Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Brage Gaasø Samsonsen

Enhanced Graph Autoencoders for Hyperspectral Anomaly Detection

Master's thesis in Electronic System Design and Innovation
Supervisor: Milica Orlandić
Co-supervisor: Vinay Chakravarthi Gogineni
June 2023

NTNU
Norwegian University of
Science and Technology

Brage Gaasø Samsonsen

Enhanced Graph Autoencoders for Hyperspectral Anomaly Detection

Master's thesis in Electronic System Design and Innovation
Supervisor: Milica Orlandić
Co-supervisor: Vinay Chakravarthi Gogineni
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis presents multiple novel Hyperspectral Anomaly Detectors (HAD), all incorporating the Robust Graph Autoencoder (RGAE) as a backbone. The proposed HADs focus on enhancing detection performance through preprocessing techniques and modifications to network architecture. A new neural network architecture, the Interpolated Autoencoder, will also be introduced. Preprocessing methods, such as PCA, KPCA, and RPCA, are evaluated for their effectiveness on the input of the Autoencoder utilized by the RGAE.

The thesis introduces a Preprocessing based RGAE and the Multikernel RGAE (MK-RGAE) that utilizes KPCA on the network inputs. Additionally, various layer setups are explored to check if an increase in the complexity of the Robust Graph Autoencoder can result in better performance. Furthermore, the Interpolated Autoencoder is employed for some of the proposed Hyperspectral Anomaly Detectors, namely the Interpolated Graph Autoencoder (IGAE), the Kernel Interpolated Graph Autoencoder (K-IGAE), and the Multikernel Interpolated Graph Autoencoder (MK-IGAE). The two latter mentioned HADs employ KPCA with different configurations.

Evaluation and analysis of the proposed HADs are performed using the ABU datasets. These datasets comprise diverse hyperspectral scenes, such as airports, beaches, and urban areas. The comparative analysis of the proposed HADs highlights two standout methods, namely the Multikernel RGAE and the Kernel Interpolated Graph Autoencoder. These HADs significantly improve upon the baseline RGAE in terms of total average AUC scores. The MK-RGAE achieved a remarkable total average AUC score of 0.9701, surpassing the RGAE score of 0.9551. Similarly, the K-IGAE resulted in an impressive AUC score of 0.9632. However, the other proposed Hyperspectral Anomaly Detectors, including the Preprocessing-based RGAE, RGAE with modified layers, IGAE, and MK-IGAE, yielded less satisfactory results, failing to surpass the baseline RGAE's detection performance. Notably, the RGAE with additional layers demonstrated reduced computational costs compared to other proposed HADs, making it a feasible option for time cost reduction.

Considering the primary objective of enhancing the detection performance of the RGAE, the MK-RGAE and K-IGAE successfully achieved the desired results. Consequently, further refinement and advancement of these approaches should be explored.

Sammendrag

Denne masteroppgaven presenterer flere hyperspektrale anomali detektorer som alle baserer seg på en toppmoderne detektor ved navnet Robust Graph Autoencoder (RGAE). Alle foreslåtte detektorer i denne masteroppgaven vil ha som hovedmål å forbedre deteksjonsytelsen i sammenligning med RGAEen. Nye implementasjoner og endringer vil innebære nye preprosesseringsmetoder som PCA, KPCA og RPCA. Nye nettverksarkitekturer for det nevrale nettverket som er tatt i bruk av RGAEen vil også bli utforsket.

Masteroppgaven presenterer en preprosesseringsbasert RGAE og en hyperspektral anomali detektor med to preprosesseringsmetoder kalt Multikernel RGAE. En ny versjon av RGAEen med nye oppsett for lagene i det nevrale nettverket vil også bli lagt frem. Sist, men ikke minst, vil et nytt nettverk ved navn Interpolated Autoencoder erstatte Autoencoderen som er tatt i bruk av RGAEen. Detektorene som bruker dette nye nettverket er kalt Interpolated Graph Autoencoder, Kernel Interpolated Graph Autoencoder og Multikernel Interpolated Graph Autoencoder der de to sistnevnte benytter seg av ulike metoder for preprosessering.

For å evaluere alle de hyperspektrale anomali detektorene vil ABU datasettet bli benyttet. Dette består av flere hyperspektrale bilder av flyplasser, strender og urbane strøk. I sammenligningen av alle detektorene som blir foreslått er det to detektorer som skiller seg ut. Multikernel RGAEen viste seg å forbedre gjennomsnitts AUC verdien fra 0.9551 til 0.9701. En forbedring ble også lagt merke til for Kernel Interpolated Graph Autoencoderen, som oppnådde en gjennomsnittlig AUC verdi på 0.9632. De andre foreslåtte detektorene viste seg å ikke forbedre deteksjonen av anomalier like mye.

Med tanke på masteroppgavens mål om å forbedre deteksjonsytelsen til RGAEen ved å forslå en rekke nye hyperspektrale anomali detektorer, kan det konkluderes at Multikernel RGAEen og Kernel Interpolated Graph Autoencoderen nådde dette målet. Det kan derfor være interessant å videreutvikle disse to detektorene.

Preface

This Masters's thesis was conducted at the Norwegian University of Science and Technology as part of the Electronics Systems Design and Innovation study programme. It was written for the HYPSO mission in the time period from the 16. of January to the 12. of June 2023.

I would like to thank my supervisor Milica Orlandić for the support and motivation needed to accomplish the work done in this thesis. My co-supervisor, Vinay Chakravarthi Gogineni, has also been of great help providing technical guidance. Last, but not least, a big thanks goes to my fellow student Katinka Müller for motivation and constructive discussions.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Chapter 1

Introduction

This chapter presents the background needed to understand the thesis's need, purpose and motivation. The background will be explained in more detail in Chapter 2. An introduction to what the thesis will focus on and the contributions made will also be presented. The final section of the introduction will provide an overview of the contents and structure of the entire thesis.

1.1 Background Information

Hyperspectral imaging is the process of capturing measurements of the emission/reflection of objects across hundreds of wavelengths within the electromagnetic spectrum. These measurements can then be combined to construct a Hyperspectral Image (HSI) [1]. Since different materials have different reflective properties, hyperspectral imaging can be used in several remote sensing applications. Satellites utilizing this technology can be used for environmental monitoring and target detection [1].

Objects within an HSI that have spectral radiation that deviates from their surroundings are considered anomalous [2]. Hyperspectral anomaly detection is the process of identifying and locating these abnormal objects. As anomalies can often indicate the presence of certain materials or objects, hyperspectral anomaly detection is a useful tool for various target and surveillance applications [3]. A more detailed explanation of hyperspectral imagery and anomaly detection is presented in Chapter 2.

1.2 Motivation

This thesis is dedicated to the HYPerspectral Smallsat for OCEAN Observation mission, commonly called HYSPO, established under by the Norwegian University of Science and Technology. The objective of the HYPSO satellite mission is to conduct oceanic monitoring, such as detecting harmful algae blooms, phytoplankton and river plumes along the coast of Norway. A hyperspectral imaging system is installed on the HYPSO satellite, making it possible to monitor the ocean. In order to fulfil the objective, there is a need for an anomaly detection algorithm for the autonomous detection of the mentioned occurrences [4].

In recent years, there has been a substantial surge of interest in applying neural networks for hyperspectral anomaly detection. The utilization of Autoencoders (AE), Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN) are examples of network architectures utilized for such tasks [2, 5]. The recently proposed Robust Graph Autoencoder (RGAE) is a state-of-the-art Hyperspectral Anomaly Detector (HAD) that utilizes an Autoencoder, achieving impressive detection performances across a wide variety of datasets. A critical factor that makes Autoencoders excellent in hyperspectral anomaly detection is that it is an unsupervised method. What separates the RGAE from other HADs utilizing Autoencoders is the addition of a graph regularization term, giving it the possibility to obtain spatial information for a given Hyperspectral Image [6]. Considering that the Robust Graph Autoencoder (RGAE) is a recently proposed HAD, it holds significant promise for further exploration and development in terms of its detection

performance.

1.3 Project Objective and Description

This thesis aims to propose novel methodologies for hyperspectral anomaly detection by enhancing the existing Robust Graph Autoencoder method to realize the HYPSO mission. Furthermore, it is essential to note that the methodologies investigated and proposed in this thesis have the potential to address additional use cases within hyperspectral anomaly detection.

Multiple novel Hyperspectral Anomaly Detectors, all of which incorporate the RGAE as a foundational component, will therefore be proposed. These new proposed Hyperspectral Anomaly Detectors focus on adding preprocessing or modifying the network architecture of the state-of-the-art detector with the ultimate goal of increasing the detection performance. The first proposed method is the RGAE with the introduction of preprocessing for the neural network input. The preprocessing methods tested are the Principal Component Analysis (PCA), the Kernel-PCA (KPCA) and the Robust-PCA (RPCA) [7].

Preprocessing will also be utilized by the Multikernel RGAE. This is a HAD that utilizes two conventional RGAEs in parallel with the addition of KPCA on the network inputs. The output of each RGAE will then be combined using decision fusion. New layer setups will also be tested, adding more complexity to the Robust Graph Autoencoder.

The last HADs that are proposed include the use of a new neural network architecture, being the Interpolated Autoencoder. This neural network has not previously been used for hyperspectral anomaly detection, but has resulted in better generalized Autoencoders [8]. The proposed HADs that use this neural network are the Interpolated Graph Autoencoder, Kernel Interpolated Graph Autoencoder and the Multikernel Interpolated Graph Autoencoder. The two latter mentioned detectors utilize KPCA for the network inputs.

To measure and analyze the detection performance of new contributions and proposed enhancements, the Airport-Beach-Urban (ABU) datasets will be utilized [9]. This is several hyperspectral datasets from different scenery, being airports, beaches and urban areas. The dataset analysis that is presented in this thesis is conducted in collaboration with Katinka Müller, Master's student at the Norwegian University of Science and Technology. She is also working with the same datasets in her Master's thesis, which is also about hyperspectral anomaly detection utilizing deep neural networks.

Regarding limitations, this thesis will only focus on enhancing the Robust Graph Autoencoder when it comes to preexisting Hyperspectral Anomaly Detectors. Optimization will not be a big focus, as the goal of the thesis is to explore what methods that can result in higher detection performances. If any implementations achieve desired performance but still have to be optimized, the optimization problem will be addressed as a topic for future investigation.

Last but not least, it must be stated that some of the literature review performed for this master thesis was done during the fall of 2022 as a semester project by the same author and supervisors as this thesis [10]. The primary objective of the aforementioned semester project was to prepare students with the necessary skills and knowledge in preparation for their Master's thesis.

1.4 Structure of Thesis

The thesis comprises several chapters exploring various aspects of the research topic. This list provides an overview of the thesis structure and the chapter's contents.

- Chapter 1: This is the Introduction chapter, and it will provide information about the thesis motivation, objective, description, limitations and structure.
- Chapter 2: This chapter focuses on the background theory and the related work to the thesis in terms of satellites, technology and methods for hyperspectral anomaly detection. This includes a description of hyperspectral imaging, a detailed description of hyperspectral anomaly detection/detectors, and some preprocessing techniques. This chapter will also describe the Robust Graph Autoencoder in detail.
- Chapter 3: In this chapter, a presentation of the proposed HADs/enhancements to the RGAE are presented. How these changes or additions were implemented is explained as well. This

chapter also provides a detailed presentation of the Interpolated Autoencoder.

- Chapter 4: This chapter presents an overview of the datasets used to measure detection performance, results achieved during experimental testing, and a final comparison of the proposed HADs. For the datasets, a detailed analysis is shown. The experimental testing shows the different proposed HADs performances using various setups in terms of hyper-parameters. In the final comparison of the proposed HADs, the different methods will be compared using the optimal setups found during experimental testing. Here, all results will be discussed.
- Chapter 5: The Conclusion and Future Directions chapter summarises the thesis's main findings, contributions, and outcomes. It provides an overview of the research, highlights key points, and offers final remarks and conclusions. The final sections of the chapter will describe the future work, such as what can lead to further improvement in detection performance to the proposed HADs.

Chapter 2

Background and Related Work

This Chapter provides an insight into what Hyperspectral Imaging is and the structural properties of a Hyperspectral Image (HSI). Furthermore, some hyperspectral imaging satellites will be briefly described, such as the HYPSO satellite. Hyperspectral anomaly detection will also be explained with the inclusion of preprocessing techniques used to enhance an anomaly detector's performance. Several methods for hyperspectral anomaly detection will be explained in the latter parts of the chapter. Some of these include the use of machine learning and neural networks. The final section of the chapter will describe the Robust Graph Autoencoder in detail.

2.1 Hyperspectral Imaging

Conventional imaging methods capture information using three wavelengths corresponding to the red, green and blue color channels. Hyperspectral Imaging is the process of measuring the spectral information of an area or an object using a continuous range of wavelengths, also known as spectral bands. These wavelengths can cover many parts of the electromagnetic spectrum, such as the ultraviolet, visible and near-infrared regions. This technique can therefore capture several hundreds of spectral bands of the same object or area [1]. The sensors used for hyperspectral imaging sample hundreds of these electromagnetic waves from 0.4μm to 2.5μm. Spectral channels can be separated to as little as 10nm [1, 11]. Each measurement for a specific wavelength can create an image of the measured object or area. Combining each image created for every wavelength captured creates a Hyperspectral Image (HSI). The HSI is composed of F spectral bands, with a spatial height H and width W as shown in Figure 2.1. The total number of pixels within a HSI is $N = H \times W$. A pixel $\mathbf{x}_n$ within a HSI will therefore contain not only one value, but a vector of F values, each corresponding to a given spectral band. Here, n denotes the number of the pixel [1, 12].

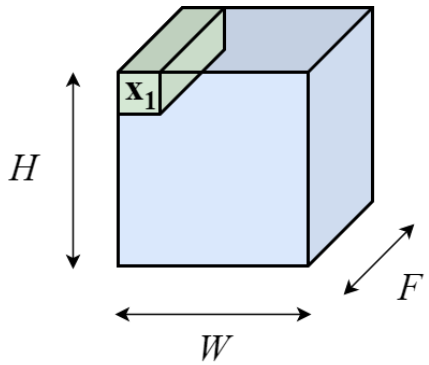


Figure 2.1: Representation of a hyperspectral image.

Combining all the spectral bands within a HSI can be used to determine what objects are located in the image based on each pixel's spectral signature [13]. The spectral signature of

a pixel refers to how a specific material responds to a particular wavelength in terms of emission/reflectance. An example of the spectral signature of a pixel $\mathbf{x}_1$ is shown in Figure 2.2. Here, the x-axis represents the wavelength/spectral band captured, and the y-axis represents the emission intensity. Depending on the texture and molecular composition of the object within the pixel, the spectral signature will vary [1]. This enables these images to be used in several target detection applications within various areas and fields such as agriculture and medicine[13, 14]. Using Hyperspectral Images enables numerous remote sensing applications, such as environmental monitoring and military target detection [1, 15]. Remote sensing refers to acquiring data without physical contact with the object in question [16]. Since Hyperspectral Images can be taken from satellites, it is possible to observe, detect and locate objects or environmental changes from orbit [1].

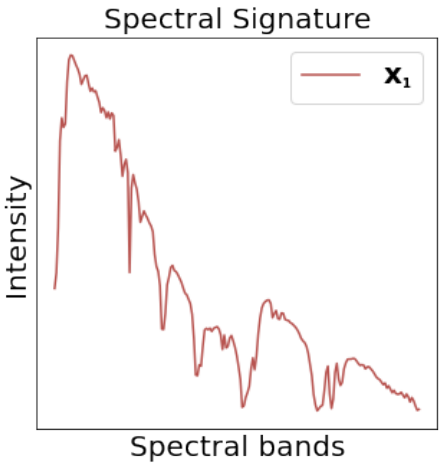


Figure 2.2: A pixels spectral signature.

2.1.1 Satellites and Technology

Multiple satellites aim to observe different properties remotely utilizing hyperspectral imagery. The HYPSO, the PRISMA and EnMAP are examples of such satellites utilized to monitor the environment of the Earth [4, 17, 18]. Systems such as the AVIRIS are also used for remote sensing [19]. The goal of the HYPerspectral Smallsat for OCEAN Observation mission, known as HYPSO, is to monitor the ocean for harmful algae blooms along the coast of Norway. If this is detected, it will send a notification to its users. The satellite is controlled by the Norwegian University of Science and Technology (NTNU). Regarding specifications, its camera can capture hyperspectral images of up to 1936 $\times$ 1194 pixels with at least 215 spectral bands. The range of the electromagnetic spectra used to capture the images is 220-967nm [4]. The HYPSO satellite orbits the Earth at an altitude of 500km and obtains a Ground Sampling Distance (GSD) of about 100m [4]. Ultimately, the research conducted in this thesis is intended to be valuable for the HYPSO satellite.

PRISMA, also known as PRecursore IperSpettrale della Missione Applicativa, is a satellite maintained by Italian Space Agency. Its purpose is to observe the spectral signatures of different materials [17]. Its exact electro-optical equipment allows it to analyze different materials' spectral fingerprints from a range of up to 615km. This equipment comprises an imaging spectrometer, Short-Wave InfraRed products, and a panchromatic camera, offering a comprehensive imaging solution with a GSD varying from 30m to 5m. The PRISMA satellite can take Hyperspectral Images containing up to 237 spectral bands with a spectral coverage of 0.4μm-2.5μm, providing a highly detailed image [1]. The satellite's launch date was in March 2019, and it remains in service to this day [17].

The German Space Agency maintains the Environmental Mapping Analysis Program satellite (EnMAP). Its purpose is to characterize the environment of the Earth on a global scale. With a spectral range from 0.42μm-2.45μm [1], the satellite can observe the environment's geochemical, biochemical and biophysical properties. This can then provide information on terrestrial and aquatic ecosystem evolution [18]. The satellite orbits with an altitude of 653km, slightly above that of the HYPSO and PRISMA satellites, whilst still achieving GSD of 30m.

AVIRIS, short for Airborne Visible/Infrared Imaging Spectrometer, is an optical instrument for remote sensing applications. The AVIRIS can capture Hyperspectral Images using spectral bands with wavelengths ranging from 0.4-2.5μm [19]. This imaging spectrometer has been used to capture all of the Airport-Beach-Urban (ABU) datasets except for the abu-beach-4 dataset [20]. More information about these datasets can be found in Chapter 4 as they will be utilized while testing the thesis' proposed Hyperspectral Anomaly Detectors.

As many of the properties these satellites aim to monitor deviate from their surroundings, such as harmful algae, oil spills and man-made objects such as ships, there are needs for precise Hyperspectral Anomaly Detectors (HAD). With HADs, monitoring changes in environment, resources, agriculture and targets such as vehicles autonomously [2, 3] is possible.

