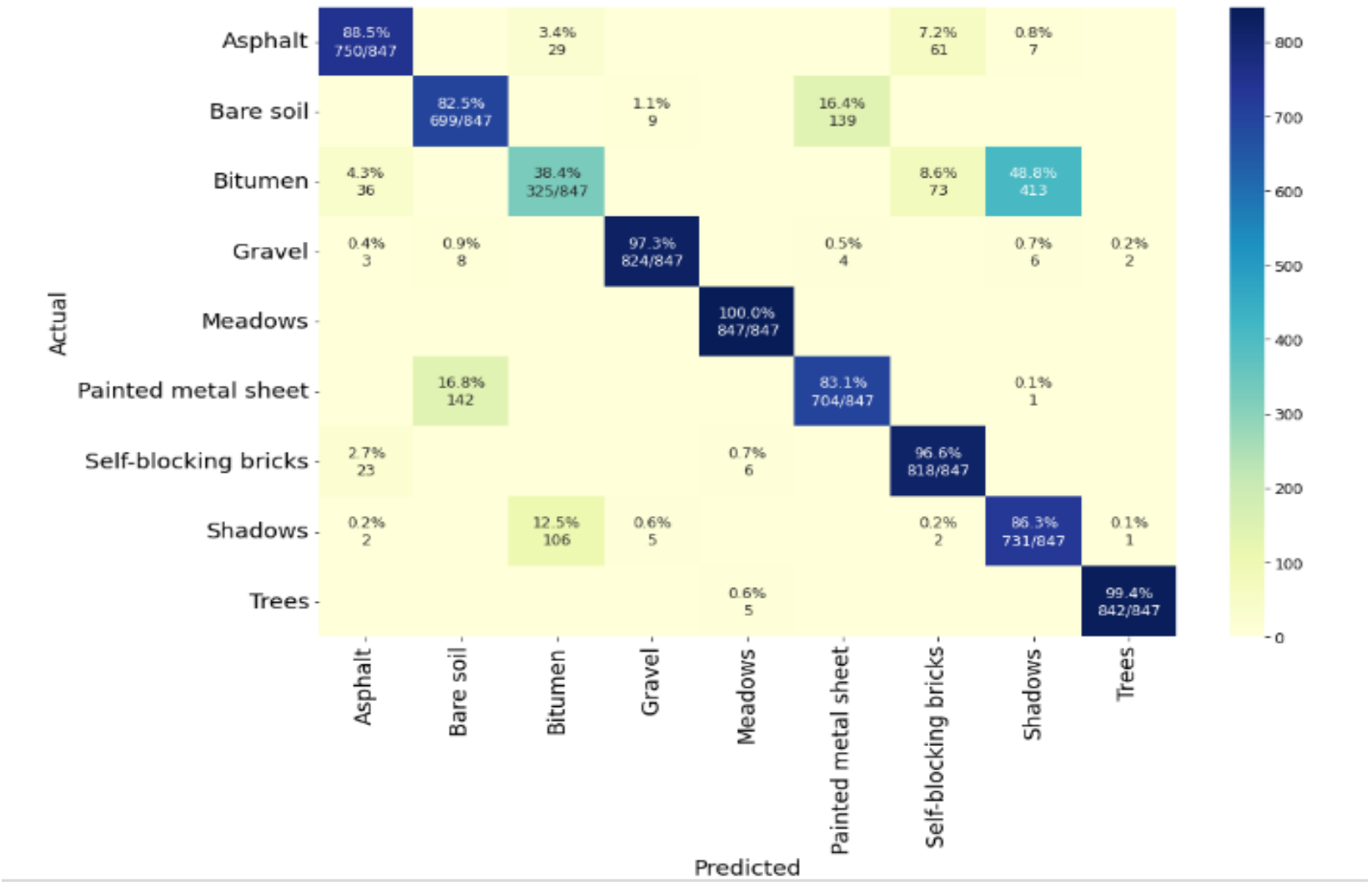


Figure 6.6: Confusion Matrix of GNGAN using Pavia University dataset

6.3.2. Evaluation matrices for GNGAN

The acquisition of Hyperspectral image and preparation process for producing Hyperspectral image dataset is time consuming, very expensive and require Image spectroscope camera. The most common Hyperspectral datasets such as Pavia University, Salina Valley, Indian Pines, Washington, Huston and others are captured either from space or aerial vehicles. This makes the datasets categorized under remote sensing datasets and remote sensing datasets are evaluated mostly with Overall accuracy, each accuracy for every class, average accuracy and Kappa Coefficient.

The above mentioned evaluation matrices has been used in this test. The following tables show the evaluation of the proposed model using evaluation matrices for each datasets.

Table 6.2: Evaluation result of GNGAN using Pavia University

No	Classes	Training sample	Predicted samples per actual samples	Accuracy (%)
1	Asphalt	100	750/ 847	88.5
2	Bare soil	100	699/ 847	82.5
3	Bitumen	100	325/ 847	38.4
4	Gravel	100	824/ 847	97.3
5	Meadows	100	847/ 847	100
6	Painted metal sheet	100	704/ 847	83.1
7	Self-blocking bricks	100	818/ 847	99.6
8	Shadows	100	731/ 847	86.3
9	Trees	100	842/ 847	99.4
Overall Accuracy (%)				**85.79**
Average Accuracy (%)				**85.79**
Kappa Coefficient (%)				**84.01**

Table 6.3: Evaluation result of GNGAN using Salina Valley

No	Classes	Training sample	Predicted samples per actual samples	Accuracy (%)
1	Broccoli green weeds 1	100	816/816	100
2	Broccoli green weeds 2	100	811/816	99.4
3	Celery	100	816/816	100
4	Corn senesced green weeds	100	782/816	95.8
5	Fallow	100	816/816	100
6	Fallow rough plow	100	816/816	100
7	Fallow smooth	100	816/816	100
8	Grapes untrained	100	640/816	78.4
9	Lettuce romaine 4wk	100	816/816	100
10	Lettuce romaine 5wk	100	789/816	96.7
11	Lettuce romaine 6wk	100	793/816	97.2
12	Lettuce romaine 7wk	100	816/816	100
13	Soil vineyard develop	100	804/816	98.5
14	Stubble	100	816/816	100
15	Vineyard untrained	100	768/816	94.1
16	Vineyard vertical trellis	100	807/816	98.9
Overall Accuracy (%)				**97.41**
Average Accuracy (%)				**97.41**
Kappa Coefficient (%)				**97.27**

6.3.3. Comparison of models

Other model were compared to the proposed model via the three evaluation matrices using both datasets. The following table shows the comparison among the models.

Table 6.4: Comparison among GNGAN, SSGAN and GCN via Evaluation Matrices using Salina Valley dataset

No	Classes	Train samples	Test sample	GNGAN	SSGAN	GCN
1	Broccoli green weeds 1	100	816	100	100	87.5
2	Broccoli green weeds 2	100	816	99.4	99.8	99.9
3	Celery	100	816	100	99.9	100
4	Corn senesced green weeds	100	816	95.8	98.9	98.5
5	Fallow	100	816	100	98.8	81.2
6	Fallow rough plow	100	816	100	100	99.0
7	Fallow smooth	100	816	100	100	97.4
8	Grapes untrained	100	816	78.4	94.9	49.6
9	Lettuce romaine 4wk	100	816	100	100	98.7
10	Lettuce romaine 5wk	100	816	96.7	98.0	97.2
11	Lettuce romaine 6wk	100	816	97.2	99.6	98.2
12	Lettuce romaine 7wk	100	816	100	100	100
13	Soil vineyard develop	100	816	98.5	95.6	100
14	Stubble	100	816	100	100	98.3
15	Vineyard untrained	100	816	94.1	65.6	98.5
16	Vineyard vertical trellis	100	816	98.9	99.4	97.5

Overall Accuracy (%)	**97.41**	**96.89**	**93.84**
Average Accuracy (%)	**97.41**	**96.89**	**93.84**
Kappa Coefficient	**97.27**	**96.68**	**93.43**

Table 6.5: Comparison among GNGAN, SSGAN and GCN via Evaluation Matrices using Pavia University dataset

No	Classes	Train sample	Test sample	GNGAN	SSGAN	GCN
1	Asphalt	100	847	88.5	77.6	90.0
2	Bare soil	100	847	82.5	70.4	52.9
3	Bitumen	100	847	38.4	37.9	64.8
4	Gravel	100	847	97.3	97.3	96.8
5	Meadows	100	847	100	100	100
6	Painted metal sheet	100	847	83.1	91.3	93.2
7	Self-blocking bricks	100	847	99.6	77.3	78.4
8	Shadows	100	847	86.3	91.5	48.5
9	Trees	100	847	99.4	99.3	99.6
Overall Accuracy (%)				**85.79**	**82.50**	**80.49**
Average Accuracy (%)				**85.79**	**82.50**	**80.49**
Kappa Coefficient (%)				**84.01**	**80.31**	**78.05**

The results of the model using evaluation matrices for both datasets can be showed using bar chart for better comparison.