2.2 Hyperspectral Anomaly Detection

As mentioned in the previous section (Section 2.1.1), there is a need for hyperspectral anomaly detection to detect various anomalous objects and events. An explanation of these topics will be presented in this section to understand hyperspectral anomalies and the detection of these occurrences. Section 2.2.1 will introduce the concept of hyperspectral anomalies and Section 2.2.2 will describe the functionality of a general Hyperspectral Anomaly Detector.

2.2.1 Anomalies in Hyperspectral Images

A hyperspectral anomaly is classified as an object within a HSI whose spectral signature deviates significantly from its surrounding environment. The occurrences of anomalies typically have a lower probability than their surroundings, often referred to as the background, and are usually small. It must also be noted that there is no general spectral signature for anomalies, as an object or property is only determined to be anomalous relative to its surroundings [5, 3]. Figure 2.3 shows an example of how an anomalous pixel $\mathbf{x}_a$, marked with blue, deviates from the background pixel $\mathbf{x}_a$ marked with red.

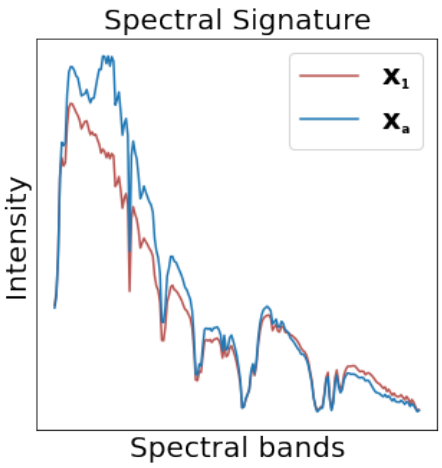


Figure 2.3: This figure highlights the difference in spectral signature of the anomalous pixel $\mathbf{x}_a$ in comparison to the background pixel $\mathbf{x}_1$.

2.2.2 Hyperspectral Anomaly Detector (HAD)

To detect a hyperspectral anomaly, there is a need for a HAD. These systems can detect and locate entities and events such as man-made objects, hazardous materials and environmental phenomenons where these targets deviate from their surroundings [2]. The input of a HAD is the HSI in question, and the output is a detection map that showcases the locations of the anomalies [5]. An illustration of the framework of such a HAD is shown in Figure 2.4. Before the Hyperspectral Image is sent into the detector itself, it is some cases exposed to preprocessing. This can be either

the use of a filter or methods such as the Principal Component Analysis (PCA) [2]. Figure 2.5 showcases the framework of a HAD with added preprocessing.

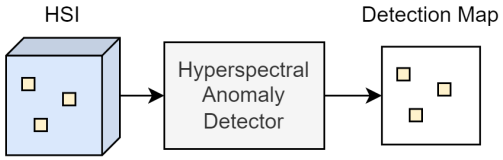


Figure 2.4: A block diagram showcasing the detection of anomalies using a hyperspectral anomaly detector.

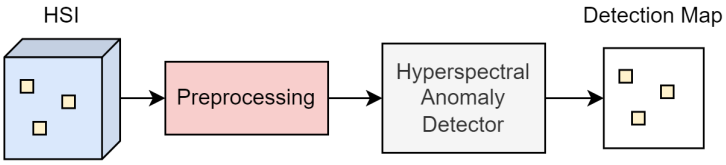
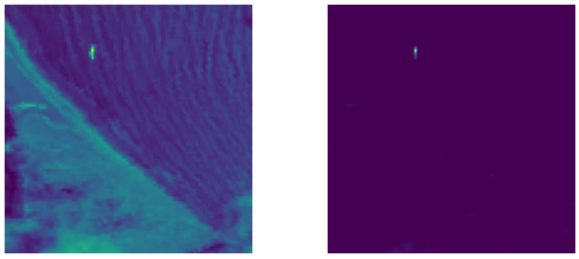


Figure 2.5: A block diagram showcasing the detection of anomalies using a hyperspectral anomaly detector with the addition of preprocessing.

A detection map, sometimes referred to as anomaly map, visualises the located anomalies separated from the background. Typically, the anomalies are intended to have a higher intensity than the background in the detection map, making it easier to locate them [2]. Figure 2.6 illustrates the difference between the input Hyperspectral Image and the output of the Hyperspectral Anomaly Detector, being the detection map. From the figure, it is much easier to locate the anomaly. In some Hyperspectral Images where much noise is present in addition to the anomalies, the noise can also be present in the detection map, making it harder to identify the anomalous pixels.



(a) Representation of the input HSI.

(b) Representation of the detection map of the anomaly.

Figure 2.6: Visualizations of the input and output of a HAD. The HSI used is the abu-beach-3 datset.

The two main performance metrics used to measure the performance of a HAD are the Area Under Curve (AUC) score and time score. The time score is measured from timing the HAD from input to output [6, 21]. An anomaly detector's ability to detect an abnormality is determined by its AUC score derived from the Receiver Operating Characteristic (ROC) curve. This curve tells us True Positive Rate (TPR) rate versus the False Positive Rate (FPR) [22]. The TPR is the rate of anomalous pixels that have been classified as anomalous. This measure is often known as sensitivity. Mathematically, the TPR is expressed as

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.1}$$

where TP (True Positive) is the number of correctly classified anomalous pixels, and FN (False Negative) is the number of misclassified anomalous pixels [22]. The FPR is the rate of pixels that

are not anomalous that have been classified as anomalous. This is known as specificity. The False Positive Rate is expressed as

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.2}$$

where FP (False Positive) is the number of non-anomalous pixels that have been misclassified, and TN (True Negative) is the number of correctly classified non-anomalous pixels.

To obtain these rates, the detection map is compared with the ground truth of the given dataset, which holds information about the location of each anomaly. A threshold is used to convert the image to a binary image. This threshold determines when a pixel will be determined as anomalous or background. The threshold is then varied to obtain a set of binary images. A TPR and a FPR are calculated for each binary image. It is then possible to plot the TPR and the FPR for each threshold resulting in the ROC curve, like the one illustrated in Figure 2.7. Calculating the area under this curve gives us an AUC score [23].

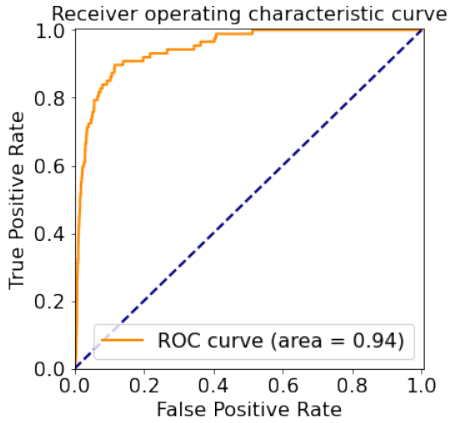


Figure 2.7: An example of an ROC curve for a HAD.

2.3 Preprocessing and Image Segmentation

This section explains different preprocessing techniques that can be performed for hyperspectral anomaly detection. Preprocessing can be a vital part of anomaly detection for many reasons. As hyperspectral images often contain large amounts of data, reducing the size of an HSI before analyzing it for anomalies can decrease the run time for a given HAD. Methods such as Principal Component Analysis (PCA) and Kernel-PCA (KPCA) can be utilized for such problems [7]. Other methods, such as image segmentation, can also benefit some applications. An example of such a technique is the Simple Iterative Clustering (SLIC) method [24]. An important note on image segmentation is that the term "pixel" is used to describe a singular numerical value in an image of two dimensions. The term "pixel" used when describing image segmentation techniques must not be confused with the "pixel" term in the Hyperspectral Images containing several numerical values.

2.3.1 Data Matrix

When analyzing a given HSI, it can be beneficial to transform the image into two dimensions since it enables easier manipulation. The transformed HSI, also known as the data matrix $\mathbf{D}$, will have the size $F \times N$ where each row represents a pixel $\mathbf{x}_n$ [7]. Here, F denotes the number of spectral bands, and N denotes the number of pixels. Figure 2.8 illustrates the data matrix.

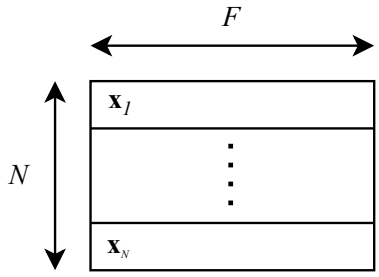


Figure 2.8: Properties of a data matrix **D**.

2.3.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used method for dimensionality reduction [25]. The principal behind PCA is to extract the essential characteristics of a data set based on the correlation within the given data. After performing PCA, the result should be that the input data has a lower dimensionality, but with as much of the variation present in the original data. A note on PCA is that the principal components are ranked from most to least essential components [26]. This means that the first dimensions of the lower dimensional output data after performing PCA better represent the original data. In a hyperspectral imaging case, the number of spectral bands can be reduced based on the higher correlation observed among certain bands. This can remove the number of redundant spectral bands [26]. A zero mean image $\mathbf{I} = [\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_n,]$ is generated based on the data matrix **D**, where n is the numbered pixel among N number of pixels [7]. $\mathbf{I}_n$ can be expressed as

$$\mathbf{I}_n = \mathbf{x}_n - \mathbf{M} = [I_{n1}, I_{n2}, ..., I_{nF}]^T. \tag{2.3}$$

where **M** is the mean image vector [7] calculated as

$$\mathbf{M} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n. \tag{2.4}$$

The covariance matrix is then calculated based of the zero mean image as

$$\mathbf{C} = \frac{1}{N}\mathbf{I}\mathbf{I}^T. \tag{2.5}$$

From the covariance matrix **C**, the eigenvalues $\mathbf{E}_v$ and eigenvectors $\mathbf{V}_v$ [7] are calculated as

$$\mathbf{C} = \mathbf{V}_v\mathbf{E}_v\mathbf{V}_v^T. \tag{2.6}$$

Using q number of eigenvectors, a $F \times q$ matrix is formed [7]. The value of q must be less than or equal to F, but it is often much smaller. The aim is to obtain a 2D image with the most effective features. To achieve this, the eigenvalues are arranged in descending order so that the top and most significant principal components can be easily accessed [7]. Finally, the projection matrix **Y** is computed like

$$\mathbf{Y} = \mathbf{w}^T \cdot \mathbf{I}. \tag{2.7}$$

A representation of a two-dimensional set of data with its principal components is illustrated in Figure 2.9. Here, u_1 contains the most information about the original data.

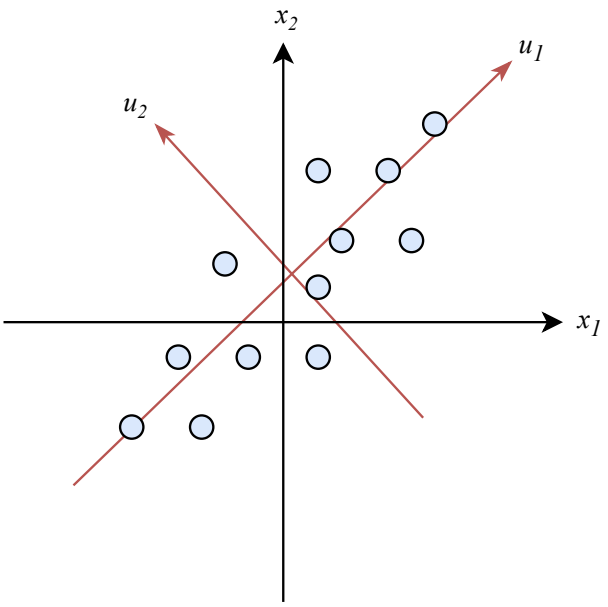


Figure 2.9: Two-dimensional set of data with its principal components u_1 and u_2 [27].

2.3.3 Kernel-Principal Component Analysis (KPCA)

KPCA operates on most of the same principals as the PCA, but rather than doing it linearly, the KPCA takes the data to a higher dimension using a kernel. This method has been observed to extract features better than that of the PCA, according to a study by L. J. Cao et al [28]. Each pixel $\mathbf{x}_n$ is taken into a higher feature space by the non-linear function $\phi(\mathbf{x}_n)$ [26]. When this is performed, the transformed data matrix $\mathbf{D}$ can be a subject of standard PCA. The kernel function, expressed as

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j). \tag{2.8}$$

is used to create the kernel matrix $\mathbf{K}$ so that $\mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ [7]. The vector $\mathbf{a}_k = [a_{k1}, a_{k2}, ..., a_{kN}]$ vector is then computed so that

$$\mathbf{K}\mathbf{a}_k = \mathbf{E}_k F \mathbf{a}_k \tag{2.9}$$

where F is the amount of spectral bands. For the cases where there is no possibility to compute the zero mean image, the approach is to calculate the Gram matrix $\tilde{\mathbf{K}}$. This will then replace the kernel matrix $\mathbf{K}$ [7], expressed as

$$\tilde{\mathbf{K}} = \mathbf{K} - 1_N\mathbf{K} - \mathbf{K}1_N + 1_N\mathbf{K}1_N \tag{2.10}$$

where 1_N are matrices of size $N \times N$ containing the value $1/N$ in each cell. The following procedure is then to calculate the kernel principal component $\mathbf{y}_k(\mathbf{x})$ as

$$\mathbf{y}_k(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{v}_k = \sum_{i=1}^{N} a_{ki}\kappa(\mathbf{x}, \mathbf{x}_i). \tag{2.11}$$

Several kernel functions can be utilized to calculate the KPCA of an HSI. Some of these are Gaussian [26], Laplacian, Sigmoid [7] and Polynomial [29] functions, each expressed in the equations (2.12), (2.13), (2.14) and (2.15) respectively. Here σ is a parameter that controls the width of the kernel functions. The c parameter is a constant that determines the shape and position of the curve of the kernel functions. For the polynomial kernel, the a parameter determines the degree of the polynomial function.

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||}{2\sigma^2}\right) \tag{2.12}$$

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||}{\sigma}\right) \tag{2.13}$$

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\frac{\mathbf{x}_i \mathbf{x}_j^T}{2\sigma^2} + c\right), \quad c \geq 0 \tag{2.14}$$

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^a, \quad c \geq 0, \quad a \geq 2 \tag{2.15}$$

Multiple kernels can be utilized to reduce dimensionality whilst better characterizing the data [30]. This has been observed to improve upon learning ability [31]. For multiple kernels, the final kernel function will be a linear combination of multiple kernels, defined as

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{M} \beta_m \kappa_m(\mathbf{x}_i, \mathbf{x}_j), \quad \beta_m \geq 0 \tag{2.16}$$

where M is the number of kernels and β_m is the weighting of each kernel [30]. The final kernel matrix will therefore be calculated as

$$\mathbf{K} = \sum_{m=1}^{M} \beta_m \mathbf{K}_m, \quad \beta_m \geq 0. \tag{2.17}$$

2.3.4 Robust-PCA (RPCA)

Robust-PCA (RPCA) is based on the principal that a data matrix $\mathbf{M}$ can be modeled based on its low-rank matrix $\mathbf{L}_0$ and its sparse matrix $\mathbf{S}_0$ [32], expressed as

$$\mathbf{M} = \mathbf{L}_0 + \mathbf{S}_0. \tag{2.18}$$

Here, the low-rank matrix describes the data matrix's underlying structure, and the sparse matrix represents the noise or outliers, illustrated in Figure 2.10. In the illustration, the data matrix $\mathbf{M}$ contains four outliers seen in the sparse representation $\mathbf{S}_0$.

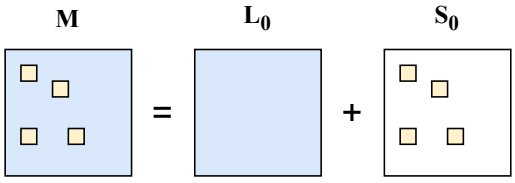


Figure 2.10: An illustration how a data matrix $\mathbf{M}$ can be expressed as a sum of its low-rank matrix $\mathbf{L}_0$ and its sparse matrix $\mathbf{S}_0$.

To model $\mathbf{M}$ as a sum of the to matrices $\mathbf{L}_0$ and $\mathbf{S}_0$, it is necessary to solve the following problem

$$\min_{\mathbf{L}_0, \mathbf{S}_0} ||\mathbf{L}_0||_* + \lambda ||\mathbf{S}_0||_1 \quad \text{subject to} \quad \mathbf{L}_0 + \mathbf{S}_0 = \mathbf{M} \tag{2.19}$$

where $||\mathbf{L}_0||_*$ denotes the nuclear norm expressed as

$$||\mathbf{L}_0||_* = \sum_{i=1}^{\min\{m,n\}} \sigma_i. \tag{2.20}$$

m and n represents the number of columns and rows in the given matrix and σ_i represents the singular values of matrix $\mathbf{L}_0$ [33]. $||\mathbf{S}_0||_1$ denotes the ℓ_1 norm of $\mathbf{S}_0$, expressed as

$$||\mathbf{S}_0||_1 = \sum_{i=1}^{m}\sum_{j=1}^{n}|\mathbf{S}_{0,(i,j)}|. \tag{2.21}$$

The expression in Equation 2.19 is known as Principal Component Pursuit (PCP). It can be solved using the Augmented Lagrange Multiplier (ALM) algorithm or simpler procedures such as the Alternating Directions Method (ADM) [33].

2.3.5 Simple Iterative Clustering (SLIC)

Simple Iterative Clustering is a method of image segmentation which relies on clustering an image into K numbers of superpixels using the well-known K-means clustering approach. A superpixel is a group or cluster of pixels with similar properties in a defined area. The first step in the process is to define a K number of superpixels $\mathcal{P} = \{\mathbf{P}_k\}_{k=1}^{K}$. These are initialized with an equal spacing in the entire image so that each superpixel has roughly the same size [24]. The distance between each superpixel is expressed as

$$d_{interval} = \sqrt{N/K} \tag{2.22}$$

where N is the total number of pixels within the image. The superpixels position is chosen by a $3 \times$ three neighbourhood using the lowest gradient pixel. This is done to avoid placing a superpixel on an edge or a noisy pixel. The initialization process is followed up by assigning each pixel in the image to its corresponding superpixel. This is determined based on the lowest distance from a given pixel to a given superpixel's $\mathbf{P}_k$ center. To calculate this distance, the spatial and color distances are computed and combined. The spatial distance is calculated using the Euclidean distance between the superpixel center and the pixel in question. To find the color distance between pixel x_i in the superpixel center and pixel x_j, the distance d_c is expressed as

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \tag{2.23}$$

is calculated. Here, the color is represented in the CIELAB color space [24], being l for the brightness, a for the color along the red-green axis and b for the color along the blue-yellow axis. Each superpixel has an expected size of about $d_{interval} \times d_{interval}$. However, the maximum size of each superpixel is $2d_{interval} \times 2d_{interval}$ with the superpixel being located in the center. The search for similar pixels is therefore only being done in this area. When all pixels have been assigned to a superpixel, all centroids (centers of superpixels) will reposition themselves using the mean pixel location for the pixels in the given superpixel $\mathbf{P}_k$. After this is done, all pixels will search for their corresponding superpixel again, which is a process that will continue until the positions of each superpixel converge [24].

2.4 Existing Methods for HAD

Hyperspectral anomaly detection can be separated into several categories. The fundamental forms of anomaly detection within this field are the statistical methods [2]. Other methods, such as machine learning, can also be used to target hyperspectral anomalies. The machine learning category can be segmented into traditional machine learning, also known as distance-based and neural network-based methods [2].

Several approaches to hyperspectral anomaly detection exist in terms of statistical methods. Several of these HADs rely on the Reed-Xiaoli (RX) method, such as Local-RX (LRX) and Global-RX (GRX). The first occurrence of the RX-based HAD was in the early 90s [2]. RX-based methods rely on modeling the background of an image as a Gaussian distribution [34]. Using the distance from a pixel to the background will then reveal whether the pixel is anomalous. This algorithm can be divided into two versions, being the global and local method later described in Section 2.5.

In terms of using traditional machine learning, several anomaly detection approaches exist. One of these methods is clustering, utilized in the density-peak clustering approach [35] and in the default Cluster-based Anomaly Detector (CBAD) [36]. One of the strengths of this approach to hyperspectral anomaly detection is the fact that the clustering method is unsupervised [2]. Unsupervised methods can learn patterns and structures from a set of unlabeled data without

any human interference. A different method is the support vector machine, which separates the anomalies from the background using a hyperplane [37]. This method has shown good generalization performance for various classification tasks, which makes it suitable for anomaly detection [2]. The Support Vector Data Description (SVDD) is a form of Support Vector Machine (SVM) which has proven as an efficient and effective anomaly detection method [38]. The traditional machine learning methods for HAD is explained in more detail in Section 2.6.

Utilizing deep neural networks for anomaly detection has proven to be a very powerful technique in recent years. Examples of such models are the default Autoencoder (AE), the Convolutional Neural Network (CNN), and the Generative Adversarial Network (GAN) [5]. The advantage of the AE and the GAN is the fact that they are unsupervised methods, like the traditional clustering approach. More advanced methods, such as the Robust Graph Autoencoder (RGAE), have recently been developed [6]. These HADs are based on simpler neural networks like the AE with added modifications that provide either better feature extraction, preprocessing, a regularization term, or other properties that enhance the performance of the given model. Other methods, such as utilizing the CNN architecture, have also been utilized for state-of-the-art HADs. The Denoising CNN regularized anomaly detection (DeCNN-AD) method, laid forth by X. Fu et al. [39], resulted in a high-performing anomaly detector, in terms of detection accuracy, beating several other Hyperspectral Anomaly Detectors utilizing deep neural networks [5]. A more detailed description of how neural networks can be used for hyperspectral anomaly detection can be found in Section 2.7.

2.5 Statistics Based Anomaly Detection

The Global-RX and the Local-RX methods for hyperspectral anomaly detection will be presented in this section. Both of the two HADs are based on the RX algorithm [2]. This section is only meant to give an insight into these methods, which is why they are not described in full detail.

2.5.1 Global Reed-Xiaoli

One Reed-Xiaoli-based method, previously mentioned, is the GRX detector. A big assumption with this approach is that the background is homogeneous. That enables the background to be modeled as a Gaussian distribution [34]. With the background modeled, it is possible to calculate the distance from each pixel to the background using the following function

$$D_{GRX}(\mathbf{x}_n) = (\mathbf{x}_n - \hat{\mu}_b)^T \hat{\Sigma}_b^{-1} (\mathbf{x}_n - \hat{\mu}_b). \tag{2.24}$$

The distance $D_{GRX}(\mathbf{x}_n)$ of pixel $\mathbf{x}_n$ is calculated based on the mean $\hat{\mu}_b$ and covariance $\hat{\Sigma}_b$. Determining whether the given pixel is anomalous is by comparing the calculated distance to a threshold [34].

2.5.2 Local Reed-Xiaoli

The Local-RX detector is a version of the GRX detector with a focus on local areas in the given HSI. In this approach, the background model has been swapped with a local normal model. To detect the anomalies, a double concentric window slides over each pixel. The largest window contains the local background, and the inner window, also known as the guard band, is assumed to be the size of the target in the image. This is illustrated in Figure 2.11. To calculate the distance, the distance function $D_{LRX}(\mathbf{x}_n)$ is utilized and expressed as

$$D_{LRX}(\mathbf{x}_n) = (\mathbf{x}_n - \hat{\mu}_{local})^T \hat{\Sigma}_{local}^{-1} (\mathbf{x}_n - \hat{\mu}_{local}) \tag{2.25}$$

where $\hat{\Sigma}_{local}$ is the covariance matrix of the local background and $\hat{\mu}_{local}$ is the mean vector.

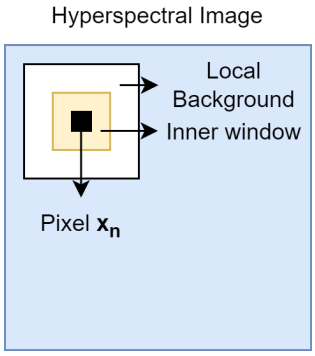


Figure 2.11: The double concentric window of the LRX detector [34].

2.6 Traditional Machine Learning for Anomaly Detection

This section will give an insight into two traditional machine learning methods of hyperspectral anomaly detection. Traditional machine learning is a branch of artificial intelligence used in prediction, regression, or classification problems [40]. This does not include the use of neural networks.

2.6.1 Support Vector Machine for HAD

The SVM is a well-known machine learning algorithm that works especially well in classification problems. It has proven to be quite a sufficient algorithm within the field of anomaly detection as well [37, 38]. According to A. Banerjee et al., their support vector method reduced false detections compared to the more traditional HADs, such as the RX-based methods.

As this algorithm is not a focus point of the thesis, it will not be described in full detail. The Support Vector Machine works by separating two or more classes in a given feature space by using a hyperplane [41]. For a linear SVM, the following hyperplane

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.26}$$

will separate two classes, where $\mathbf{w}$ is the normal vector and b is a bias. $\mathbf{x}$ is a point in the feature space. Using nonlinear hyperplanes to separate two or more classes is also possible. Given that the data is separable, the SVM can classify the data point.

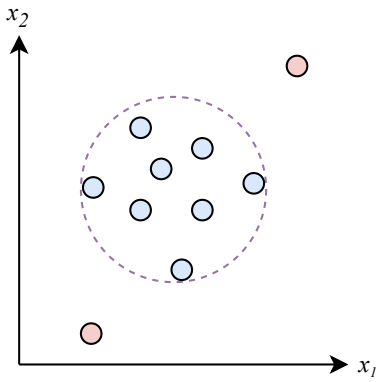


Figure 2.12: This is how anomalies are detected using an SVM. The anomalies are represented by the red dots, the background by the blue dots and the hyperplance as the striped purple circle. x_1 and x_2 represents two features in the feature space [2].

The hyperplane, for a linear classifier, is generated by solving the optimization problem:

$$\begin{aligned} &\text{minimize}_{\mathbf{w},b}\ \frac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to}\quad y_i{\cdot}((\mathbf{w}\cdot\mathbf{x}_i)+b)\geq 1, \quad i=1,...,m \end{aligned} \tag{2.27}$$

where y_i is the label of data point $\mathbf{x}_i$ and m is the number of data points. This can be solved through its Lagrangian dual [41]. Figure 2.12 illustrates how an SVM can classify anomalies using a hyperplane that ideally separates all background data from anomaly data.

2.6.2 Clustering for HAD

Clustering is within the branch of traditional machine learning and has been used for anomaly detection in various instances [42, 43]. One method of clustering is the K-means clustering algorithm. This approach relies on clustering data into K numbers of clusters $\mathbf{C}_c = \{\mathbf{C}_1, ..., \mathbf{C}_K\}$. A pixel is determined to be anomalous if the distance from its location to the center of its corresponding cluster is above a certain threshold. This is illustrated in Figure 2.13.

The K-means clustering algorithm works by first initializing K number of cluster centers $\boldsymbol{\mu}_c = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K\}$ in the data space, often referred to as centroids [44]. Each center can be randomly chosen in the data space. The next step is to assign all data points to their closest centroid. In a hyperspectral anomaly detection case, these points would be pixels in the HSI. To determine what centroid is the closest to a given data point $\mathbf{x}_n$, the Euclidian distance between the pixel and each centroid is calculated. The data point is then assigned to cluster $\boldsymbol{\mu}_k$, where k is calculated as

$$k = \underset{j}{\text{argmin}}(||\mathbf{x}_i - \boldsymbol{\mu}_j||^2). \tag{2.28}$$

Each cluster center is reassigned after each data point has been assigned to a cluster. The new location of each cluster is based on the mean position of all the data points assigned to the corresponding cluster [44]. This can be expressed as

$$\boldsymbol{\mu}_j = \frac{1}{N_j}\sum_{n=1}^{N_j}\mathbf{x}_n \tag{2.29}$$

where, in this case, $\mathbf{x}_n$ denotes data points in cluster $\mathbf{C}_j$, where N_j is the total number of data points in the cluster. After relocating all cluster centers, each data point will be assigned to its closest cluster based on the Euclidean distance. This entire process is repeated several times or until the cluster centers converge [44].

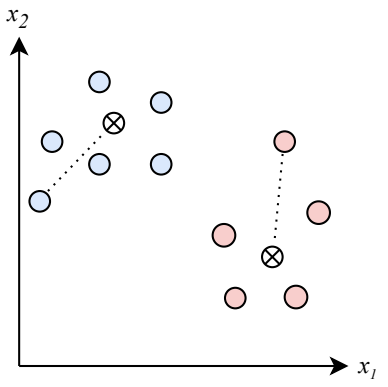


Figure 2.13: This is how anomalies look in a clustered set of data. Here, the different colors represent which data points belong to which cluster. The cross marked dots represent each centroid and the two data points which are located the furthest away from their corresponding centroid are registered as anomalous.

2.7 Deep Learning for HAD

Deep learning has proven to be a powerful tool for hyperspectral anomaly detection due to the remarkable ability of non-linear feature extraction [5]. The Autoencoder, generative, and convolutional neural networks are examples of deep learning networks used for such tasks. In Section 2.7.1, a general explanation of neural networks is presented, followed up by an explanation of the Autoencoder in Section 2.7.2. These sections are then followed up by a more specific instance of the Autoencoder for HAD, being the Robust Graph Autoencoder.

2.7.1 Neural Networks

Neural networks are models composed of several layers of neurons that are utilized to solve regression problems, classification problems, or similar issues based on input data. Standard neural networks comprise several layers: the input, output, and hidden layers(s) connecting the two. Each layer is built up as a set of nodes that are in some way connected to the previous and subsequent layers. In each node, some mathematical operation is performed, such as multiplying the input with a weight and adding a bias. If each node in a layer is connected to all of the nodes in a previous layer, the layers are fully connected [45]. An illustration of a neural network can be seen in Figure 2.14.

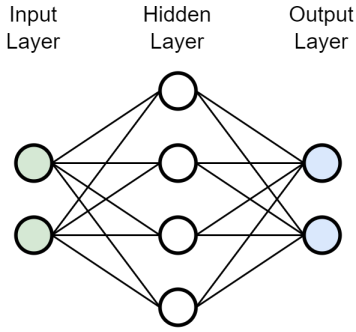


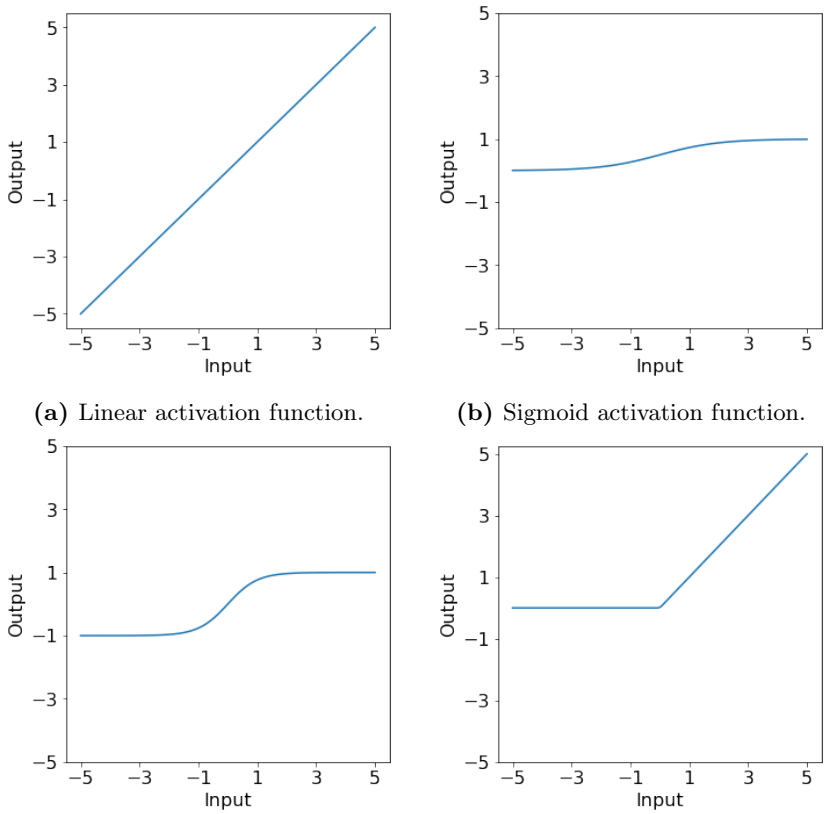
Figure 2.14: A fully connected neural network composed of an input layer, an output layer and a singular hidden layer.

Many neural networks, including all networks described in this thesis, are feed-forward networks. In a feed-forward neural network, the data flows only from the input to the output in one direction. These networks are also referred to as multi-layer perceptrons. After passing data through the network, the weights and biases in all of the layers are adjusted through what is called backpropagation. Through several iterations of this process, the network can learn and adjust to the input data so that the output is satisfying [45]. A node in a given layer m will receive the output of the previous layer $m-1$ as input. Here a mathematical operation is performed, expressed by

$$\mathbf{h}^{(m)} = y\left(\mathbf{W}^{(m)}\left(\mathbf{h}^{(m-1)}\right)^T + \mathbf{b}^{(m)}\right)^T, \quad m = 1, ..., M \tag{2.30}$$

where $\mathbf{h}^{(m)}$ is the output of the node and $\mathbf{h}^{(m-1)}$ is the input. $\mathbf{W}^{(m)}$ and $\mathbf{b}^{(m)}$ are the weight and bias. In a network setup such as shown in Figure 2.14, each connection between two nodes has its own weight, whereas the bias is applied equally for all inputs of a specific node. The function $y(\cdot)$ is called an activation function, which is usually applied before sending the output to the next layer in the network [6]. Commonly used activation functions are the Linear, Sigmoid, Tanh, and ReLU functions [46]. These are illustrated in Figure 2.15.

The adjustment of weights and biases in a feed-forward neural network is done through backpropagation. The goal is to minimize the network's objective function $J(\Theta)$, which is often the same as the loss function. Θ, in this case, denotes the weights and biases in the network $\Theta = \{\mathbf{W}_t^{(m)}, \mathbf{b}_t^{(m)}\}$. A loss function gives an estimate of the error of a given network. The loss is calculated and sent back through the network to adjust the weights and biases. Based on the loss, the weights and biases are adjusted so that the objective function will minimize. This is done



(a) Linear activation function.

(b) Sigmoid activation function.

(c) tanh activation function.

(d) ReLU activation function.

Figure 2.15: An illustration of different activation functions.

over several iterations, also known as epochs [45]. Examples of loss functions used to evaluate the quality of a neural network are the Mean Square Error (MSE) loss, reconstruction loss, binary cross-entropy loss, and Kullback-Leibler divergence [47, 48, 49]. To avoid over- or under-fitting, an early stopping criterion is often utilized. This method stops the training when a criterion, such as the loss, converges. Over-fitting occurs when a network is trained excessively on specific data, causing it to perform well only on that particular training data. Under-fitting results from the network being unable to capture the variability in the training data, in other words, the opposite of over-fitting [50].

Gradient descent is a popular method used during the training of a neural network. This method relies on using the gradient of the objective function $J(\Theta)$ to adjust the weights and biases during backpropagation [51]. Batch Gradient Descent (BGD) is a common instance of this method, utilizing the entire training data to optimize the network parameters (weights and biases). The parameters $\Theta = \{\mathbf{W}_t^{(m)}, \mathbf{b}_t^{(m)}\}$ are updated as

$$\mathbf{W}_t^{(m)} = \mathbf{W}_{t-1}^{(m)} - \mu \frac{\partial J^{(m)}(\Theta)}{\partial \mathbf{W}_t^{(m)}} \tag{2.31a}$$

$$\mathbf{b}_t^{(m)} = \mathbf{b}_{t-1}^{(m)} - \mu \frac{\partial J^{(m)}(\Theta)}{\partial \mathbf{b}_t^{(m)}} \tag{2.31b}$$

where μ is the learning rate of the neural network, and t denotes the current iteration of the training process. A common issue with the BGD is that it is slow due to utilizing the entire training dataset during each epoch [51]. A solution to this problem is to utilize Stochastic Gradient Descent (SGD). Using this method, only one sample of the training dataset is used during each training iteration. This method is quicker and superior in terms of redundancy as BGD uses a massive amount of data. A negative side with SGD is that it suffers from fluctuation due to the frequent calculation of gradients [51].

To fix both the slowness of BGD and the fluctuation using SGD, an approach is to use Mini Batch Gradient Descent (MBGD). With this method, a small batch of the training data is used to train the network during each epoch. This results in a training time lower than the BGD and fewer fluctuations than the SGD. To avoid using the same data over and over again, the batch of training samples are chosen at random by shuffling the training dataset [51].