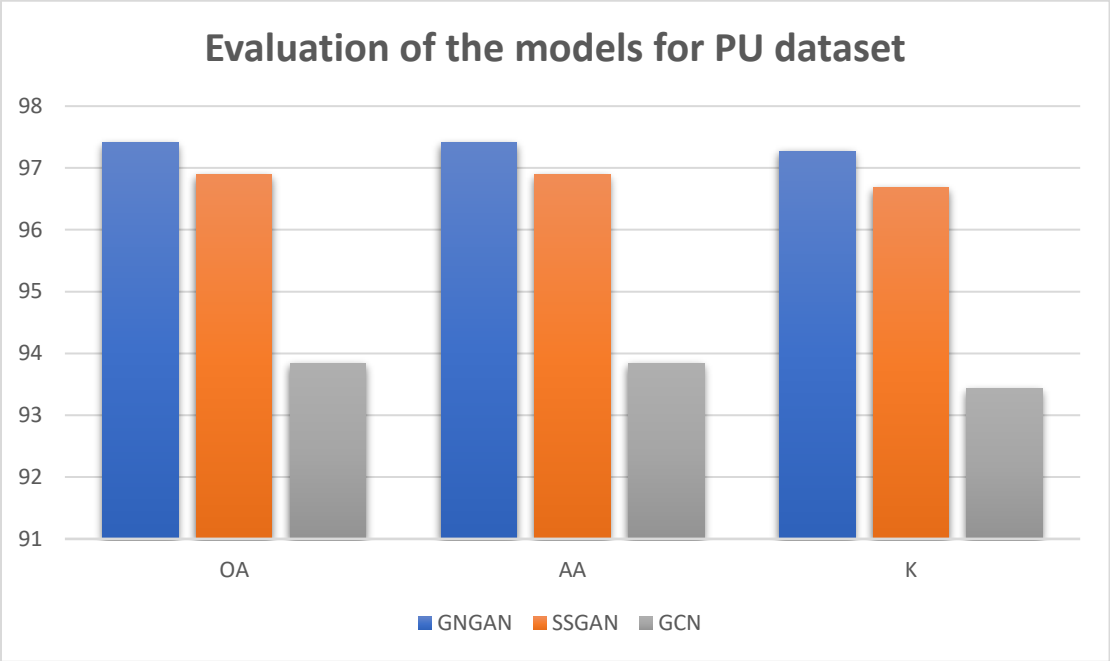


Figure 6.7: Bar Chart of models' evaluation result using Pavia University dataset

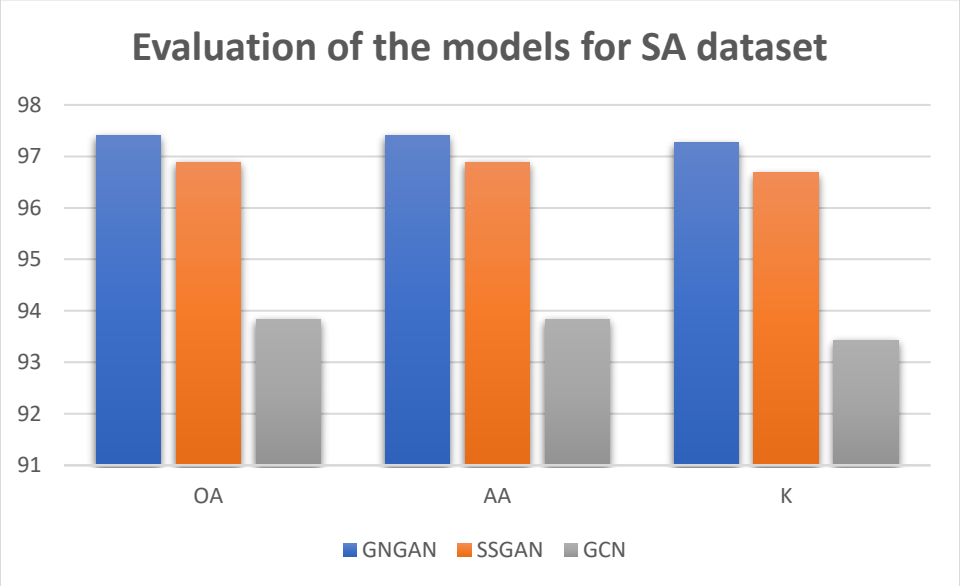
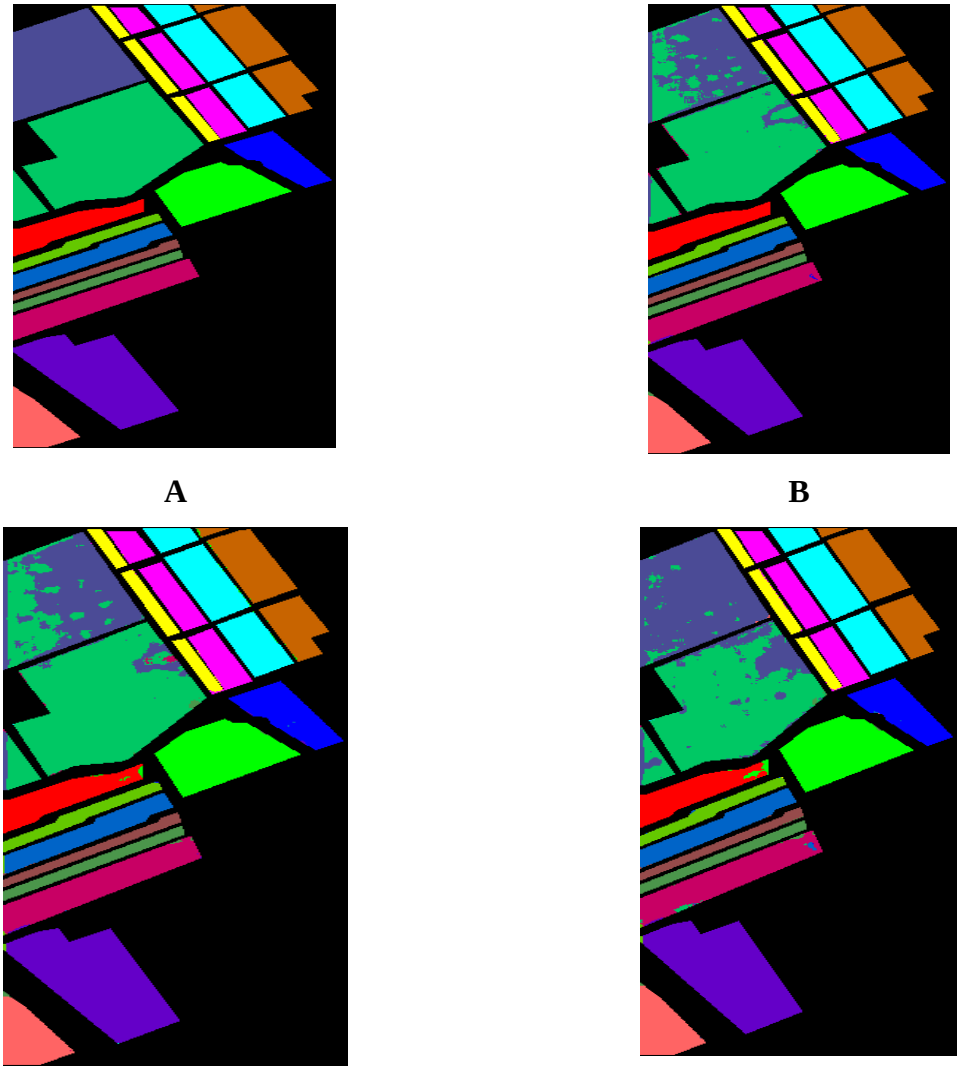


Figure 6.8: Bar Chart of models' evaluation result using Salina Valley dataset

6.3.4. Classification map for the models

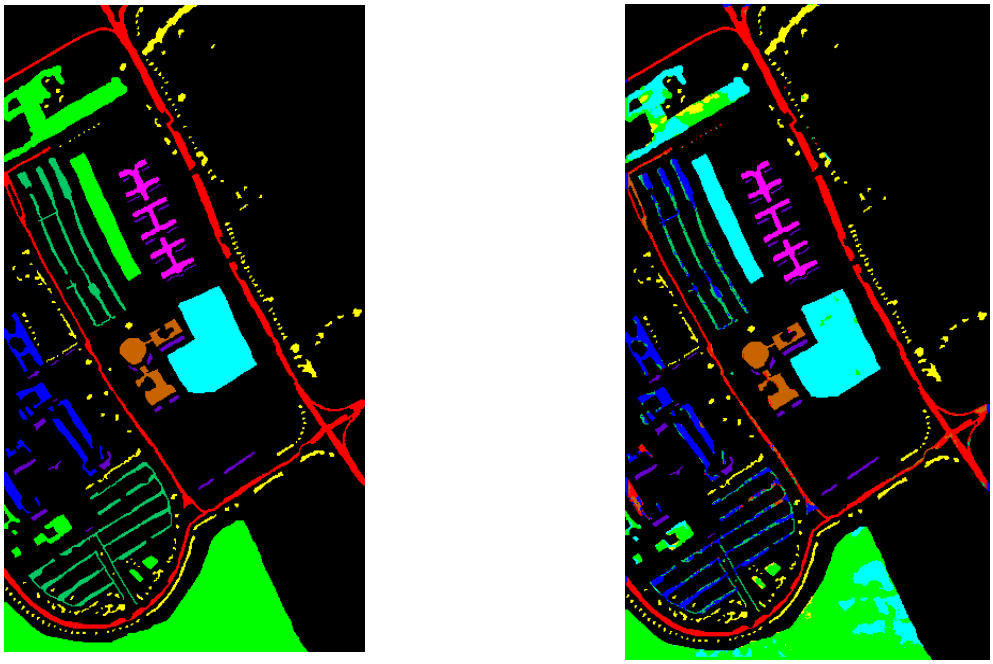
The ground truth of HSI is a 2D map where each foreground pixel categorized in their respective class and distinguished by different color. These classification map were produced after feeding the entire dataset including training, testing and discarded samples to the three model.



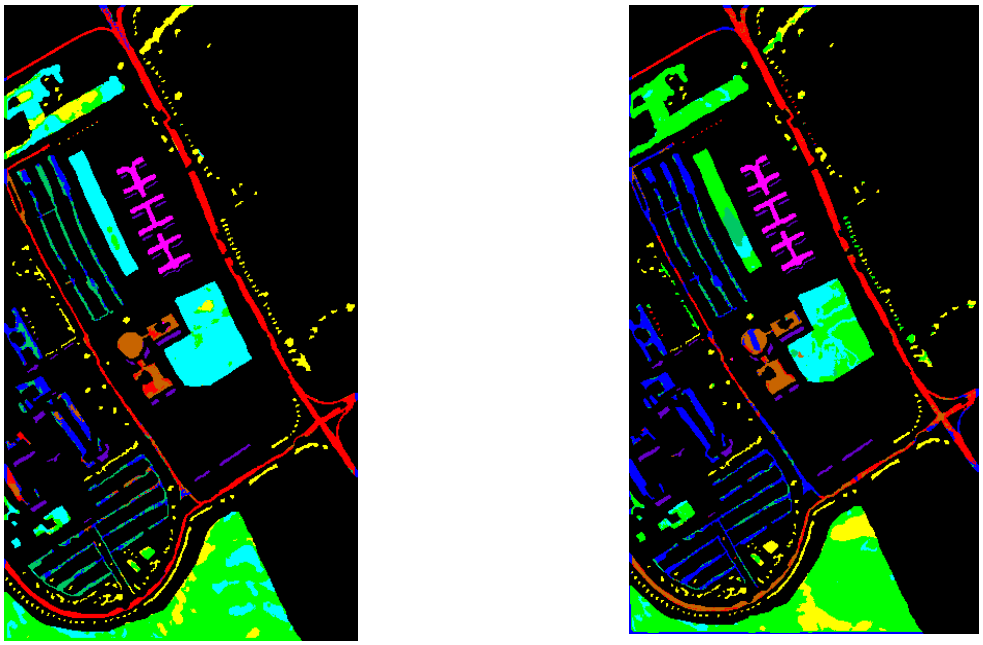
A **B**

C **D**

Figure 6.9: classification map for Salina Valley dataset. A) Ground truth, B) SSGAN, C) GCN, D) GNGAN



A **B**



C **D**

Figure 6.10: classification map for Pavia University dataset. A) Ground truth, B) SSGAN, C) GCN, D) GNGAN

6.3.5. Comparison of Combination mechanisms

In discriminator of GNGAN, Convolutional block and Graph network block were connected with concatenation layer after it was found a better alternative compared to other combination method through experiment. In order to find an effective method for combination mechanism, three combination methods were considered which were: Concatenation **(CN)**, Hadamard product **(HP)** and Element wise addition **(EWA)**.

Table 6.6: Comparison among Combination mechanisms for GNGAN

Evaluation Matrices	Combination Mechanism		
	CN	HP	EWA
Salina Valley			
OA	97.41	92.79	90.31
AA	97.41	92.79	90.31
K	97.21	92.30	89.66
Pavia University			
OA	85.79	78.53	72.13
AA	85.79	78.53	72.13
K	84.01	75.85	68.65

6.3.6. Impact of epoch

The GNGAN was trained using 200 epochs or 10,000 iteration which means each epoch contains 50 batches in both datasets and the product of epoch with number of batches is number of iteration. The result showed that GNGAN perform better when it trained in 200 epochs compared to 50, 100 and 150 epochs. In the following table shows, the performance of GNGAN is compared after trained in different epoch ranges from 50 to 200 using evaluation matrices.

Table 6.7: Impact of epoch on GNGAN for SV and PU

Evaluation Matrices	Epochs			
	50	100	150	200
Salina Valley				
OA	87.50	91.64	96.07	97.41
AA	87.50	91.64	96.07	97.41
K	86.67	91.08	95.81	97.21
Pavia University				
OA	75.18	81.80	84.88	85.79
AA	75.18	81.80	84.88	85.79
K	72.07	79.53	82.99	84.01