To further improve a neural network's performance, a common approach is to utilize an optimizer. Some of these are the momentum, the Root Mean Square Propagation (RMSP), and the Adaptive Moment Estimation (ADAM) optimizers [51]. Momentum is a technique that focuses on accelerating the learning process of a neural network while dampening the oscillations when trying to locate the minima of the objective function [51]. The approach is to calculate the updated parameters as

$$\mathbf{W}_t^{(m)} = \mathbf{W}_{t-1}^{(m)} - \mu m_t^w \tag{2.32a}$$

$$\mathbf{b}_t^{(m)} = \mathbf{b}_{t-1}^{(m)} - \mu m_t^b \tag{2.32b}$$

where the momentum variables m_t^w and m_t^b are expressed as

$$m_t^w = \beta_1 m_{t-1}^w + (1-\beta_1)\frac{\partial J_t(\Theta)}{\partial \mathbf{W}_t^{(m)}} \tag{2.33a}$$

$$m_t^b = \beta_1 m_{t-1}^b + (1-\beta_1)\frac{\partial J_t(\Theta)}{\partial \mathbf{b}_t^{(m)}}. \tag{2.33b}$$

$\beta_1 \in [0,1]$ is a constant that controls the impact of the most recent momentum variables compared to the previous ones. If β_1 were to be set to 0, the network's learning would be the same as not using an optimizer [51]. A different optimizer that also aims to prevent oscillations when locating the minima of the objective function is the Root Mean Square Propagation optimizer. The updated weights and biases are given by

$$\mathbf{W}_t^{(m)} = \mathbf{W}_{t-1}^{(m)} - \frac{\mu}{\sqrt{v_t^w + \epsilon_r}}\frac{\partial J_t(\Theta)}{\partial \mathbf{W}_t^{(m)}} \tag{2.34a}$$

$$\mathbf{b}_t^{(m)} = \mathbf{b}_{t-1}^{(m)} - \frac{\mu}{\sqrt{v_t^b + \epsilon_r}}\frac{\partial J_t(\Theta)}{\partial \mathbf{b}_t^{(m)}}. \tag{2.34b}$$

ϵ_r is a small constant to prevent division by 0 and the variables v_t^w and v_t^b are the sum of the squared gradients for the weights and biases in the network [51]. These are given by

$$v_t^w = \beta_2 v_{t-1}^w + (1-\beta_2)\left(\frac{\partial J_t(\Theta)}{\partial \mathbf{W}_t^{(m)}}\right)^2 \tag{2.35a}$$

$$v_t^b = \beta_2 v_{t-1}^b + (1-\beta_2)\left(\frac{\partial J_t(\Theta)}{\partial \mathbf{b}_t^{(m)}}\right)^2 \tag{2.35b}$$

where $\beta_2 \in [0,1]$ is a constant, such as β_1, which controls the impact of the current gradient compared to the previous ones [51].

Using a combination of the momentum and RMSP optimizers will result in the Adaptive Moment Estimation (ADAM) optimizer. This optimizer aims to speed up the network's learning and reduce the oscillations when minimizing the objective function. The network parameters are updated as

$$\mathbf{W}_t^{(m)} = \mathbf{W}_{t-1}^{(m)} - \hat{m}_t^w\left(\frac{\mu}{\sqrt{\hat{v}_t^w + \epsilon}}\right) \tag{2.36a}$$

$$\mathbf{b}_t^{(m)} = \mathbf{b}_{t-1}^{(m)} - \hat{m}_t^b\left(\frac{\mu}{\sqrt{\hat{v}_t^b + \epsilon}}\right). \tag{2.36b}$$

The variables $\hat{m}_t^w$, $\hat{m}_t^b$, $\hat{v}_t^w$ and $\hat{v}_t^b$ are expressed as

$$\hat{m}_t^w = \frac{m_t^w}{1-\beta_1^t}, \quad \hat{v}_t^w = \frac{v_t^w}{1-\beta_2^t} \tag{2.37a}$$

$$\hat{m}_t^b = \frac{m_t^b}{1-\beta_1^t}, \quad \hat{v}_t^b = \frac{v_t^b}{1-\beta_2^t} \tag{2.37b}$$

where m_t^w and m_t^b are computed as shown in Equation 2.33 and the variables v_t^w and v_t^b are computed as shown in Equation 2.35 [51].

2.7.2 Autoencoder for HAD

Autoencoders are a type of neural network architecture specified in unsupervised learning. An Autoencoder comprises both an encoder and a decoder. The purpose of the neural network architecture is to encode a set of data $\mathbf{X}$ into a compressed and meaningful representation. This is then possible to decode using a decoder. The output of the encoder is called the latent-variable $\mathbf{Z}$ and the output of the decoder $\hat{\mathbf{X}}$ is a reconstructed version of the input data $\mathbf{X}$ [52]. Figure 2.16 shows an illustration of the process that occurs when sending data $\mathbf{X}$ through an Autoencoder. As with other neural networks, the Autoencoders are composed of several layers of artificial neurons. Each connection between two nodes has its corresponding weight, and every node has its corresponding bias. An activation function is also applied before sending the output of a neuron further through the network.

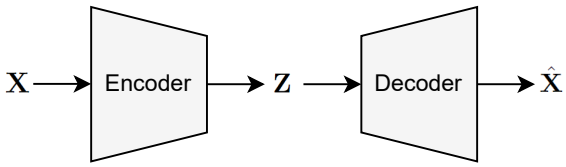


Figure 2.16: An overview of the encoding and decoding process in an autoencoder [52].

The encoded output $\mathbf{Z}$ can be expressed as a learned representation of the input data using the function $f_\theta(\cdot)$ [8]. This function is what transforms the input data to a lower-dimensional latent space and is expressed as

$$\mathbf{Z} = f_\theta(\mathbf{X}). \tag{2.38}$$

Decoding the latent-variable $\mathbf{Z}$ can be expressed as

$$\hat{\mathbf{X}} = g_\phi(\mathbf{Z}) \tag{2.39}$$

where $g_\phi(\cdot)$ is the function describing the decoders operations [8].

To measure the performance of an Autoencoder, a common approach is to look at the reconstruction loss. This quantity describes how close the output $\hat{\mathbf{X}}$ is to the input $\mathbf{X}$. A perfect Autoencoder, in most cases, has 0 reconstruction loss. The reconstruction loss, in most cases the Mean Square Error loss, is expressed as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} ||\hat{\mathbf{x}}_i - \mathbf{x}_i||^2 \tag{2.40}$$

where N is the number of samples that are to be reconstructed. The number of samples N could be the total number of pixels if one were to reconstruct an HSI.

Using Autoencoders for anomaly detection in hyperspectral images is utilizing the fact that the Autoencoders struggle with reconstructing anomalies compared to the image's background. This is due to the significant deviation in spectral signature. As a result, anomalies will ideally look just like the background in the reconstructed image. When subtracting the reconstructed image from the original image, the anomalous pixels will leave residuals in the detection map. These residuals are used to detect/locate the anomalies [6]. This process is illustrated in Figure 2.17. The mathematical expression for creating the detection map X_{det} would therefore be expressed as

$$\mathbf{X}_{det} = \mathbf{X} - \hat{\mathbf{X}} \tag{2.41}$$

where the matrix $\mathbf{X}_{det}$ can be either summarized or sorted based on the spectral intensities so that the final detection map X_{det} is in two dimensions [6].

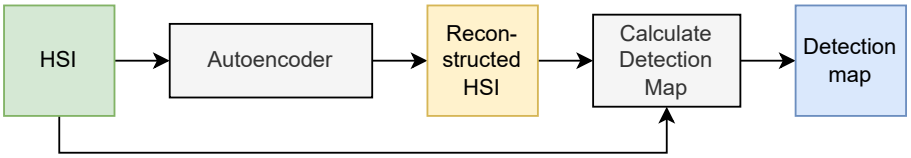


Figure 2.17: The hyperspectral anomaly detection process of an autoencoder.

2.7.3 Robust Graph Autoencoder

A new method for hyperspectral anomaly detection was proposed in 2022, called the Robust Graph Autoencoder This anomaly detection method utilizes an Autoencoder with the combination of a graph regularization term [6, 53]. As Autoencoders can lose all sense of spatial information during training, this can cause a decrease in detection performance [6]. Autoencoders also suffer from being able to reconstruct the input so well that they can reconstruct anomalies if there were anomalies present in the training data. This will then cause a faulty detection map as there would not be any residuals where the anomalous pixels are located [6]. A solution to these problems is therefore to use a graph regularization term along with the Autoencoder. This term's purpose is to ensure that similar-looking pixels in the input are represented similarly in the latent space when encoded by the Autoencoder [6].

The entire Robust Graph Autoencoders structure can be seen in Figure 2.18. It is composed of two main segments, being the Autoencoder and the Graph Term Creation, as well as the computation of the data matrix $\mathbf{D}$ (Explained in Section 2.3.1), the modified loss function and the computation of the detection map in the Detection Map block. For the detection map X_{det} computation, each pixel is individually chosen to contain the highest spectral intensity from the $\mathbf{X}_{det}$ matrix (Calculated as shown in Section 2.7.2). The Autoencoders structure is relatively simple, as it only comprises two sets of weights and biases. This is illustrated in Figure 2.19. From the figure, it is possible to see that the encoding process takes place from the first layer to the output of the second layer. For the decoder, the decoding process takes place from the middle layer's output to the last layer's output. The output of the first layer is just the regular input $\mathbf{X}$, which is the data matrix computed for the Hyperspectral Image. The output of the middle layer, however, is the encoded variable $\mathbf{Z}$. This variable is then decoded, outputting the reconstructed version of the input as $\hat{\mathbf{X}}$. It is also worth noting that this neural network utilizes the Adaptive Moment Estimation optimizer to boost the training process.

Regarding activation functions, both the encoder and the decoder use the Sigmoid function. The layer dimensionalities can be seen in Table 2.1. Here, F denotes the number of spectral bands and n_hid the number of neurons or hidden dimensions. Layer 1, 2, and 3 refer to the first, second, and third layers in Figure 2.19. Mathematical expressions for the encoding and decoding process can be expressed as

$$f_\theta(\mathbf{X}) = y_{sigmoid}\left(\mathbf{W}_{encoder}\mathbf{X} + \mathbf{b}_{encoder}\right) \tag{2.42}$$

and

$$g_\phi(\mathbf{X}) = y_{sigmoid}\left(\mathbf{W}_{decoder}\mathbf{X} + \mathbf{b}_{decoder}\right). \tag{2.43}$$

based on the network parameters.

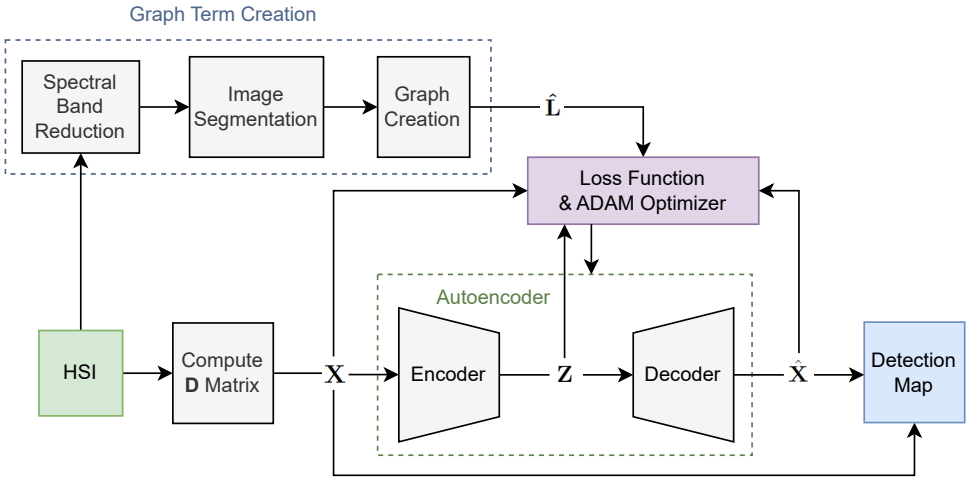


Figure 2.18: An overview of the RGAEs structure.

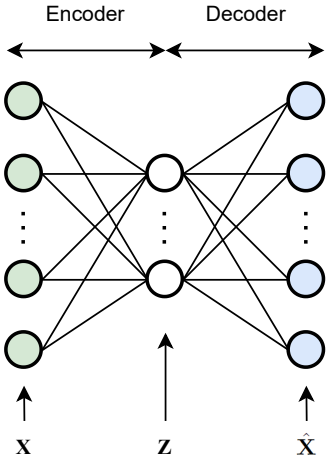


Figure 2.19: An overview of the layer setup of the RGAE [6].

Table 2.1: Layer setup for the RGAE.

Layer	Weight size	Bias size	Activation function	Number of parameters
Layer 1 (Input)	-	-	-	-
Layer 2	$n\_hid \times F$	n_hid	Sigmoid	$n\_hid(F+1)$
Layer 3	$F \times n\_hid$	F	Sigmoid	$n\_hid(F+1)$

The Graph Creation segment illustrated in Figure 2.18 comprises three main blocks, the Spectral Band Reduction, Image Segmentation, and Graph Creation Blocks. The first step to creating the graph is to compute the principal component of the HSI in the Spectral Band Reduction block. It is then possible to segment the first principal component of the HSI using SLIC (Explained in Section 2.3.5) in the Image Segmentation block. Following the segmentation process, the next step is to construct a graph term based on the superpixels $\mathcal{P} = \{\mathbf{P}_k\}_{k=1}^{K}$ created. The loss function will use this term along with the input variable $\mathbf{X}$, the latent variable $\mathbf{Z}$, and the output variable $\hat{\mathbf{X}}$ of the autoencoder. A description of the loss function will be presented later in this section. The first step of creating the graph term is to compute the adjacency matrix $\hat{\mathbf{W}}$. This matrix represents the similarity between pixels $\mathbf{x}_i$ and $\mathbf{x}_j$ in the Hyperspectral Image. To construct this matrix, each row within it is formed by

$$\hat{w}_{ij} = \begin{cases} \exp\left(-\frac{||\mathbf{x_i}-\mathbf{x_j}||_2^2}{\sigma^2}\right) & u = v \\ 0 & u \neq v \end{cases}, \qquad 1 \leq u \text{ and } v \leq K \tag{2.44}$$

where the two pixels $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the superpixels $\mathbf{P}_u$ and $\mathbf{P}_v$ respectively. In this setting, σ is a scalar parameter. The next step of the process is then to calculate the degree matrix $\hat{\mathbf{D}}$ [6], where each diagonal element d_{ii} is expressed as

$$d_{ii} = \sum_{j=1}^{N} \hat{w}_{ij} \tag{2.45}$$

where N is the number of pixels. The last step of the process is to compute the Laplacian matrix $\hat{\mathbf{L}}$, which will be used in the graph regularization term. To calculate the matrix, both the degree matrix $\hat{\mathbf{D}}$ and the adjacency matrix $\hat{\mathbf{W}}$ are needed [6]. It is the possible to calculate $\hat{\mathbf{L}}$ as

$$\hat{\mathbf{L}} = \hat{\mathbf{D}} - \hat{\mathbf{W}}. \tag{2.46}$$

The graph term can then be constructed as

$$\mathcal{L}_g(\mathbf{\Theta}) = \frac{\lambda}{N} \text{Tr}\left[\mathbf{Z}^T \hat{\mathbf{L}} \mathbf{Z}\right]. \tag{2.47}$$

where $\mathbf{\Theta}$ denotes the weights and biases in the network. The λ parameter is a weighting of the graph term, and N is the number of pixels. This term can then be used in the loss function. Figure 2.20 illustrates the process that was just explained. The loss function of the RGAE can be created using a combination of the $\ell_{2,1}$ norm and the graph regularization term. The $\ell_{2,1}$ norm is expressed as

$$\ell_{2,1} = ||\hat{\mathbf{X}} - \mathbf{X}||_{2,1} = \sqrt{\sum_{h=1}^{H} \sum_{j=W}^{W} (\mathbf{X}_{h,w} - \hat{\mathbf{X}}_{h,w})^2} \tag{2.48}$$

so that the entire loss is expressed as

$$\mathcal{L}_{\text{RGAE}}(\hat{\mathbf{X}}, \mathbf{X}, \mathbf{Z}) = \frac{1}{2N} ||\hat{\mathbf{X}} - \mathbf{X}||_{2,1} + \frac{\lambda}{N} \text{Tr}[\mathbf{Z}^{\text{T}} \hat{\mathbf{L}} \mathbf{Z}]. \tag{2.49}$$

Using the $\ell_{2,1}$ norm rather than the MSE loss makes the RGAE more robust to noise and anomalies as it is less sensitive to outliers [6]. The objective function of the model is the same as the loss so that the goal is to minimize $J(\Theta)$ [53] expressed as

$$J(\Theta) = \min_{\Theta} \left(\frac{1}{2N} ||\hat{\mathbf{X}} - \mathbf{X}||_{2,1} + \frac{\lambda}{N} \text{Tr}\left[\mathbf{Z}^T \hat{\mathbf{L}} \mathbf{Z}\right] \right). \tag{2.50}$$

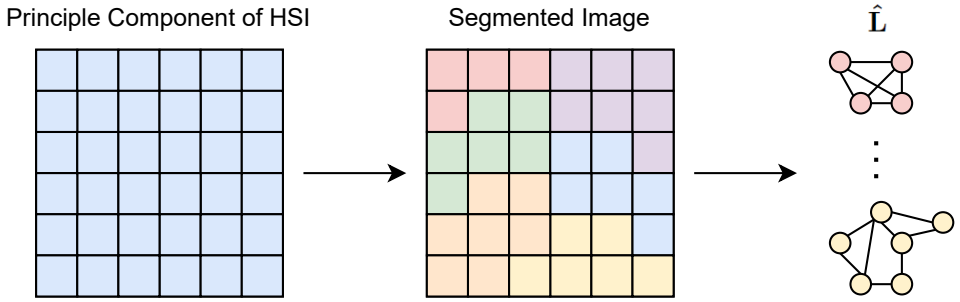


Figure 2.20: The creation of the Laplacian matrix $\hat{\mathbf{L}}$ [6].

All input parameters for the Robust Graph Autoencoder are listed in Table 2.2. λ, K, n_hid, and μ are parameters that must be optimized for each dataset. As the table explains, the μ parameter is the learning rate of the Autoencoder utilized by the RGAE.

Table 2.2: Input parameters for the RGAE.

Parameter	**Denotation**
Image	The HSI
λ	The representation of the two terms in the objective function
K	The number of superpixels
n_hid	The number of hidden layers
μ	The learning rate
n_epochs	The number of maximum epochs

Chapter 3

Proposed Enhanced Graph Autoencoders for HAD

The Robust Graph Autoencoder has proven to be a powerful anomaly detector in terms of documented performance Since it is a relatively recently proposed HAD, it might have room for improvement. This chapter will introduce the proposed HADs based on the Robust Graph Autoencoder with the goal of boosting the detection performance. These changes are to the preprocessing aspect, as shown in Section 3.1. Section 3.2 propose a more significant change, using multiple preprocessing methods for two RGAEs in parallel. An RGAE with a change of layer setup is presented in Section 3.3. Other changes regarding the network architecture are also investigated. This is done by replacing the Autoencoder with an Interpolated Autoencoder. The Hyperspectral Anomaly Detectors utilizing this architecture are described in detail in Section 3.4, Section 3.5 and Section 3.6. The difference between the three proposed models in these sections is the additions of preprocessing presented for some of the proposed HADs. A detailed explanation of the Interpolated Autoencoder will be presented in Section 3.4 before the description of the Interpolated Graph Autoencoder.