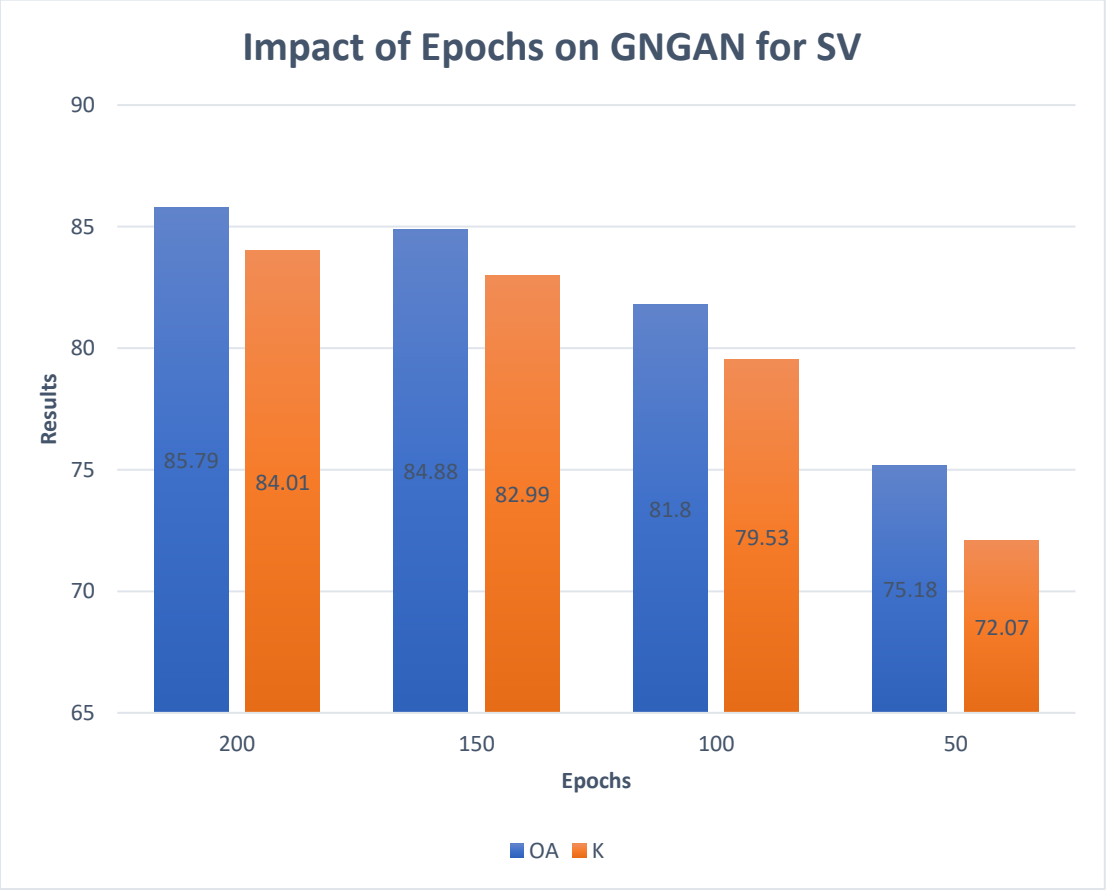


Figure 6.11: Bar chart of Impact of epoch on GNGAN for SV

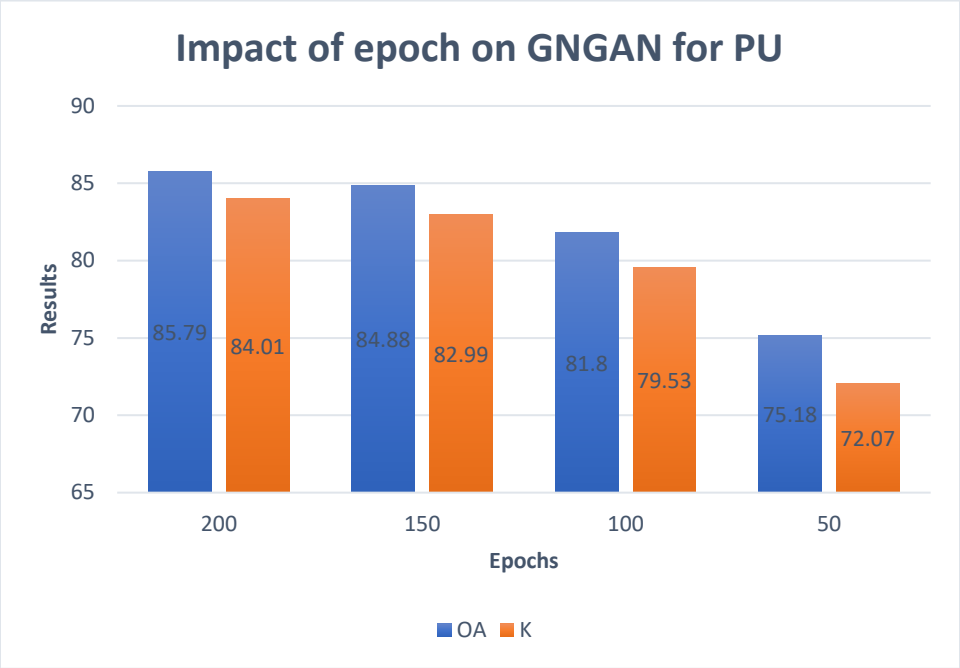


Figure 6.12: Bar chart of Impact of epoch on GNGAN for PU

6.3.7. Influence of number of training samples

The GNGAN model is a semi-supervised classification model that is used when there is small number of labeled training dataset. In this study, 100 training samples per class for both datasets, were used during training. The value of these samples were selected after conducting experiments on various number of samples per class and compare the performance of the model with them. The following tables shows the performance of GNGAN with various number of training samples for both datasets.

Table 6.8: Influence of number of training sample on GNGAN for PU

No of training sample per class	50	80	100
Total no of training sample	450	720	900
Train ratio to total no of samples before data balancing	1.05%	1.6%	2.1%

Train ratio to total no of samples After data balancing	5.27%	8.4%	10.5%
Test result of GNGAN with OA	77.62	82.18	85.79
Test result of GNGAN with AA	77.61	82.18	85.79
Test result of GNGAN with K	74.82	79.95	84.01

Table 6.9: Influence of number of training sample on GNGAN for SV

No of training sample per class	50	80	100
Total no of training sample	800	1280	1600
Train ratio to total no of samples before data balancing	1.4%	2.3%	2.9%
Train ratio to total no of samples After data balancing	5.4%	8.7%	10.9%
Test result of GNGAN with OA	94.86	95.72	97.41
Test result of GNGAN with AA	94.86	95.72	97.41
Test result of GNGAN with K	94.52	95.44	97.21

6.4. Discussion

In this section, the test results obtained from GNGAN, SSGAN and GCN using Overall accuracy (OA), Each and Average Accuracy (EA) and (AA), and Kappa Coefficient (K) are discussed to interpret what the result implicate for each evaluation matrix. GNGAN performs better that the other models based on the results from OA, AA and K. But when the results compared based on EA, GNGAN performs better in some classes but performs less in other classes.

The OA and AA results for each model have same value due to the same number of test samples for each class in both datasets. The OA matrix tells us the number of correctly classified samples over total number samples whereas EA matrix tells us the number of correctly classified samples in single class over total number of sample in single class. AA is the average of EA for all classes.

K is used to control the instances that may be classified correctly by chance. If K has higher value, it means the classifier has a good performance.

6.4.1. Effect of Convolutional block

GNGAN contains Convolutional and Graph network block where as SSGAN contains Convolutional block same with that of GNGAN. GCN model on the other hand contains Graph network block same with that of GNGAN. The effect of the convolutional block can be studied by comparing the performance of GNGAN with that of GCN model.

When OA and AA are used as evaluation matrix, the result showed that GNGAN outperforms GCN model by scoring 97.41% and 85.79% compared to 93.84% and 80.49% using Salina Valley and Pavia University dataset respectively. This shows GNGAN model can classify accurately compared with GCN model due to the presence of Convolutional block in the GNGAN for utilizing Euclidian structure of Hyperspectral image.

When K is used as evaluation matrix, the result showed that GNGAN outperforms GCN model by scoring 97.21% and 84.01% compared to 93.43% and 78.05% using Salina Valley and Pavia University datasets respectively. There is only small gap between OA and K which shows that GNGAN classify all classes in reliable manner based on the performance in the case of both datasets.

Based on the result, we can generalized GNGAN performs better than GCN because for utilizing both Euclidian and Non-Euclidian structures of HSI by Convolutional block and Graph network block respectively. GCN model has the exact same structure of Graph network block and trained in same environment, so we can say the difference in performance is caused by the presence of Convolutional block. This result address research question one.

6.4.2. Effect of Graph network block

When OA and AA are used as evaluation matrix, the result showed that GNGAN outperforms SSGAN model by scoring 97.41% and 85.79% compared to 96.89% and 82.50% using Salina Valley and Pavia University dataset respectively. This shows GNGAN model can classify accurately compared with SSGAN model due to the presence of Graph network block in the GNGAN for utilizing non-Euclidian structure of Hyperspectral image.

When K is used as evaluation matrix, the result showed that GNGAN outperforms SSGAN model by scoring 97.21% and 84.01% compared to 96.68% and 80.31% using Salina Valley and Pavia University datasets respectively. There is only small gap between OA and K which shows that GNGAN classify all classes in reliable manner based on the performance in the case of both datasets.

Based on the result, we can generalized GNGAN performs better than SSGAN because for utilizing both Euclidian and Non-Euclidian structures of HSI by Convolutional block and Graph network block respectively. SSGAN model has the exact same structure of GNGAN except not having Graph network block and combination mechanism and trained in same environment, so we can say the difference in performance is caused by the presence of Graph network block. This result address research question two.

6.4.3. Effect of Concatenation mechanism

Concatenation CN, Hadamard Product HP and Element wise Addition EWA were compared as combination mechanisms for Convolutional block and Graph network block in GNGAN and CN outperform the others during testing. GNGAN has better performance when using CN as combination mechanism by scoring 97.41% and 85.79% compared with when GNGAN uses HP by scoring 92.79% and 78.53% and EWA by scoring 90.31% and 72.13% using Salina Valley and Pavia University datasets.

Based on the result, we can generalize that Concatenation for combination mechanism is better than other combination mechanism such as Hadamard product and Element wise Addition. Therefore research question three is addressed by this result.

6.4.4. Error Analysis

In this section, EA is used as metrics to discuss the drawback of GNGAN compared with SSGAN and GCN for classification of certain classes in both Salina Valley and Pavia University datasets. Overall GNGAN has better performance and it perform better in some classes. But in other classes, SSGAN and GCN perform better which is summarized in the following table.

Table 6.10: Error Analysis for GNGAN compare with SSGAN

No	Classes	GNGAN	SSGAN
Salina Valley dataset			
1	Broccoli green weeds 2	99.4	99.8
2	Corn senesced green weeds	95.8	98.9
3	Grapes untrained	78.4	94.9
4	Lettuce romaine 5wk	96.7	98.0
5	Lettuce romaine 6wk	97.2	99.6
6	Vineyard vertical trellis	98.9	99.4
Pavia University dataset			
7	Painted metal sheet	83.1	91.3
8	Shadows	86.3	91.5

SSGAN performs better than GNGAN compared with EA metrics for the above classes in both Salina Valley and Pavia University datasets. These may be caused by two reasons due to the resemblance of SSGAN and GNGAN. These reasons are: the effect of concatenation method and the graphs used in Graph network model may has shallow features for certain classes which makes them not enough for HSI classification to get abundant information (*Guo F. et al, 2021*).

Table 6.11: Error Analysis for GNGAN compare with GCN

No	Classes	GNGAN	GCN
Salina Valley dataset			
1	Broccoli green weeds 2	99.4	99.9
2	Corn senesced green weeds	95.8	98.5
3	Soil vineyard develop	98.5	100
4	Lettuce romaine 5wk	96.7	97.2
5	Lettuce romaine 6wk	97.2	98.2
6	Vineyard untrained	94.1	98.5

Pavia University dataset			
7	Bitumen	38.4	64.8
8	Painted metal sheet	83.1	93.2
9	Asphalt	88.5	90.5

GCN performs better than GNGAN compared with EA metrics for the above classes in both Salina Valley and Pavia University datasets. These may be caused by two reasons due to the resemblance of GCN and Graph network block in GNGAN. These reasons are: the effect of concatenation method and the effect of convolutional block.

In most the classes mentioned on tables above, GNGAN has lower performance that SSGAN and GCN which shows the effect of concatenation mechanism that cause in the reduction of the performance.

Chapter Seven

Conclusion and Further work

7.1. Conclusion

Hyperspectral Images are used in many application areas due to their features for identifying the composition of objects in the image using spectral bands ranges from infrared to ultraviolet electromagnetic spectrum. HSI can be described in two forms, one in Euclidean structure whereas the second form is in non-Euclidean structure form. In order to increase the performance of HSI classifications, model should consider both forms. HSI images are also known for having small amount of labeled samples and due to this case, supervised classification models has lower performance compare to semi-supervised classification models. GAN are known as one of semi-supervised classification model for HSI.

Generative Adversarial Networks are used as a semi-supervised classification model for Hyperspectral images in many prior works. But most of the prior works considers joint spatial spectral feature extraction out of Euclidian structure from Hyperspectral image cubes. The relationship among pixels based on their spectral features which form graph structures, has not been done via GAN based semi-supervised classification models. In this work, the proposed model Graph Network based Generative Adversarial Network model uses both extraction method to accommodate both structures.

The proposed GNGAN model has one Generator which produces 3D HSI cubes with the shape of (7, 7, 50) and has Discriminator which operates in unsupervised and supervised mode. The discriminator in unsupervised mode is used to discriminate the generated cubes and real unlabeled cubes as fake or real whereas the discriminator in supervised mode classify the real labeled HSI inputs in the form of cubes and graphs.

The discriminator is composed of Convolutional block, Graph network block, Concatenation mechanism for composition and layers of fully connected layers. Convolutional block is responsible for utilizing Euclidian structures of HSI for extraction of joint spatial spectral feature extraction, Graph network block is responsible for utilizing non-Euclidian structure of HSI by

using spectral relationship among pixels. Concatenation mechanism was selected for combining Graph network block and Convolutional block after comparing with other combination mechanisms like Hadamard product and Element wise Addition methods.

Compared to other model like SSGAN and GCN, GNGAN has a better performance as semi-supervised classification model when it is evaluated with overall accuracy, average accuracy and kappa coefficient as matrices and using Salina Valley and Pavia University datasets. The GNGAN has a performance of 97.41 % and 97.27 % of overall accuracy and kappa coefficient respectively for Salina Valley datasets. It has a performance of 85.79% and 84.01% of overall accuracy and Kappa coefficient respectively for Pavia University dataset. GNGAN outperform both SSGAN and GCN by 0.5% and 3.66% for Salina Valley dataset respectively and 3.8% and 6.17% for Pavia University dataset respectively.

7.2. Further work

GNGAN has its own limitation on utilizing HSI feature since the performance of the model is less than to that of SSGAN and GCN in some classes. The following three suggestions are discussed below for further works in order to improve the performance of the GNGAN or develop new model that solves the drawback of GNGAN.

The number of the graph's nodes are fixed throughout the convolution process which reduce the final classification performance if the input graph is not accurately converted from the patches. This may cause the misrepresentation of HSI by the graph which results in misclassification. To overcome such problem two recommendation can be suggested. These are: construction of graph generated from non-local pixels to get abundant information on the HSI structure and applying graph aggregation such as diffusional graph pooling to the convolution process to change the number of node throughout the convolution process.

The graph is generated from HSI patches by using spectral relationship among the pixels. But graph can represents spatial relationship among pixels and super pixels. To increase the performance of GNGAN, one recommendation can be suggested which is to utilize both spatial and spectral graphs in Graph network block of GNGAN.

The experiment result shows GNGAN performs less in some class and in most of these classes, both SSGAN and GCN performs better than GNGAN. These shows the combination mechanism can be improved to increase the performance of GNGAN. To overcome, two recommendation can be suggested. These are: changing the position where Convolutional block and Graph network block combine and utilizing different combination mechanism to improve the performance of GNGAN even more.

Reference

Ahmad, M., Khan, A. M., Mazzara, M., Distefano, S., Ali, M., & Sarfraz, M. S. (2020). A fast and compact 3-D CNN for hyperspectral image classification. IEEE Geoscience and Remote Sensing Letters.

Arjovsky, M., Chintala, S., & Bottou, L. (2017, July). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.

Arun, P. V., Buddhiraju, K. M., Porwal, A., & Chanussot, J. (2020). CNN-based super- resolution of hyperspectral images. IEEE Transactions on Geoscience and Remote Sensing, 58(9), 6106 -6121.

Aydemir, M. S., & Bilgin, G. (2017). Semisupervised hyperspectral image classification using small sample sizes. IEEE Geoscience and Remote Sensing Letters, 14(5), 621-625.

Ding, Y., Zhao, X., Zhang, Z., Cai, W., & Yang, N. (2021). Multiscale graph sample and aggregate network with context-aware learning for hyperspectral image classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 4561-4572.

ElMasry, G., Gou, P., & Al-Rejaie, S. (2021). Effectiveness of specularity removal from hyperspectral images on the quality of spectral signatures of food products. Journal of Food Engineering, 289, 110148.

Enenkel, M., Steiner, C., Mistelbauer, T., Dorigo, W., Wagner, W., See, L., Atzberger, C.,Schneider, S. and Rogenhofer, E., 2016. A combined satellite-derived drought indicator to support humanitarian aid organizations. Remote Sensing, 8(4), p.340

Feng, J., Feng, X., Chen, J., Cao, X., Zhang, X., Jiao, L., & Yu, T. (2020). Generative adversarial networks based on collaborative learning and attention mechanism for hyperspectral image classification. Remote Sensing, 12(7), 1149.

Feng, J., Wu, X., Chen, J., Zhang, X., Tang, X., & Li, D. (2019, July). Joint multilayer spatial-spectral classification of hyperspectral images based on CNN and ConvLSTM. In IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium (pp. 588-591). IEEE.

Ge, L., Huang, B., Wei, W., & Pan, Z. (2021). Semi-Supervised classification of hyperspectral images using discrete nonlocal variation Potts Model. Mathematical Foundations of Computing, 4(2), 73.

Gogineni, Rajesh & Chaturvedi, Ashvini. (2019). Hyperspectral Image Classification. 10.5772/intechopen.88925

Goodfellow, Ian & Pouget-Abadie, Jean & Mirza, Mehdi & Xu, Bing & WardeFarley, David & Ozair, Sherjil & Courville, Aaron & Bengio, Y.. (2014). Generative Adversarial Networks. Advances in Neural Information Processing Systems. 3.10.1145/3422622

Gopal Krishna; 2019; Indian Pines, Salina Valley and Pavia University; available at: https://github.com/gokriznastic/HybridSN/tree/master/data

Guilloteau, C., Oberlin, T., Berné, O., & Dobigeon, N. (2020, June). Fusion of hyperspectral and multispectral infrared astronomical images. In 2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM) (pp. 1-5). IEEE.

Guo, B., Gunn, S. R., Damper, R. I., & Nelson, J. D. (2008). Customizing kernel functions for SVM -based hyperspectral image classification. IEEE Transactions on Image Processing, 17(4), 622 -629.

Guo, F., Li, Z., Xin, Z., Zhu, X., Wang, L., & Zhang, J. (2021). Dual Graph U-Nets for Hyperspectral Image Classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 8160-8170.

Hahn, A., Tummala, M., & Scrofani, J. (2019, December). Extended Semi-Supervised Learning GAN for Hyperspectral Imagery Classification. In 2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS) (pp. 1-6). IEEE.