A note for all of the proposed methods is that they are isolated, meaning that each change is implemented and tested without adding other changes. In a later chapter, each proposed HAD will also be compared to the conventional Robust Graph Autoencoder. Finally, all proposed HADs and the conventional RGAE are implemented with early stopping, the ADAM optimizer and normalization of the HSIs.

3.1 Preprocessing based RGAE for HAD

The conventional RGAE lacks preprocessing. Adding preprocessing has the potential to boost detection performance. We propose a preprocessing based RGAE, wich will be tested out with several methods of preprocessing. A new block called Preprocessing has therefore been added. In this block, different methods of preprocessing are utilized. This new Hyperspectral Anomaly Detector is illustrated in Figure 3.1 where the red block highlights the new addition to the existing RGAE. Preprocessing methods that will be tested are default PCA, KPCA, and RPCA. For the KPCA, several kernels will be tested out, being the Sigmoid, Laplacian, Polynomial, and Exponential kernels. In terms of utilizing RPCA, both the signal space and the sparse signal will be tested.

For every method of preprocessing, except the RPCA, the first component of the output data from the Preprocessing block is chosen in the Spectral Band Reduction block. In the RPCA case, all components are summarized. The difference in approaches is due to the PCA and KPCA outputting the best features in the first component, which is not the case for RPCA.

With some exceptions, this preprocessing implementation uses a Matlab implementation of the Robust Graph Autoencoder [6]. To compute the PCA, the pre-existing code from the creators of the RGAE is utilized. For the KPCA, the Matlab code created by X. Song et al. is utilized [54]. In terms of RPCA implementation, the decomposition package from the sklearn toolbox in

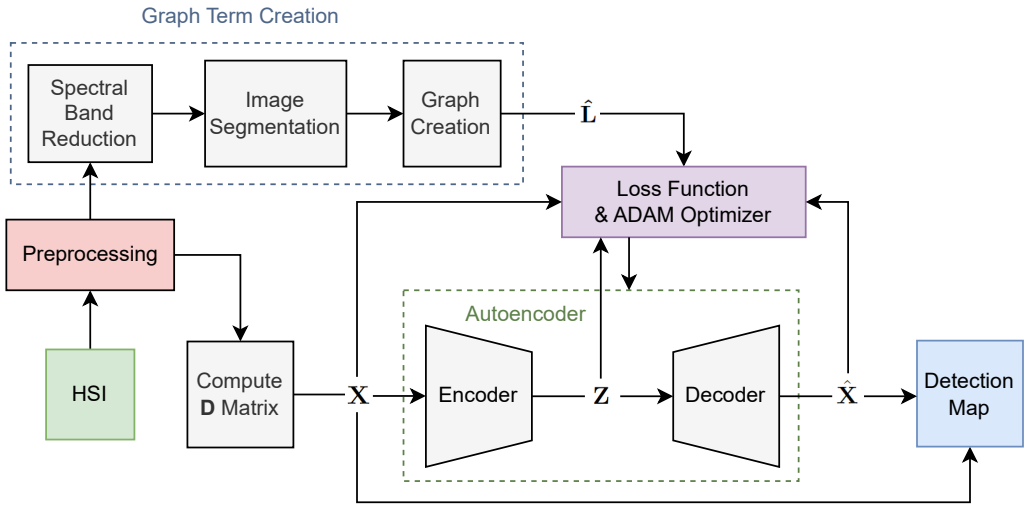


Figure 3.1: Representation of a RGAE with modified preprocessing.

Python is used [55]. The RPCA components are then loaded in Matlab for further use. Each preprocessing method will be tested using a dimensionality of $[50, 100, 300]$ to check whether this makes a difference in detection performance.

3.2 PCA Based Multikernel RGAE for HAD

The Multikernel RGAE (MK-RGAE) utilize multiple kernels for the KPCA using two parallel RGAEs. In Figure 3.2, an overview of the entire HAD is presented. The Hyperspectral Image is first sent as input to both RGAE blocks. These blocks are each a fully functional RGAE that uses KPCA as preprocessing as input for both the graph creation and the Autoencoder. The difference between the RGAE 1 and RGAE 2 is that they use different kernels for preprocessing. After the two RGAEs finish their output, their detection maps are combined using decision fusion to produce the final detection map. Combining the two detection maps is expressed as

$$X_{\text{det}} = \alpha_{\text{MK-RGAE}} X_{det,1} + (1 - \alpha_{\text{MK-RGAE}}) X_{det,2} \tag{3.1}$$

where $X_{det,1}$ is the detection map of the RGAE 1 and $X_{det,2}$ is the detection map of the RGAE 2. To optimize the $\alpha_{\text{MK-RGAE}}$ parameter, it will be chosen from a grid search from 0 to 1 using an increase of 0.01 for each optimization test.

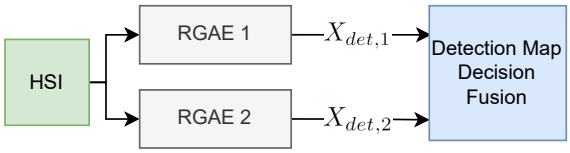


Figure 3.2: The multikernel RGAE.

Figure 3.3 shows an overview of how each of the RGAE blocks in the previous figure is designed (They follow the same design). Regarding the difference to the conventional RGAE, preprocessing is the only deviance. PCA is no longer performed for the dimensionality reduction in the Spectral Band Reduction block, as it has been fully replaced by KPCA. As with the Preprocessing based RGAE for HAD, the first component of the KPCA output is used and sent to the Image Segmentation block. The input of the Autoencoder itself has also been changed from the default HSI to using KPCA. As shown in the figure, the final output $X_{det,i}$ of the network is sent to the decision fusion of the two detection maps.

This Hyperspectral Anomaly Detector is implemented using the Matlab version of the Robust Graph Autoencoder as a backbone [6]. Everything regarding the implementation of KPCA utilizes

the code by X. Song et al. [54].

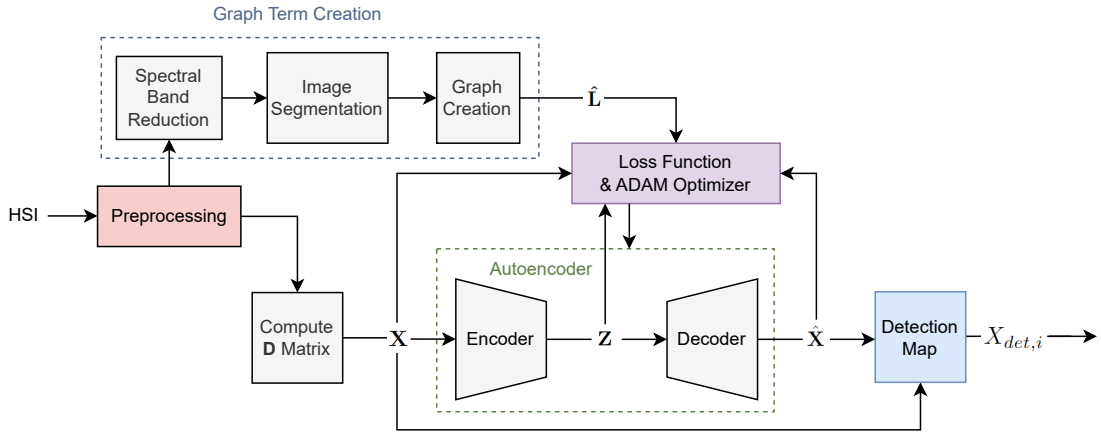


Figure 3.3: The RGAE blocks in the Multikernel RGAE.

3.3 RGAE with Additional Layers

This proposed HAD will add complexity to the neural network of the conventional Robust Graph Autoencoder. With this alteration, more layers are added to the encoder and the decoder of the Autoencoder. The number of layers in the encoder and decoder is increased from one to two layers as more layers can increase the learning capacity of the neural network [56].

Table 3.1 informs about the dimensionality of the new layer setup and the activation functions used. Layer 1, as with the conventional RGAE, acts as a standard input layer without applying weight or adding bias. Layer 2 and layer 3 act as the encoding part of the network, while layers 3 and 4 decode the latent variable. The parameter n_hid_2 has half the size of n_hid_1, so that $n\_hid_2 = \frac{1}{2} n\_hid_1$. The number of hidden dimensions n_hid_1 is chosen on a grid search among the values $[50, 100, 200]$. Regarding activation functions, the last encoding layer and the last decoding layer utilize the Sigmoid function. The other two layers will be tested with the Sigmoid and the ReLU activation functions. The layer setup using ReLU-Sigmoid for the encoder and ReLU-Sigmoid for the decoder will be referred to as the ReLU/Sigmoid setup, while the setup using Sigmoid-Sigmoid for both the encoder and the decoder will be referred to as the Sigmoid/Sigmoid setup. A final note for this Hyperspectral Anomaly Detector is that it is implemented using Matlab. The preexisting code of the Robust Graph Autoencoder is used as a backbone [6].

Table 3.1: Layer setup for the RGAE utilizing multiple layers in the encoder and the decoder.

Layer	**Weight size**	**Bias size**	**Activation function**	**Number of parameters**
Layer 1 (Input)	-	-	-	-
Layer 2	$n\_hid_1 \times F$	n_hid_1	ReLU/Sigmoid	$n\_hid_1(F+1)$
Layer 3	$n\_hid_2 \times F$	n_hid_2	Sigmoid	$n\_hid_2(F+1)$
Layer 4	$F \times n\_hid_2$	F	ReLU/Sigmoid	$n\_hid_2(F+1)$
Layer 5	$F \times n\_hid_1$	F	Sigmoid	$n\_hid_1(F+1)$

3.4 Interpolated Graph Autoencoder for HAD

As mentioned in the introduction of the chapter, the implementation of an Interpolated Autoencoder (IAE) will be tested. This neural network architecture replaces the AE, which is utilized by the Robust Graph Autoencoder. Interpolated Autoencoders are based on interpolating several encoded latent variables before decoding the interpolated result. Interpreting latent variables can

result in semantically meaningful data and better generalization than using a conventional AE [8]. This is due to the Interpolated Autoencoder (IAE) working with a more varied set of data. An IAE uses two (or more) inputs. Since data from the two inputs $\mathbf{X}_1$ and $\mathbf{X}_2$ are randomly sampled during training, the inputs often contain different features. This makes the IAE adapt to a more varied set of data rather than memorizing the expected input [8]. Figure 3.4 shows the process of encoding and decoding the multiple inputs to a singular interpolated output. In this case, the inputs represent the data matrices $\mathbf{D}$ (Explained in Section 2.3.1) of the same Hyperspectral Image.

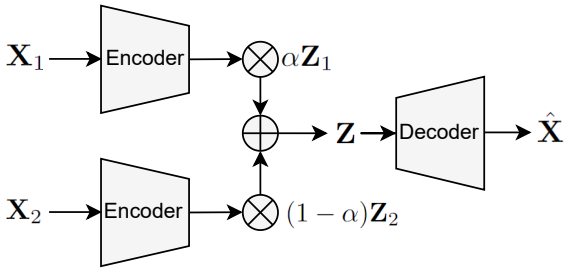


Figure 3.4: An overview of an Interpolated Autoencoder utilizing two encoders and decision fusion to interpolate the latent-variables $\mathbf{Z}_1$ and $\mathbf{Z}_2$.

The encoder works like with a normal Autoencoder except for encoding several inputs in parallel [8]. Encoding an input $\mathbf{X}_i$ could therefore be expressed as

$$\mathbf{Z}_i = f_\theta(\mathbf{X}_i). \tag{3.2}$$

where i denotes the numbered input. In terms of the decoding process, the multiple latent variables have to be interpolated using decision fusion [8] so that

$$\mathbf{Z} = \alpha\mathbf{Z}_1 + (1-\alpha)\mathbf{Z}_2, \quad \alpha \in [0,1]. \tag{3.3}$$

This is valid for an Interpolating Autoencoder utilizing two encoders in parallel. The expression for the output of the decoder [8] is then

$$\hat{\mathbf{X}}_\alpha = g_\phi(\mathbf{Z}). \tag{3.4}$$

As the interpolated autoencoder relies on a weighting of the two latent variables, the loss has to use the same weighting if reconstruction loss is to be utilized. The reconstruction loss is based on the difference between the interpolation of the two inputs and the output of the entire autoencoder. The MSE loss of an interpolated autoencoder can therefore be expressed as

$$\mathcal{L}_{\text{I-MSE}} = \frac{1}{N}\sum_{i=1}^{N} ||\hat{\mathbf{x}}_{\alpha,i} - \mathbf{x}_{\alpha,i}||^2 \tag{3.5}$$

where $\hat{\mathbf{x}}_{\alpha,i}$ is the reconstructed version of $\mathbf{x}_{\alpha,i}$, where the latter is expressed as

$$\mathbf{x}_{\alpha,i} = \alpha\mathbf{x}_{1,i} + (1-\alpha)\mathbf{x}_{2,i} \tag{3.6}$$

where $\mathbf{x}_{1,i}$ and $\mathbf{x}_{2,i}$ are pixels within the inputs $\mathbf{X}_1$ and $\mathbf{X}_2$ respectively. N represents the number of pixels.

One of the proposed HADs using this network topology is illustrated in Figure 3.5 with the red colors highlighting the new implementations. This new setup will, from now on, be referred to as the Interpolated Graph Autoencoder (IGAE). Here, the HSI is used for both inputs and fused after being encoded. Each input is still denoted as $\mathbf{X}_1$ and $\mathbf{X}_2$. As the data is chosen randomly from each input when training, there are hopes of obtaining some semantically meaningful data from the fusion. The parallel encoders share their weights and biases.

In terms of parameters, the Interpolated Graph Autoencoder (IGAE) has all the same parameters as the conventional RGAE (Table 2.2), as well as the α parameter determining the balance of the decision fusion. For the loss, it is calculated as with the conventional Robust Graph Autoencoder using the $\ell_{2,1}$ norm rather than the Mean Square Error loss, and the graph regularization term. The loss is therefore expressed as

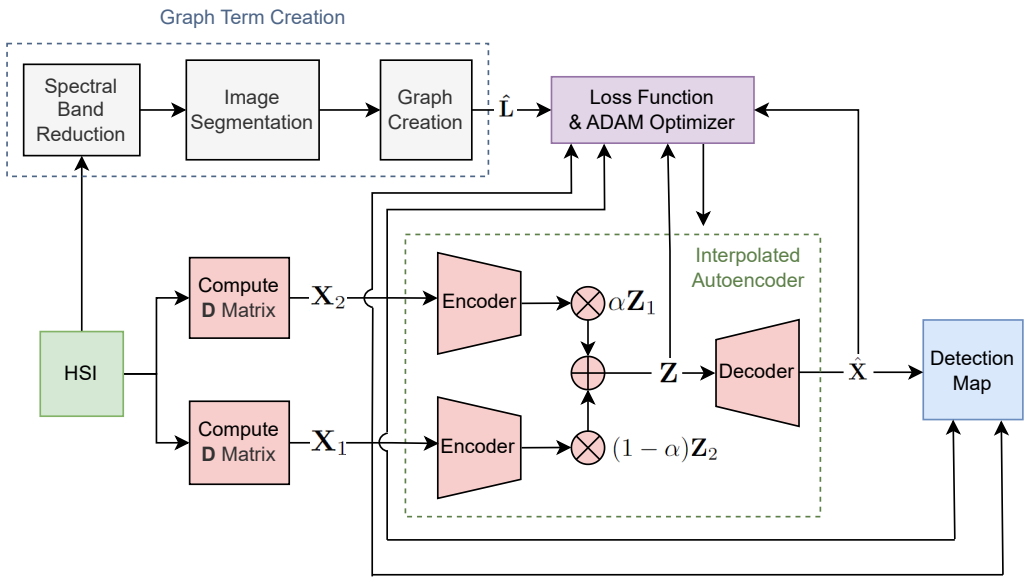


Figure 3.5: Representation of the Interpolated Autoencoder.

$$\mathcal{L}_{\text{IGAE}} = \frac{1}{2N}||\hat{\mathbf{X}}_\alpha - \mathbf{X}_\alpha||_{2,1} + \frac{\lambda}{N}\text{Tr}[\mathbf{Z}^{\text{T}}\hat{\mathbf{L}}\mathbf{Z}] \tag{3.7}$$

where

$$\mathbf{X}_\alpha = \alpha\mathbf{X}_1 + (1-\alpha)\mathbf{X}_2. \tag{3.8}$$

The detection map is calculated using $\mathbf{X}_\alpha$ and $\hat{\mathbf{X}}_\alpha$. This is also illustrated in Figure 3.5 as the inputs $\mathbf{X}_1$ and $\mathbf{X}_2$ are passed into the Detection Map block along with the network output $\hat{\mathbf{X}}_\alpha$. To optimize the α value, a grid search from 0 to 1 using 0.1 increments is used. Since the entire neural network needs to be trained for each test of a specific α parameter value, the grid search for the parameter is not the same incremental value as the MK-RGAE since it would be highly time-consuming.

To ease with implementing this Hyperspectral Anomaly Detector, the PyTorch package in Python is utilized [57]. Since the RGAE is intended to be a backbone, it is translated from Matlab to Python using the preexisting code [6].

3.5 PCA Based Kernel Interpolated Graph Autoencoder for HAD

The Interpolated Graph Autoencoder does not include the use of preprocessing. This proposed HAD therefore introduces preprocessing to the IGAE in the previous section. The modified Interpolated Graph Autoencoder is named the Kernel Interpolated Graph Autoencoder (K-IGAE) due to the use of KPCA. Figure 3.6 illustrates the new setup with the red block highlighting the new implementation to the IGAE. In the Preprocessing block, KPCA is performed. This setup is tested to check if the combination of latent variables from a HSI that has been subject to preprocessing and a default HSI can provide a better detection map. As the inputs $\mathbf{X}_1$ and $\mathbf{X}_2$ for the Encoder need to share the same dimensionality, the output of the Preprocessing block has the same number of spectral dimensions as the original HSI (F). $\mathbf{X}_1$ is the KPCA output and $\mathbf{X}_2$ is the original HSI. For the graph creation, PCA is still performed in the Spectral Band Reduction block.