He, S., Tang, H., Lu, X., Yan, H., & Wang, N. (2021). MSHCNet: Multi-Stream Hybridized Convolutional Networks with Mixed Statistics in Euclidean/Non-Euclidean Spaces and Its Application to Hyperspectral Image Classification. arXiv preprint arXiv:2110.03346.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. arXiv preprint arXiv:1706.08500

Hong, D., Gao, L., Yao, J., Zhang, B., Plaza, A., & Chanussot, J. (2020). Graph convolutional networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 59(7), 5966-5978.

Hsieh, T. H., & Kiang, J. F. (2020). Comparison of CNN algorithms on hyperspectral image classification in agricultural lands. Sensors, 20(6), 173

Hu, W. S., Li, H. C., Pan, L., Li, W., Tao, R., & Du, Q. (2020). Spatial–spectral feature extraction via deep ConvLSTM neural networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 58(6), 4237-4250.

Hu, Y., An, R., Wang, B., Xing, F., & Ju, F. (2020). Shape adaptive neighborhood information-based semi-supervised learning for hyperspectral image classification. Remote Sensing, 12(18), 2976.

Jakub L & Vladimir B. (2019). GAN in Action, Deep learning with Generative Adversarial Network, Manning Publications, page 141.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8110-8119).

Khan, M. J., Khan, H. S., Yousaf, A., Khurshid, K., & Abbas, A. (2018). Modern trends in hyperspectral image analysis: A review. IEEE Access, 6, 14118-14129.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Li, C., Yang, S. X., Yang, Y., Gao, H., Zhao, J., Qu, X., & Gao, J. (2018). Hyperspectral remote sensing image classification based on maximum overlap pooling convolutional neural network. Sensors, 18(10), 3587.

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2015). Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.

Li, Y., Zhang, H., & Shen, Q. (2017). Spectral–spatial classification of hyperspectral imagery with 3D convolutional neural network. Remote Sensing, 9(1), 67.

Liang, H., Bao, W., & Shen, X. (2021). Adaptive weighting feature fusion approach based on generative adversarial network for hyperspectral image classification. Remote Sensing, 13(2), 198.

Liu, S., Cao, Y., Wang, Y., Peng, J., Mathiopoulos, P. T., & Li, Y. (2021). DFL-LC: Deep feature learning with label consistencies for hyperspectral image classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 3669-3681

Liu, Q., Xiao, L., Yang, J., & Wei, Z. (2020). CNN-enhanced graph convolutional network with pixel -and superpixel-level feature fusion for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 59(10), 8657-8671.

Liu, Z., & Zhou, J. (2020). Introduction to graph neural networks. Synthesis Lectures on Artificial Intelligence and Machine Learning, 14(2), 1-127.

Lv, W., & Wang, X. (2020). Overview of hyperspectral image classification. Journal of Sensors, 2020.

Mather, P. M., & Koch, M. (2011). Computer processing of remotely-sensed images: an introduction. John Wiley & Sons.

Menilk Sahlu, 2021, A Branching Convolutional Encoder-based Spatio-spectral Dimensionality Reduction Model for Hyperspectral Image Classification and Change Detection, Adama Science and Technology University, available at http://etd.astu.edu.et/handle/123456789/1684

Mehta, A., Sinha, H., Narang, P., & Mandal, M. (2020). Hidegan: A hyperspectral-guided image dehazing gan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 212-213).

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957.

Mohan, Alkha & Venkatesan, M.. (2020). Spatiospectral Feature Extraction and Classification of Hyperspectral Images Using 3D-CNN

Mou, L., Lu, X., Li, X., & Zhu, X. X. (2020). Nonlocal graph convolutional networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 58(12), 8246 -8257.

Odena, A. (2016). Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583.

Sawant, S. S., & Prabukumar, M. (2020). A review on graph-based semi-supervised learning methods for hyperspectral image classification. The Egyptian Journal of Remote Sensing and Space Science, 23(2), 243-248.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. IEEE transactions on neural networks, 20(1), 61-80.

Shi, Changjiang & Zhang, Zhijie & Zhang, Wanchang & Zhang, Chuanrong & Xu, Qiang. (2022). Learning Multiscale Temporal–Spatial–Spectral Features via a Multipath Convolutional LSTM

Neural Network for Change Detection With Hyperspectral Images. IEEE Transactions on Geoscience and Remote Sensing. 10.1109/TGRS.2022.3176642.

Sudharsan, S., Hemalatha, R., & Radha, S. (2019, March). A survey on hyperspectral imaging for mineral exploration using machine learning algorithms. In 2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET) (pp. 206-212). IEEE.

Thomas Kipf, (2016, September, 30). Graph Convolutional Network, https://tkipf.github.io/graph -convolutional-networks.

Vishvanathan, Sowmya & Kp, Soman & Hassaballah, Mahmoud. (2019). Hyperspectral Image: Fundamentals and Advances. 10.1007/978-3-030-03000-1_16.

Wan, S., Gong, C., Zhong, P., Pan, S., Li, G., & Yang, J. (2020). Hyperspectral image classification with context-aware dynamic graph convolutional network. IEEE Transactions on Geoscience and Remote Sensing, 59(1), 597-612.

Xie, T., & Grossman, J. C. (2018). Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. Physical review letters, 120(14), 145301.

Xie, W., Zhang, J., Lei, J., Li, Y., & Jia, X. (2021). Self-spectral learning with GAN based spectral –spatial target detection for hyperspectral image. Neural Networks, 142, 375-387.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks?. arXiv preprint arXiv:1810.00826..

Y. Yan, Y. Zhao, H.-f. Xue, X.-d. Kou, and Y. Liu, "Integration of spatial-spectral information for hyperspectral image classification," in 2010 Second IITA International Conference on Geoscience and Remote Sensing, pp. 242–245, Qingdao, 2010

Yang, P., Tong, L., Qian, B., Gao, Z., Yu, J., & Xiao, C. (2020). Hyperspectral image classification with spectral and spatial graph using inductive representation learning network. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 791-800.

Yang, Y., Ma, B., Liu, X., Zhao, L., & Huang, S. (2021). Gsap: A global structure attention pooling method for graph-based visual place recognition. Remote Sensing, 13(8), 1467.

Zeng, H., Liu, Q., Zhang, M., Han, X., & Wang, Y. (2020). Semi-supervised hyperspectral image classification with graph clustering convolutional networks. arXiv preprint arXiv:2012.10932.

Zhan, Y., Hu, D., Wang, Y., & Yu, X. (2017). Semisupervised hyperspectral image classification based on generative adversarial networks. IEEE Geoscience and Remote Sensing Letters, 15(2), 212-216.

Zhang, F., Bai, J., Zhang, J., Xiao, Z., & Pei, C. (2020). An optimized training method for GAN-based hyperspectral image classification. IEEE Geoscience and Remote Sensing Letters, 18(10), 1791-1795.

Zhang, H., Zou, J., & Zhang, L. (2022). EMS-GCN: An End-to-End Mixhop Superpixel-Based Graph Convolutional Network for Hyperspectral Image Classification. IEEE Transactions on Geoscience and Remote Sensing, 60, 1-16.

Zhang, M., Luo, H., Song, W., Mei, H., & Su, C. (2021). Spectral-Spatial Offset Graph Convolutional Networks for Hyperspectral Image Classification. Remote Sensing, 13(21), 4342.

Zhao, Z., Hu, D., Wang, H., & Yu, X. (2021). Center Attention Network for Hyperspectral Image Classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.

Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision (pp. 2223-2232).

Zhu, L., Chen, Y., Ghamisi, P., & Benediktsson, J. A. (2018). Generative adversarial networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 56(9), 5046-5063.

Appendix A: Result of Comparison models

A.1. Results using Salina Valley Dataset

A.1.1. SSGAN model

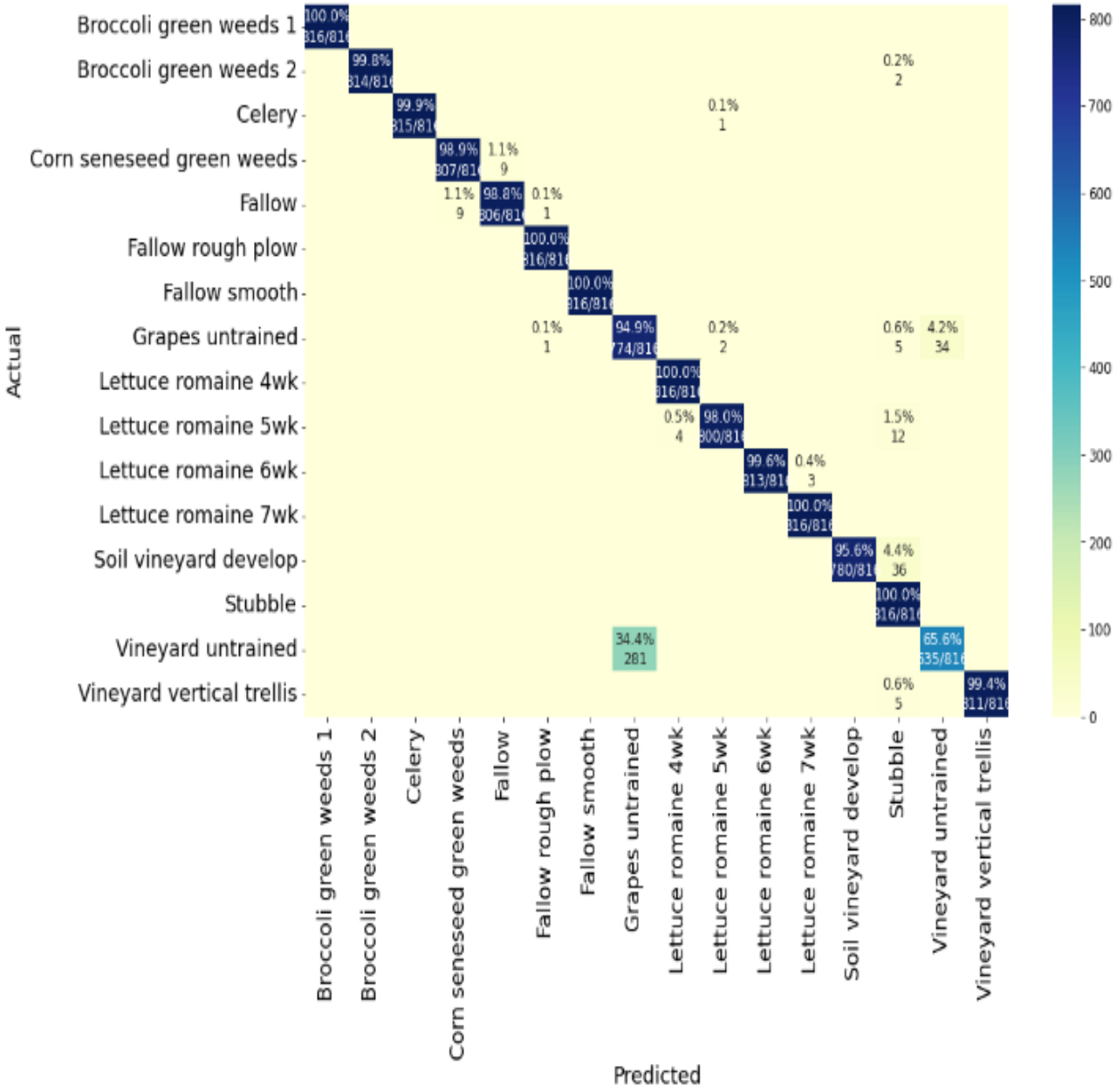


Figure A.1: Confusion Matrix for SSGAN using Salina Valley

A.1.2. GCN model

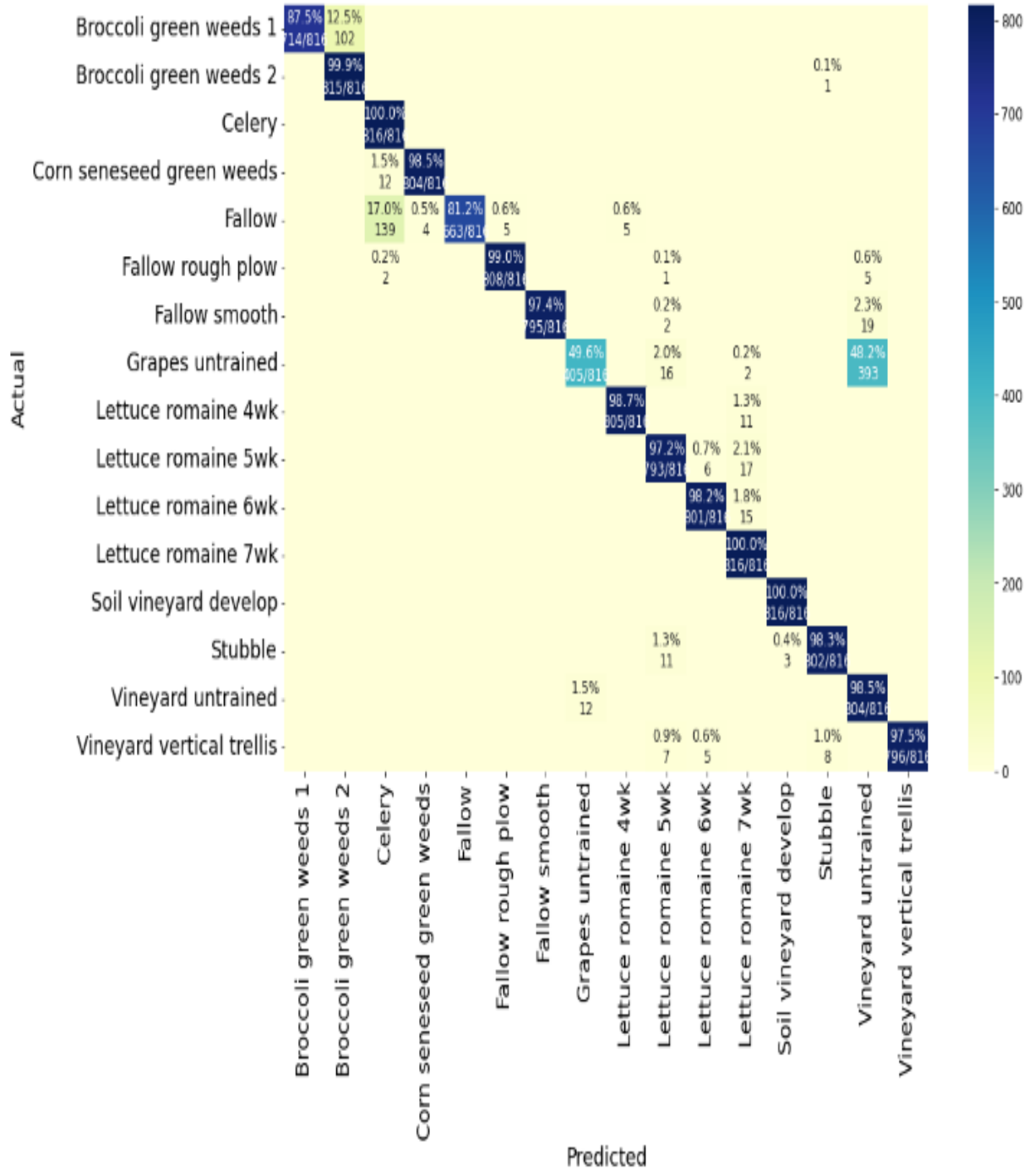


Figure A.2: Confusion Matrix for GCN using Salina Valley

A.2. Result using Pavia University Dataset

A.2.1. SSGAN model

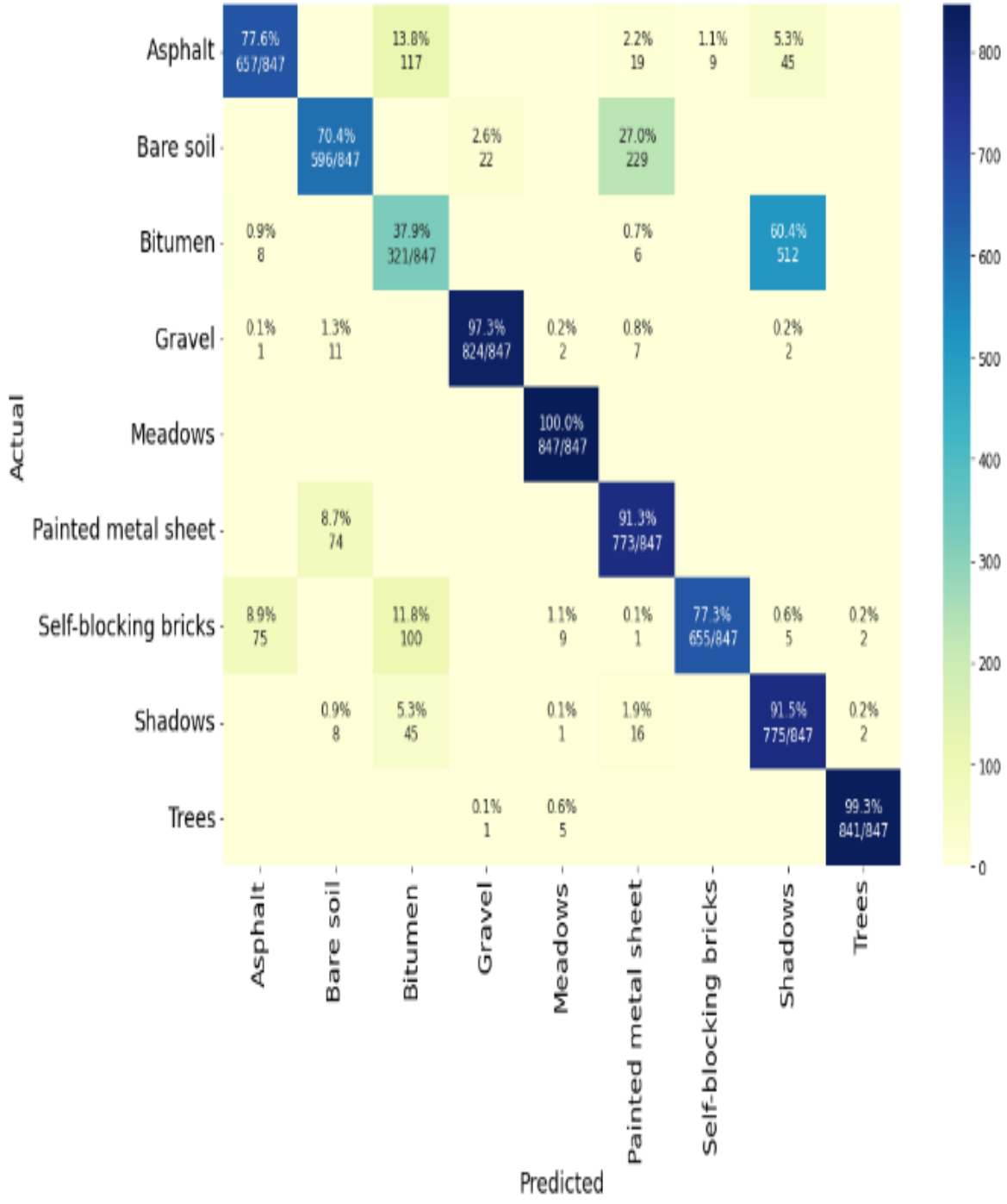


Figure A.3: Confusion Matrix for SSGAN using Pavia University

A.2.2. GCN model

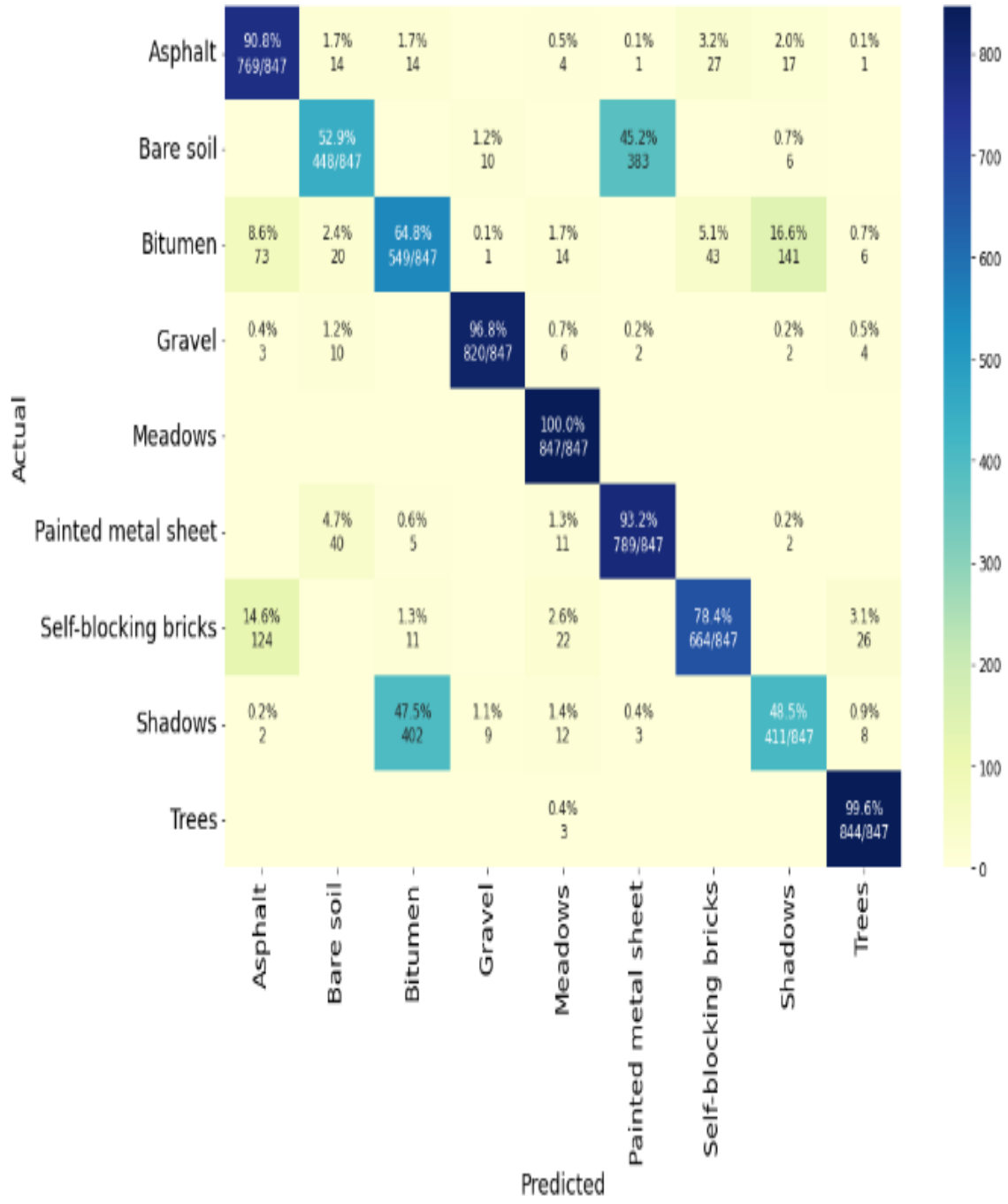


Figure A.4: Confusion Matrix for GCN using Pavia University

Appendix B: Sample of code

The proposed model was implemented by using different python libraries and Tensorflow framework. Here are the imported python libraries.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Conv3D, Conv2DTranspose, Conv3DTranspose, Flatten, Dense, Reshape, LeakyReLU, Lambda, BatchNormalization, Activation
from tensorflow.keras.layers import Dropout, Input, concatenate, GlobalAveragePooling2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras import utils, activations
from tensorflow.keras import backend as K

import math

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, cohen_kappa_score

import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
import os
import random

import spectral
import spektral
from spektral.layers import GCNConv, DiffPool, GlobalAttentionPool
from spektral.utils import gcn_filter

from PIL import Image
from operator import truediv
import pandas as pd
from plotly.offline import init_notebook_mode
import plotly.graph_objs as go