As with the conventional RGAE, the decoder takes $\mathbf{Z}$ as input and computes $\hat{\mathbf{X}}$. The encoder and the decoder are composed of the same number of layers, weights and biases, as with the conventional RGAE. The loss function $\mathcal{L}_{\text{K-IGAE}}$ is a combination of both the graph regularization term and the $\ell_{2,1}$ norm, expressed as shown in Equation 3.7.

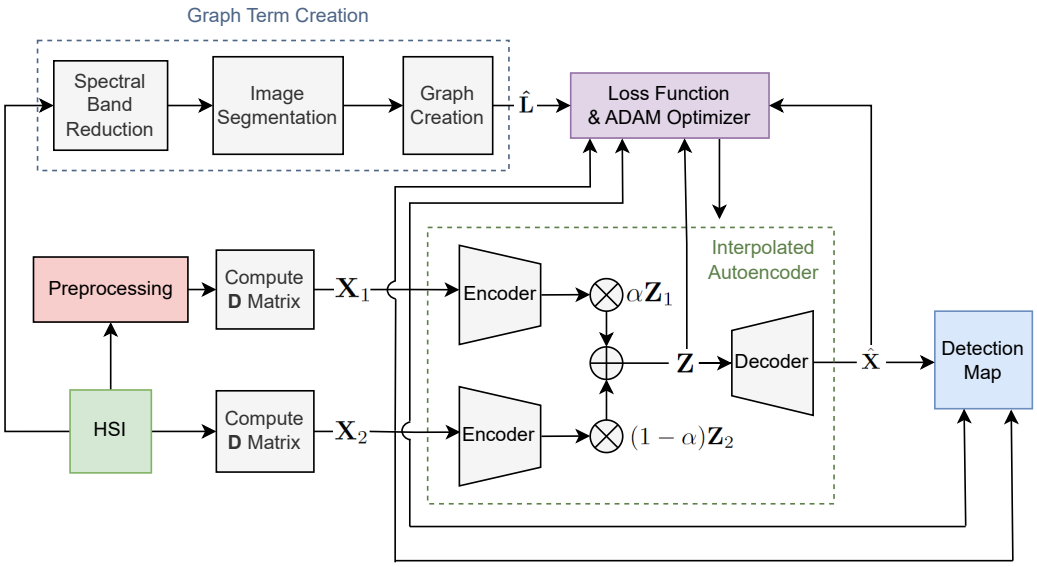


Figure 3.6: Representation of the Interpolated Autoencoder utilizing KPCA.

Like implementing the IGAE, the K-IGAE utilizes the PyTorch package to define the neural network [57]. The Python translation of the Matlab implementation of the Robust Graph Autoencoder is used as a backbone [6]. In terms of KPCA computation, the code created by X. Song et al. is also used [54].

3.6 PCA Based Multikernel Interpolated Graph Autoencoder for HAD

This proposed HAD also utilizes the Interpolated Autoencoder explained in Section 3.4. What makes this HAD different from the IGAE and the K-IGAE is the fact that it utilizes KPCA for both of the inputs of the parallel encoders, which is why it is named the Multikernel Interpolated Graph Autoencoder (MK-IGAE). Figure 3.7 illustrates the new design of the Hyperspectral Anomaly Detector.

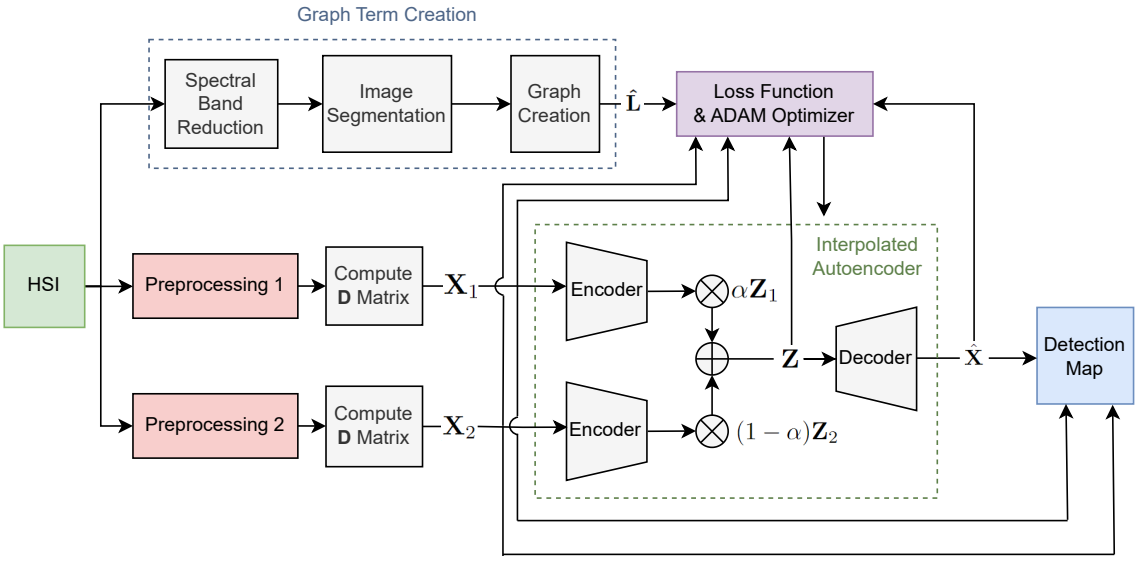


Figure 3.7: Representation of the Interpolated Autoencoder utilizing multiple kernels for KPCA.

The red blocks indicate the new additions that separate it from the IGAE. Two inputs, $\mathbf{X}_1$ and $\mathbf{X}_2$, are sent into the encoders and then combined using decision fusion of the latent variables $\mathbf{Z}_1$ and $\mathbf{Z}_2$. Each of the inputs results from performing KPCA using two different kernels on the HSI. For the layer setups, the encoder and decoder share the same dimensionalities as the RGAE as shown in Table 2.1. In terms of loss, it is calculated the same way as for the IGAE and the K-IGAE, shown in Equation 3.7.

The Preprocessing blocks are implemented using Matlab using the library described in Section 3.1 [54]. The Spectral Band Reduction and the entire Graph term creation segment are performed in Matlab. To accomplish this, the preexisting code for the RGAE is utilized. The neural network is implemented in Python using the PyTorch package [57]. Here, the inputs $\mathbf{X}_1$, $\mathbf{X}_2$, the Laplacian matrix $\hat{\mathbf{L}}$ and the ground truth of the Hyperspectral Image are loaded from saved Matlab data files (.mat). They are then sent into the neural network.

Chapter 4

Results

This Chapter provides an analysis of the datasets used to measure the detection performance of each proposed Hyperspectral Anomaly Detector. Following the dataset analysis, a description of the test setup utilized to obtain the results achieved in this thesis is provided. Experimental testing results are then shown, illustrating the performance of the HADs for different test setups. Each HAD will be compared with the conventional RGAE, but not with each other. The chapter's final section compares the different proposed HADs detection performance and time score. This will highlight what proposed HADs worked the best in terms of the goal of the thesis.

4.1 Dataset analysis

The datasets used to measure the performances of the proposed HADs are the ABU datasets [58]. These datasets can be sectioned into three categories, airport images, beach images, and images of urban areas. There are four datasets of airports, 4 of beaches, and 5 of urban scenarios. In terms of analysis, data size, numerical properties, spectral and spatial analysis' are presented. The first section, being Section 4.1.1, focuses on the numerical properties of each dataset. Section 4.1.2 analyzes the spectral signatures for a handful of datasets. Following that section, a spatial analysis is shown (Section 4.1.3) with an exploration of how the preprocessing affects the data.

4.1.1 Numerical Properties of the Datasets

Table 4.1 shows the specifications of each dataset in terms of size. The table shows that the number of spectral bands varies from 102-207 bands. Regarding spatial resolution, all images except abu-beach-1 and abu-beach-4 are 100×100 pixels. The two mentioned datasets are larger in spatial resolution, being 150×150 pixels. For the preprocessing methods, the larger datasets will likely demand more processing power resulting in longer computational time. The worst datasets in terms of this are the abu-beach-1 and the abu-beach-4 datasets, as their number of pixels are over two times as many as any other dataset.

Table 4.2 shows an overview of other numerical properties of each dataset. As seen from the table, the number of anomalies varies greatly from dataset to dataset. The abu-beach-3 dataset only contains one single anomaly, while abu-beach-2 contains 58. However, this does not paint the entire picture, as the number of anomalous pixels to all pixels ratio will affect an Autoencoder's ability to replicate anomalies. The more anomalies present in the dataset, given that they have similar spectral properties, the more likely the autoencoder is to be able to reproduce the anomalies. This could be a problem for datasets such as abu-airport-3, abu-beach-2, and abu-urban-3, assuming the spectral properties of the anomalies are similar. Datasets that could prove easier to detect anomalies based on the numerical properties are likely to be abu-beach-1 and abu-beach-3 as these contain a low percentage of anomalous pixels that could cause the autoencoders severe problems to replicate.

Table 4.1: The height, width, number of spectral dimensions and the size of each dataset.

Dataset	H	W	F	**Size**
abu-airport-1	100	100	205	2050000
abu-airport-2	100	100	205	2050000
abu-airport-3	100	100	205	2050000
abu-airport-4	100	100	191	1910000
abu-beach-1	150	150	188	4230000
abu-beach-2	100	100	193	1930000
abu-beach-3	100	100	188	1880000
abu-beach-4	150	150	102	2295000
abu-urban-1	100	100	204	2040000
abu-urban-2	100	100	207	2070000
abu-urban-3	100	100	191	1910000
abu-urban-4	100	100	205	2050000
abu-urban-5	100	100	205	2050000

Table 4.2: This table shows the anomaly pixels to all pixels ratio, the number of anomalies, the number of anomalous pixels and the minimal and maximal anomaly size (in pixels) for each dataset.

Dataset	**Anomaly Ratio**	**Number of Anomalies**	**Number of Anomalous Pixels**	**Minimal Anomaly Size**	**Maximal Anomaly Size**
abu-airport-1	0.0144	13	144	2	19
abu-airport-2	0.0087	2	87	43	44
abu-airport-3	0.0170	16	170	2	49
abu-airport-4	0.0060	3	60	10	39
abu-beach-1	0.0008	1	19	19	19
abu-beach-2	0.0202	58	202	1	7
abu-beach-3	0.0011	1	11	11	11
abu-beach-4	0.0030	7	68	3	18
abu-urban-1	0.0067	9	67	3	14
abu-urban-2	0.0155	20	155	4	16
abu-urban-3	0.0052	11	52	2	10
abu-urban-4	0.0272	25	272	4	34
abu-urban-5	0.0232	30	232	2	13

Table 4.3: This table shows the range of spectral values in each dataset.

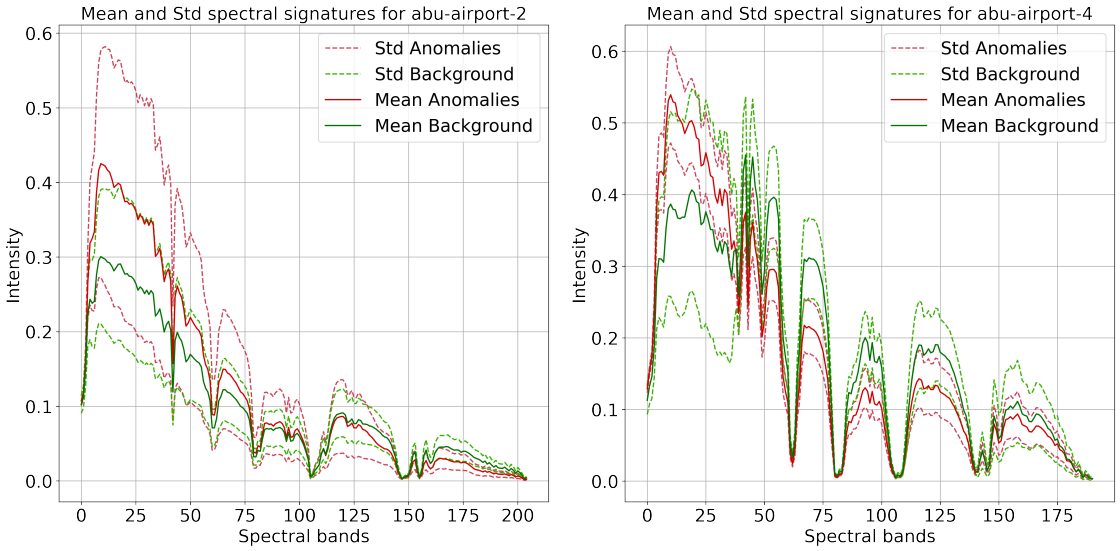
Dataset	**Minimal Value**	**Maximal Value**
abu-airport-1	0	6604
abu-airport-2	0	6644
abu-airport-3	-1	5651
abu-airport-4	-1	5061
abu-beach-1	-32	6593
abu-beach-2	-19	4100
abu-beach-3	-34	5404
abu-beach-4	0	1
abu-urban-1	-50	6534
abu-urban-2	-50	14060
abu-urban-3	0	7208
abu-urban-4	-2	19492
abu-urban-5	-1	7120

Table 4.3 shows each dataset's minimal and maximal value. Regarding the range of values within each dataset, the wide spread of spectral intensities proves the need for normalization. This is therefore done for all of the proposed HADs and the conventional RGAE.

4.1.2 Spectral Analysis

Two datasets from each of the three scenarios have been chosen for the spectral analysis. This is done to highlight the difference between the best and worst anomalies to background separability for each scenario. Each dataset has its corresponding plot illustrating the mean value of the two categories, background and anomalous pixels. The standard deviation for both categories is also included in these plots. Low overlap between the two categories does not necessarily mean that the dataset contains anomalies that are easy to classify. This is due to the background and the anomalies possibly deviating from their respective mean graphs. However, this analysis can indicate which datasets are easier to separate the anomalies from the background. Finally, all data has been normalized before calculating the mean and standard deviation for the anomalies and the background.

Figure 4.1 illustrates spectral plots for the datasets abu-airport-2 and abu-airport-4. For abu-airport-2, the first 50 spectral bands show a gap between the anomalies' mean and the background's mean. This is seen in Figure 4.1a. However, the standard deviation of each category shows that there is an overlap between the intensities of the anomalies and the background. On the other hand, it shows slightly better separability than the abu-airport-4 dataset when looking at the first 50 spectral bands. It is worth noting that this does not apply to the other spectral bands. When looking at spectral band 60 and onwards, the abu-airport-4 dataset shows slightly better separability, although the relative spectral intensity difference between the mean anomaly and the mean background is minimal.

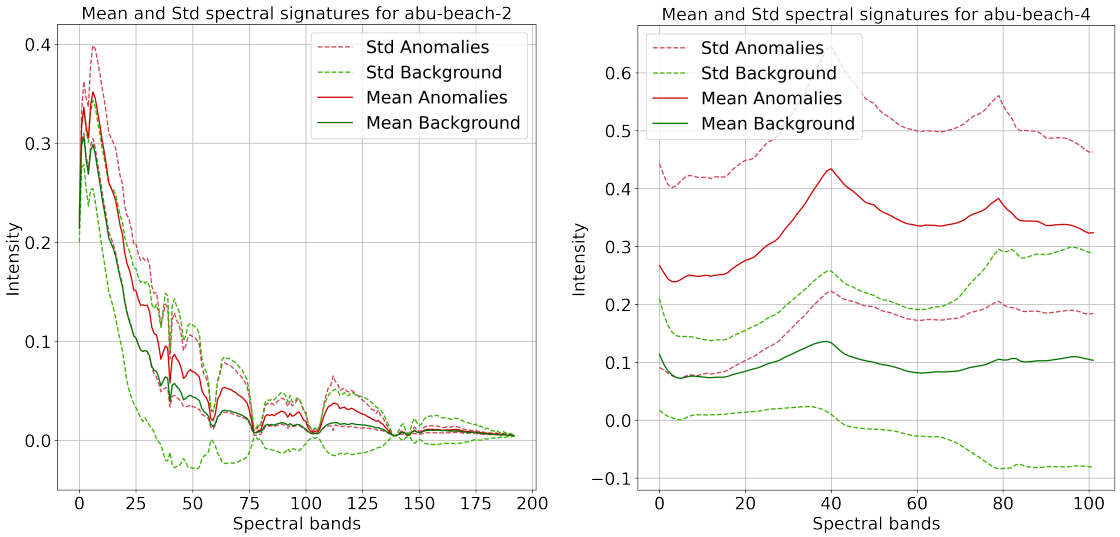


(a) abu-airport-2 **(b)** abu-airport-4

Figure 4.1: Spectral analysis plot of mean and standard deviation of anomalies and background pixels for abu-airport-2 and abu-airport-4.

In Figure 4.2, an overview of the spectral plots for the abu-beach-2 and abu-beach-4 can be seen. The first noticeable difference is that the overlap between the anomalies and background pixels in the spectral signature is far worse for the abu-beach-2 dataset. This could indicate that separating the anomalies from the background in the abu-beach-4 dataset will be easier. Regarding the standard deviation for the abu-beach-2 dataset, there is a relatively big span of spectral intensity for almost all of the bands. This could be a result of noise within the image.

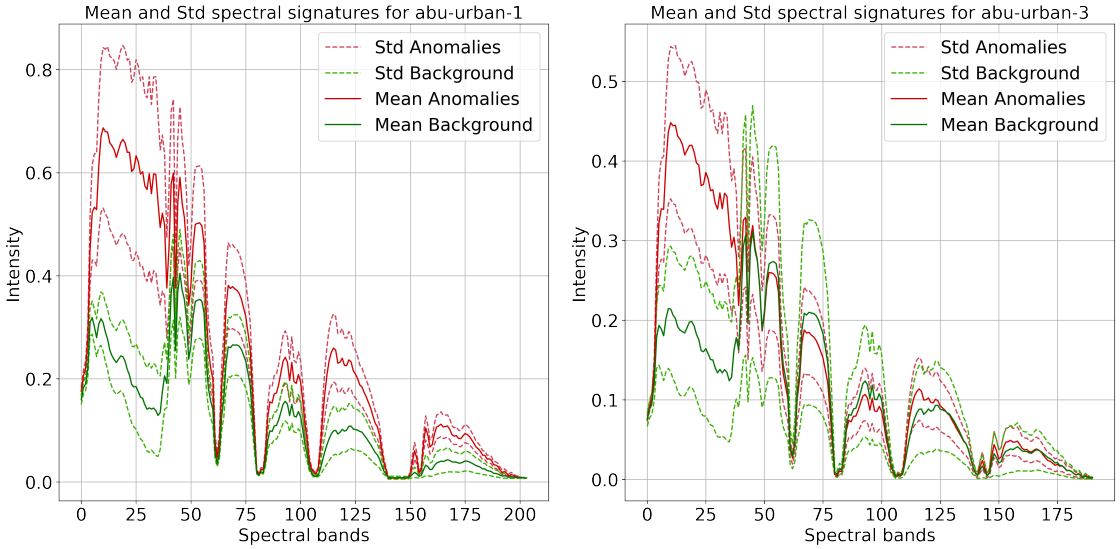
For the urban scenarios, the spectral plots of abu-urban-1 and abu-urban-3 can be seen in Figure 4.3. As seen in Figure 4.3a, there is no overlap between the mean and standard deviation of the two categories in the first 40-45 spectral bands. This indicates high separability and potentially an easy dataset to identify anomalies. Although not bad, the abu-urban-3 dataset shows a slight



(a) abu-beach-2 **(b)** abu-beach-4

Figure 4.2: Spectral analysis plot of mean and standard deviation of anomalies and background pixels for abu-beach-2 and abu-beach-4.

overlap in the same first spectral bands. The mean spectral signature of the background swaps from being lower than that of the anomaly mean to higher for the abu-urban-3 dataset from about the 60th spectral band and onwards. This is not the case for the abu-urban-1 dataset, as the mean of the anomaly is constantly higher than that of the background.



(a) abu-urban-1 **(b)** abu-urban-3

Figure 4.3: Spectral analysis plot of mean and standard deviation of anomalies and background pixels for abu-urban-1 and abu-urban-3.

4.1.3 Spatial Analysis

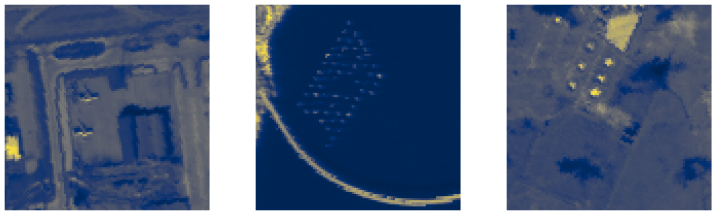
For the spatial analysis, the three datasets abu-airport-2, abu-beach-2, and abu-urban-1 have been chosen for a qualitative analysis of the preprocessing methods. These datasets have been chosen as they vary in anomaly to background separability, as described in the previous section. Each preprocessing method, along with the default HSI image, has been compressed into one spectral dimension to compare the different methods better. This is done by summarizing all spectral bands. To help locate each anomaly in the given dataset, Figure 4.4 has been included. This figure

illustrates the location and size of the anomalies in each of the three datasets.

(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.4: Ground truth overview of three datasets.

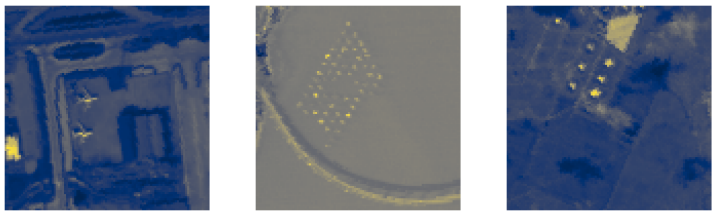
Figure 4.5 illustrates the default HSI. Although visible to the human eye, the anomalies are weak in overall intensity. In Figure 4.5a, the planes (which are the anomalies) are noticeable, but only the central parts of them. As seen in the abu-urban-1 plot, the anomalies are very easy to locate, which was also suggested by the spectral analysis of the dataset. Noise is a potential problem as both the object in the lower left of Figure 4.5a, the curve in Figure 4.5b, and the large object in Figure 4.5c are not considered to be anomalies. The noise in the abu-beach-2 dataset was also indicated in the spectral analysis.



(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.5: Spatial overview of three of the default (unprocessed) datasets.

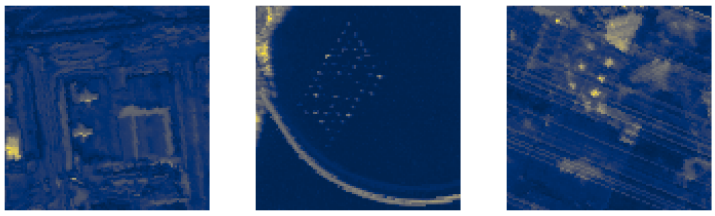
Some visible improvements upon the anomalies visibility have been achieved using PCA, as seen in Figure 4.6. In Figure 4.6a, the planes have a larger area of high spectral intensity compared with the default image in Figure 4.5a. Noise is still an issue, but the curve in Figure 4.6b and the object in Figure 4.6c have slightly reduced intensity. The background of the abu-beach-2 dataset has also seen a reduction in intensity, which highlights the anomalies even more. With the slight noise reduction, this could be an improvement upon just using the default HSI as input.



(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.6: Spatial overview of PCA performed on three of the datasets.

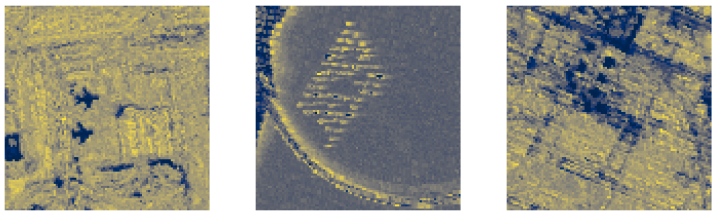
The results are harder to interpret with the appliance of KPCA utilizing the Sigmoid kernel. This can be seen in Figure 4.7. The intensity levels for the noise objects mentioned earlier have seen a significant reduction in comparison to the default HSI image. However, the occurrence and intensity of other noise has increased. A result of this is a more unclear image. This is most noticeable for the abu-urban-1 dataset in Figure 4.7c. In terms of the anomalies, they are almost identical in intensity compared to not using any form of preprocessing. On the other hand, it seems like more of the actual anomalies have been highlighted. Since the noisy objects have seen a considerable decrease in intensity without negatively affecting the anomalies, this could prove to be a good preprocessing method.



(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.7: Spatial overview of KPCA utilizing the Sigmoid kernel performed on three of the datasets.

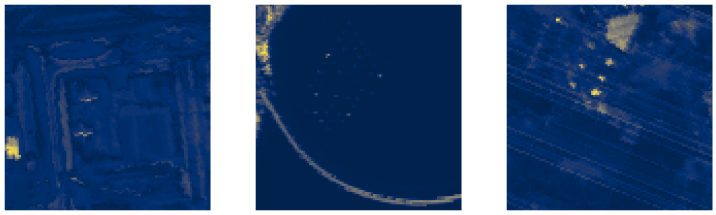
Figure 4.8 shows the use of KPCA with the Laplacian kernel on the default dataset. The first notable change is the inversion of overall intensity as most of the anomalies' spectral intensity has been lowered and the background increased. That said, the anomalies for the abu-airport-2 dataset have a somewhat clear outline. However, this can not be said for the abu-beach-2 and abu-urban-1 datasets, as both seem very noisy. It is also unclear whether an autoencoder can pick up upon the anomalies when their intensity has been lowered. As the images are inverted and the intensity of the anomalies are reduced by a large margin, it is hard to believe that this form of preprocessing will achieve better results than that of not using any form of preprocessing methods.



(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.8: Spatial overview of KPCA utilizing the Laplacian kernel performed on three of the datasets.

With the use of KPCA using the Polynomial kernel, as seen in Figure 4.9, the intensity of the anomalies has been reduced. Some anomalies have even disappeared entirely, indicating that this kernel should not result in better detection performance when used as preprocessing for a Hyperspectral Anomaly Detector. On the other hand, the background seems more uniform, which could result in the opposite effect.

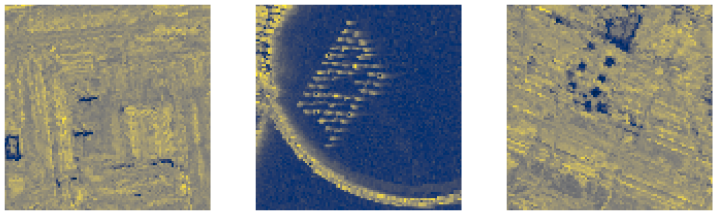


(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.9: Spatial overview of KPCA utilizing the Polynomial kernel performed on three of the datasets.

As with the Laplacian kernel, the Exponential kernel produces an inverted image when using KPCA on the default datasets. When looking at the abu-beach-2 dataset, the highlight around the anomalies has a high intensity, while the center of the anomalies is pretty dark. This could potentially confuse a HAD. For the abu-airport-2 and abu-urban-1, both datasets have reasonably well-defined anomalies. There is, however, a lot of noise in the urban scenario and it is hard to say whether the autoencoders can output a relevant reconstruction of the HSI given the inverted anom-

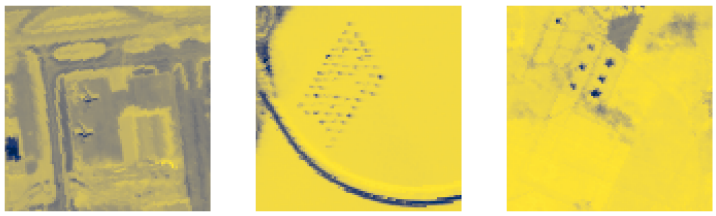
alies. Therefore, It should not be expected to see an improvement when utilizing the Exponential kernel for KPCA as preprocessing.



(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.10: Spatial overview of KPCA utilizing the Exponential kernel performed on three of the datasets.

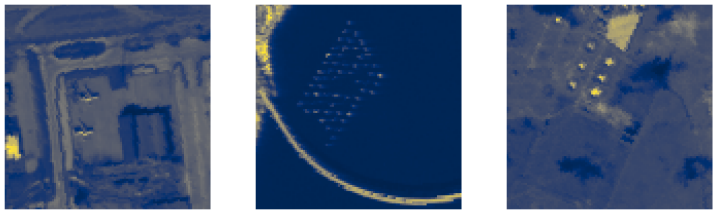
Figure 4.11 represents the low rank representation when performing RPCA on each dataset. As expected with this method, the anomalies have been reduced in intensity. A lot of the noise has been reduced as well. On the other hand, this means that the noise will be included in the sparse representation.



(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.11: Spatial overview of the low rank representation of RPCA performed on three of the datasets.

Figure 4.12 shows an overview of the sparse representation when performing RPCA on the datasets. It is very close to the original dataset regarding the intensity of the anomalies and the noise, but the background has been slightly reduced. As this preprocessing method hasn't proven to be that adequate to the human eye, a huge performance boost is not likely utilizing this method.



(a) abu-airport-2 **(b)** abu-beach-2 **(c)** abu-urban-1

Figure 4.12: Spatial overview of the sparse representation of RPCA performed on three of the datasets.

4.2 Results From Testing

This section will go through all the experimental testing performed on the different proposed changes and implementations done to the Robust Graph Autoencoder. The first two subsections describe the test setup and the optimization of the conventional RGAE, respectively. The RGAE had to be optimized as all the proposed HADs utilize some, if not all, of the RGAEs parameters. Section 4.2.3 goes over the tests done to the preprocessing additions as well as the change in layers while Section 4.2.4 goes over the testing of the Interpolated Autoencoder implementation.

Each section will give a presentation of the detection performance achieved from each setup and a parameter description for the best-performing implementation where it needs to be specified. ROC curves and detection maps are presented to help visualize the detection performances.

4.2.1 Test Setup

Table 4.4 showcases the system specifications of the computer utilized for training models implemented using Python to define the neural network. The setup includes an AMD Ryzen 5 5600X processor. It is accompanied by an NVIDIA GeForce RTX 3060 graphics card with 12GB of video memory. The computer also has 16GB of DDR4 RAM, ensuring efficient memory management during model training.

Table 4.4: System specifications for the computer used when training the different models.

Item	Specification
Processor	AMD Ryzen 5 5600X, 3.7GHz, 6 cores, 12 Logical processors
Graphics Card	NVIDIA Geforce RTX 3060 12GB
RAM	16GB DDR4
Storage	512GB M.2 SSD
OS	Microsoft Windows 10 Pro
System manufacturer	Multicom

4.2.2 Optimizing the Conventional RGAE

As every proposed HAD will use the conventional RGAE for comparison and as a backbone, it needed to be optimized. The parameters this concern are the λ, K, n_hid and μ parameters. A grid search chose each parameter among values specified by the original creators [6]. These values are $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$ for the λ parameter, $[50, 100, 150, 300, 500]$ for K and $[0.1, 0.01, 0.001]$ for the learning rate μ. The number of hidden dimensions used is 100 as it was proven to be the most efficient, as stated by G. Fan et al. [6]. The number of epochs n_epochs were set to 500.

Table 4.5 show all of these optimized parameters. All parameters, with n_epochs as an exception, are tuned to each dataset using the grid search. n_epochs is not tuned as early stopping prevented the model's training from ever reaching the limit of epochs if the loss were to ever converge.

Table 4.5: Parameter settings for each dataset.

Dataset	λ	K	n_hid	μ	n_epochs
abu-airport-1	0.01	300	100	0.01	500
abu-airport-2	0.001	50	100	0.01	500
abu-airport-3	0.01	150	100	0.01	500
abu-airport-4	0.0001	150	100	0.01	500
abu-beach-1	0.0001	300	100	0.01	500
abu-beach-2	0.0001	500	100	0.01	500
abu-beach-3	0.0001	200	100	0.01	500
abu-beach-4	0.0001	150	100	0.01	500
abu-urban-1	0.001	100	100	0.01	500
abu-urban-2	0.1	150	100	0.01	500
abu-urban-3	0.01	500	100	0.001	500
abu-urban-4	0.1	50	100	0.001	500
abu-urban-5	0.01	100	100	0.01	500

4.2.3 Testing of RGAE Using Preprocessing and Modified Layer Setup

This section presents all results achieved by testing the HADs utilizing the conventional Autoencoder. First, the Preprocessing based RGAE for HAD is in focus. Table 4.6 shows the results of

the different preprocessing methods used for the network and graph input. The spectral dimensionality of the output for each preprocessing technique is also specified in the table. As mentioned in Section 3.1, the PCA and KPCA methods use the first spectral dimension as input for the image segmentation. With the RPCA methods, all spectral bands were summed into an image of two dimensions.

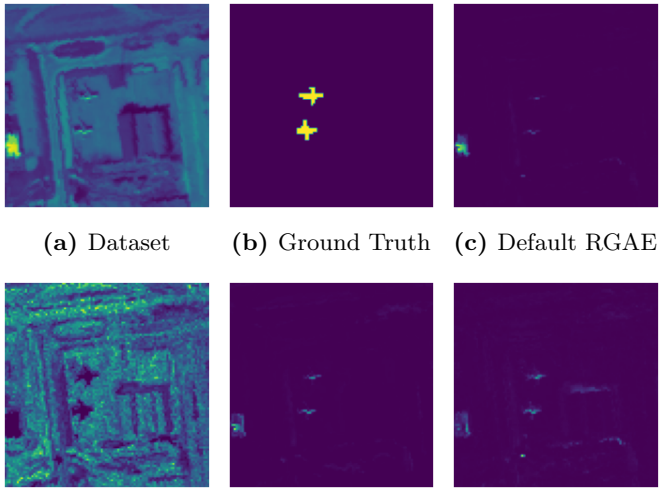
Table 4.6: AUC score for different kernels and dimensions for each dataset. The numbers in the **Method** column denotes the number of dimensions utilized.

Method	**Airport Average**	**Beach Average**	**Urban Average**	**Total Average**
Conventional RGAE	0.9237	**0.9643**	**0.9728**	**0.9551**
PCA 50	0.5796	0.9119	0.8562	0.7882
PCA 100	0.5898	0.9427	0.9089	0.8211
PCA 300	0.7458	0.7714	0.6067	0.7002
Sigmoid 50	0.9245	0.9576	0.9392	0.9403
Sigmoid 100	0.9256	0.9583	0.9451	0.9432
Sigmoid 300	**0.9319**	0.9595	0.9466	0.9460
Laplace 50	0.0521	0.2814	0.1814	0.1724
Laplace 100	0.0648	0.2509	0.2485	0.1927
Laplace 300	0.1517	0.3403	0.3726	0.2947
Exponential 50	0.1873	0.5456	0.3848	0.3735
Exponential 100	0.2797	0.5045	0.4342	0.4083
Exponential 300	0.5016	0.5874	0.6065	0.5684
Polynomial 50	0.8106	0.9483	0.8980	0.8866
Polynomial 100	0.8273	0.9458	0.8975	0.8907
Polynomial 300	0.8315	0.9426	0.8931	0.8894
RPCA Sparse Rank F	0.7859	0.9114	0.7982	0.8293
RPCA Low Rank F	0.7451	0.8832	0.6809	0.7629

As is clear from the table, the preprocessing method that worked the best was the Sigmoid kernel for the KPCA utilizing 300 spectral dimensions. However, no method was better than the default RGAE. The results from the Exponential and Laplacian kernels were registered as terrible, but an observation was made when plotting the detection maps. These two kernels invert the intensities of the anomalies and the background, causing a terrible detection performance. This was also a concern, as stated in the dataset analysis (Section 4.1.3). The PCA method also proved to result in bad performances. Using RPCA was not expected to provide good results, as stated in the dataset analysis, due to the substantial presence of noise in the sparse representation. This proved to be a correct expectation as seen from the results achieved using the RPCA preprocessing method. All preprocessing methods except for the Sigmoid KPCA utilizing 300 dimensions will therefore be excluded from the comparison of the best performing proposed HADs in Section 4.3.

Figure 4.13 shows an overview of the detection maps for some of the most successful methods of preprocessing methods utilized by the RGAE. The detection maps are all for the abu-airport-2 dataset. From the figure, it is clear that the use of the Laplacian kernel causes a bad AUC score due to the inversion of the data. This was also expressed as a potential issue in the dataset analysis. Using the Polynomial kernel, there is slightly more noise in the detection map compared to the default RGAE, but far less noise than using the Sigmoid kernel. However, using the Sigmoid kernel resulted in the best performance, likely due to the anomalies increasing in size, although at the cost of added noise.