from matplotlib.pyplot import cm
import seaborn as sns
```

Code snippet B.1: imported libraries

The proposed model was trained using customized training method by using *"train_on_batch ()"* Keras on built function from model sub library.

```
def train(X, Y, G, L, disc, gcn, disc_sup_model, gen_model, disc_uns_model, gan_model, latent_dim, epoch, batch_size, Cno, sum_report, name):
  # find the numbers of batches in a single epoch
  batch_per_epoch = int(X.shape[0]/batch_size)
  n_step = batch_per_epoch*epoch
  half_batch = int(batch_size/2)

  for i in range(n_step):
    Xsub, Ysub, Gsub, Lsub = superviseddata(X, Y, G, L, half_batch, Cno)
    discout = disc(Xsub)
    gcnout = gcn([Gsub, Lsub])
    c_loss, c_acc = disc_sup_model.train_on_batch([discout, gcnout], Ysub)

    real_image, real_label = real_unlabeled(X, half_batch, Cno)

    fake_image, fake_label = generate_fake_samples(gen_model, latent_dim, batch_size)

    d_loss_real = disc_uns_model.train_on_batch(real_image, real_label)
    d_loss_fake = disc_uns_model.train_on_batch(fake_image, fake_label)



    d_loss = 0.5*(d_loss_real + d_loss_fake)



    Z = generate_noise(latent_dim, batch_size)
    R = np.ones((batch_size, 1))
    g_loss = gan_model.train_on_batch(Z, R)
    if (i+1) % batch_per_epoch == 0:
      a = i + 1
      sum_report[1].append(d_loss)
      sum_report[0].append(g_loss)
      sum_report[2].append(c_loss)
      sum_report[3].append(c_acc * 100)
      summery_performance(a, g_loss, d_loss, c_loss, c_acc, batch_per_epoch, disc_sup_model, name)
```

Code snippet B.2: Training function