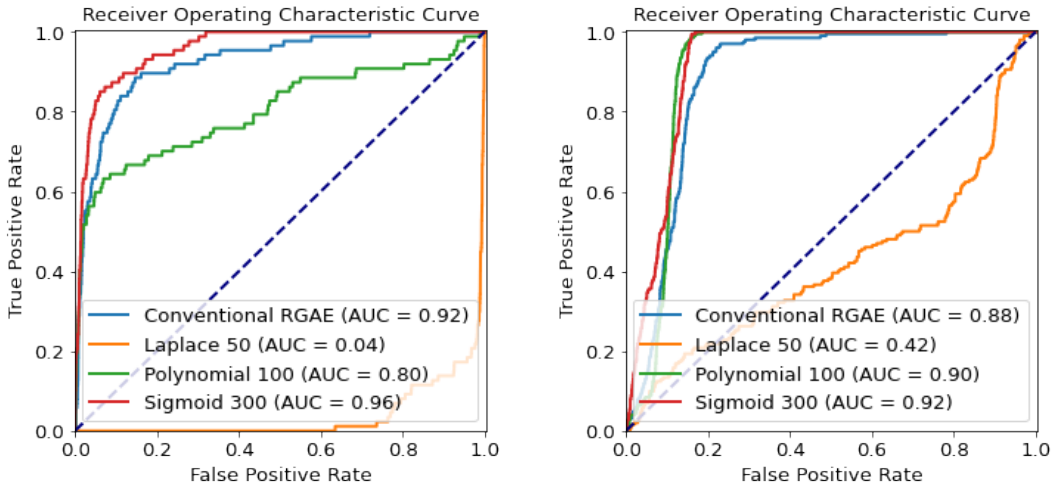
Some ROC curves for the abu-airport-2 and the abu-beach-2 datasets have been plotted and illustrated in Figure 4.14. These plots show what was evident in the overall AUC scores and the detection map, being that the Sigmoid kernel is superior to the other preprocessing methods. The ROC curve for the Sigmoid-based preprocessing HAD in Figure 4.14a is steeper than the others, indicating a higher sensitivity. It is also evident that the abu-beach-2 dataset proved to be a problematic HSI to locate the anomalies for the HADs, as predicted in the spectral analysis.



(a) Dataset **(b)** Ground Truth **(c)** Default RGAE

(d) Laplace 50 **(e)** Poly 100 **(f)** Sigmoid 300

Figure 4.13: Overview of the Detection Maps for the RGAE using different preprocessing techniques on the abu-airport-2 dataset.



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.14: Overview of the ROC curves for the RGAE using different preprocessing techniques.

Results of the tests conducted using the Multiple Kernel setup with RGAEs in parallel are shown in Table 4.7. The kernels used for the KPCA were based on what kernels proved to give meaningful results when testing the different KPCA methods as preprocessing for the conventional RGAE. The Sigmoid and Polynomial kernels were chosen as they had the highest performances, as shown in Table 4.6. The Laplacian kernel was also chosen for testing to check if the inversion of intensity could prove meaningful in a detection map fusion. This kernel is also quite different regarding the mathematical operation performed on the input compared to the Sigmoid kernel.

As seen in Table 4.7, the use of multiple KPCA kernels for two RGAEs in parallel resulted in an overall higher detection when utilizing 100 dimensions with the combination of the Sigmoid and Laplacian kernel. The conventional RGAE beat the multikernel method when utilizing all other combinations except for the MK-RGAE Sigmoid/Laplace 300 setup. Since the multikernel method using 100 dimensions (Lapl/Sigm) had the best detection performance, it will be brought for further comparison in Section 4.3.

Table 4.7: AUC score for different MK-RGAE setups. **Method** column denotes the kernel setup and the number of spectral dimensions used.

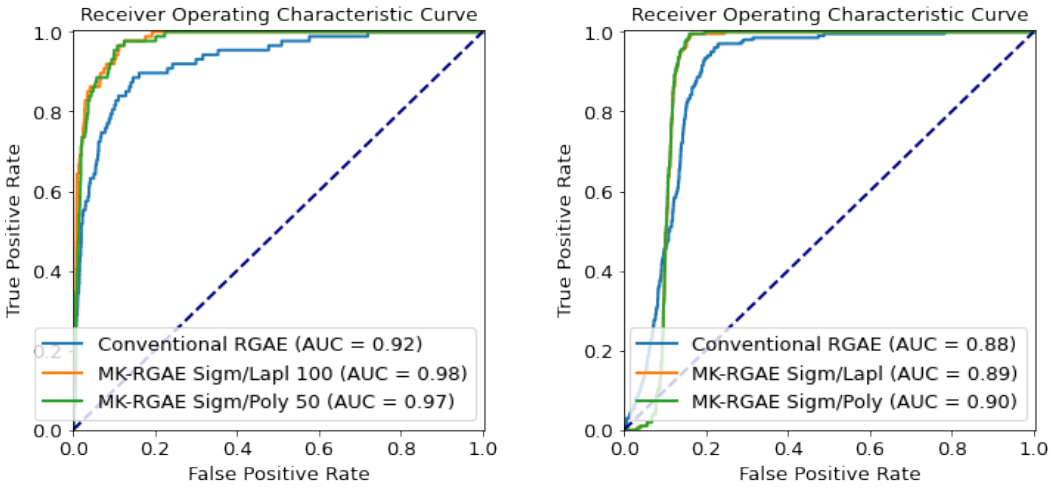
Method	Airport Average	Beach Average	Urban Average	Total Average
Conventional RGAE	0.9237	**0.9643**	0.9728	0.9551
MK-RGAE Sigmoid/Laplace 50	0.9313	0.9341	0.8918	0.9170
MK-RGAE Sigmoid/Laplace 100	0.9686	0.9510	**0.9865**	**0.9701**
MK-RGAE Sigmoid/Laplace 300	**0.9719**	0.9373	0.9691	0.9602
MK-RGAE Sigmoid/Polynomial 50	0.9299	0.9355	0.8914	0.9168
MK-RGAE Sigmoid/Polynomial 100	0.9301	0.9309	0.8943	0.9165
MK-RGAE Sigmoid/Polynomial 300	0.8481	0.9103	0.8935	0.8847

Detection maps for some of the different MK-RGAE setups are shown Figure 4.15 for the abu-airport-2 dataset. It is evident that the reason why the detection performance on the airport datasets were higher for these MK-RGAE setups is due to the more noticeable anomalies when compared to the conventional RGAE. A side effect of this is, sadly, the addition of more noise as we also saw with just using a single kernel for preprocessing. It is difficult for the human eye to determine the best detection map among the two proposed MK-RGAEs in the figure. However, the actual AUC score reveals that the MK-RGAE with the combination of the Sigmoid and Laplacian kernel is the best.

(a) Dataset **(b)** Ground Truth **(c)** Default RGAE **(d)** Sigm/Lapl 100 **(e)** Sigm/Poly 50

Figure 4.15: Overview of the Detection Maps for the best performing MK-RGAE setups on the abu-airport-2 dataset.

The ROC plots in Figure 4.16 also show that the two best performing MK-RGAEs are very similar in terms of performance on the abu-airport-2 dataset. Both of the proposed models confidently beat the conventional RGAE in terms of AUC score on the two datasets that are used for the ROC plots, although the performance gap is marginal for the beach scene. The curves in the ROC plot for the abu-beach-2 dataset also indicate a higher sensitivity for the MK-RGAEs, as their ROC curves are steeper than the curve for the conventional RGAE.



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.16: Overview of the ROC curves for the MK-RGAE setups. The MK-RGAE Sigm/Lapl uses 100 spectral dimensions whilst the MK-RGAE Sigm/Poly uses 50.

Table 4.8 shows the optimal α parameters for the decision fusion of the MK-RGAE for the best performing combinations of the two kernels. Here, the input $\mathbf{X}_1$ is the HSI with the KPCA with the Sigmoid kernel performed, and $\mathbf{X}_2$ has been subject to either the Polynomial or Laplacian kernel.

Table 4.8: Alpha parameter setting for each dataset for the MK-RGAE Sigmoid/Laplace 100 and the MK-RGAE Sigmoid/Polynomial 50.

Dataset	**MK-RGAE (Sigm/Lapl)** 100 $\alpha_{\text{MK-RGAE},1}$	**MK-RGAE (Sigm/Poly)** 50 $\alpha_{\text{MK-RGAE},2}$
abu-airport-1	0.00	1.00
abu-airport-2	0.93	1.00
abu-airport-3	0.84	0.00
abu-airport-4	0.03	0.92
abu-beach-1	0.99	0.99
abu-beach-2	0.97	0.00
abu-beach-3	0.02	1.00
abu-beach-4	1.00	0.00
abu-urban-1	0.01	0.21
abu-urban-2	0.86	0.04
abu-urban-3	0.00	0.52
abu-urban-4	0.99	0.00
abu-urban-5	0.01	0.14

The results achieved by testing the RGAE with additional layers can be seen in Table 4.9. Although it is evident from the table that the conventional RGAE method resulted in overall better detection performance, the additional layers utilizing both the ReLU/Sigmoid activation functions, as specified in Section 3.3, were better in both the airport and in the urban scenarios. The method utilizing Sigmoid/Sigmoid proved to be better than the default method in the urban scene, but not anywhere else. It is also clear that the Sigmoid/Sigmoid network failed to achieve better detection performance than the ReLU/Sigmoid version. With these results, the ReLU/Sigmoid network will be discussed and further analyzed in Section 4.3.

Table 4.9: AUC score for different layer setups. **Method** column denotes the layer setup in terms of activation functions used.

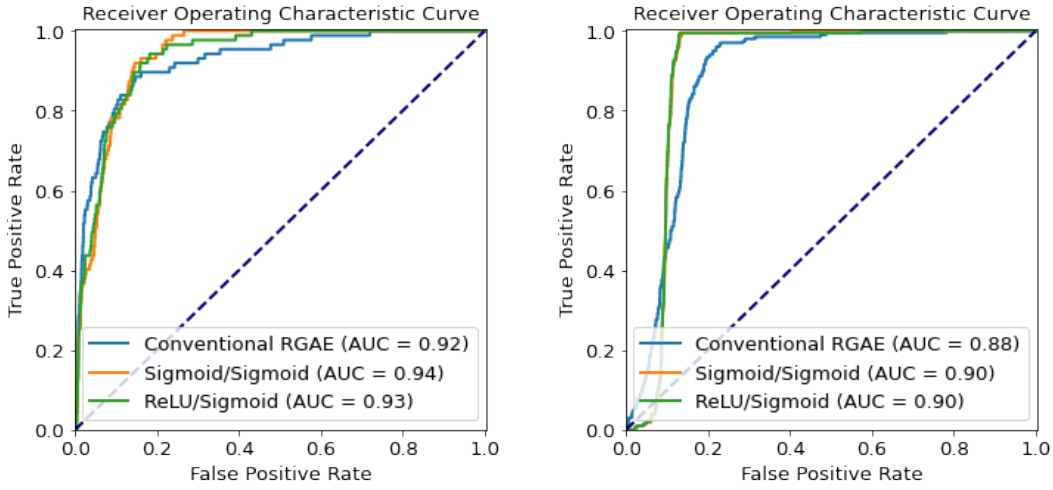
Method	Airport Average	Beach Average	Urban Average	Total Average
Conventional RGAE	0.9237	**0.9643**	0.9728	**0.9551**
RGAE Sigmoid/Sigmoid	0.9225	0.9433	0.9802	0.9511
RGAE ReLU/Sigmoid	**0.9373**	0.9399	**0.9812**	0.9550

A figure illustrating the detection maps for the different layer setups is shown in Figure 4.17. From the figures, it is possible to see that the modified layer setups have resulted in a less noisy detection map than that of the conventional RGAE except for the noisy block in the lower left corner of the dataset. One could also argue that the anomalies have slightly increased in size and intensity for the new proposed layer setups.

(a) Dataset **(b)** Ground Truth **(c)** Default RGAE **(d)** Sigm/Sigm **(e)** ReLU/Sigm

Figure 4.17: Overview of the Detection Maps for the different layer setups on the abu-airport-2 dataset.

From the ROC curves in Figure 4.18, there is an indication of a slight increase in performance for the new layer setups for both of the datasets abu-airport-2 and abu-beach-2. As with the MK-RGAE setups in Figure 4.16, the ROC curves for the proposed changes are steeper than that of the conventional RGAE. Again, this indicates a higher sensitivity.



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.18: Overview of the ROC curves for the different layer setups.

The layer dimensions in Table 4.10 show both the dimensionality of the first and second layer in the encoder and decoder. n_hid_1 and n_hid_2 are described in more detail in Section 3.3.

Table 4.10: Parameter settings for each dataset in terms of layer dimensionality.

	RGAE Sigmoid/Sigmoid		RGAE ReLU/Sigmoid	
Dataset	n_hid_1	n_hid_2	n_hid_1	n_hid_2
abu-airport-1	50	25	200	100
abu-airport-2	200	100	100	50
abu-airport-3	50	25	50	25
abu-airport-4	200	100	100	50
abu-beach-1	200	100	200	100
abu-beach-2	200	100	200	100
abu-beach-3	100	50	50	25
abu-beach-4	200	100	100	50
abu-urban-1	50	25	100	50
abu-urban-2	50	25	50	25
abu-urban-3	50	25	50	25
abu-urban-4	200	100	50	25
abu-urban-5	100	50	100	50

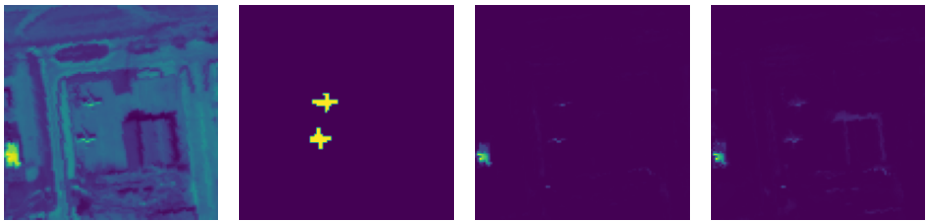
4.2.4 Testing of New Architecture

This section will show the results of all the HADs utilizing the Interpolated Autoencoder. With the Interpolated Graph Autoencoder, the detection performance achieved through testing can be seen in Table 4.11. This HAD achieved underwhelming results compared to the conventional RGAE, as seen from the table. Here, the IGAE was beaten on every dataset type in terms of average AUC scores.

Table 4.11: AUC score for the IGAE in comparison with the RGAE.

Method	**Airport Average**	**Beach Average**	**Urban Average**	**Total Average**
Conventional RGAE	**0.9237**	**0.9643**	**0.9728**	**0.9551**
Interpolated Graph Autoencoder	0.9174	0.9642	0.9708	0.9523

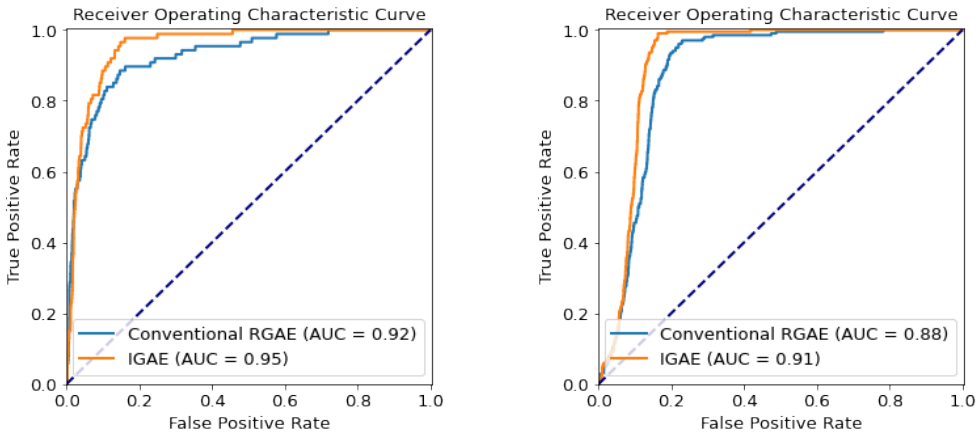
Figure 4.19 shows the detection maps for the abu-airport-2 dataset for both the conventional RGAE and the IGAE. From the detection maps, it is clear that the Interpolated Graph Autoencoder has added a lot more noise than that of the conventional RGAE. On the other hand, the anomalies themselves are more significant in size, covering more of the actual anomalous pixels.



(a) Dataset **(b)** Ground Truth **(c)** Default RGAE **(d)** IGAE

Figure 4.19: Overview of the Detection Maps for the IGAE on the abu-airport-2 dataset.

From the ROC graphs in Figure 4.20, it is possible to see that the more prominent anomalies in the detection map of the IGAE proved to result in higher detection performance. This is also the case for the abu-beach-2 dataset. On the other hand, these results are not truly representative as the average AUC scores were lower than that of the conventional RGAE.



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.20: Overview of the ROC curves for the RGAE and the IGAE.

The optimal α parameter for each dataset for the IGAE can be seen in Table 4.12. As especially seen in the beach scene, the interpolated autoencoder struggled to pick up upon any semantically meaningful data when interpolating, giving an α value of 0.

Table 4.12: Alpha parameter setting for each dataset.

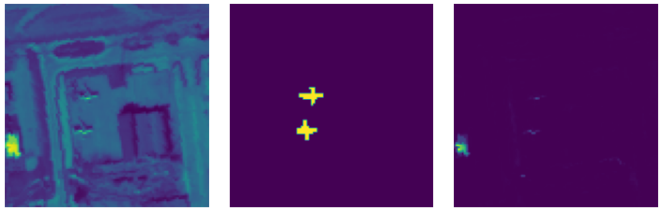
Dataset	**IGAE** α_{IGAE}
abu-airport-1	0.0
abu-airport-2	0.4
abu-airport-3	0.3
abu-airport-4	0.1
abu-beach-1	0.0
abu-beach-2	0.0
abu-beach-3	0.0
abu-beach-4	0.0
abu-urban-1	0.2
abu-urban-2	0.0
abu-urban-3	0.0
abu-urban-4	0.5
abu-urban-5	0.1

The kernels utilized for the K-IGAE were the Sigmoid kernel, as it proved to be the most efficient in terms of performance when testing the different preprocessing techniques, as well as the Polynomial and the Laplacian kernel. Table 4.13 show that this interpolation method improves the average detection performance in every scenario when utilizing the Sigmoid kernel. The use of the Laplacian and Polynomial kernels for KPCA proved to be not so effective. Using the Laplacian kernel reduced the overall average AUC score significantly, but actually increased the detection performance on the beach datasets. Using the Polynomial kernel improved the detection performance slightly from the RGAE in terms of overall average AUC score. On the other hand, it only improved upon the urban average AUC score.

Figure 4.21 shows the detection map for the three different K-IGAEs along with the conventional Robust Graph Autoencoder. As many of the other detection maps have shown, larger anomalies tend to result in a higher detection performance even with the addition of more noise. This seems to be the case again as the K-IGAE utilizing the Sigmoid kernel has a detection map containing more noise but larger anomalies, resulting in a higher AUC score. The use of the Polynomial and the Laplacian kernels does not increase the size of the anomalies as much as with the Sigmoid kernel, which is likely why the K-IGAEs utilizing the Polynomial and Laplacian kernels does not result in as good performances.

Table 4.13: AUC score for the K-IGAE in comparison with the RGAE.

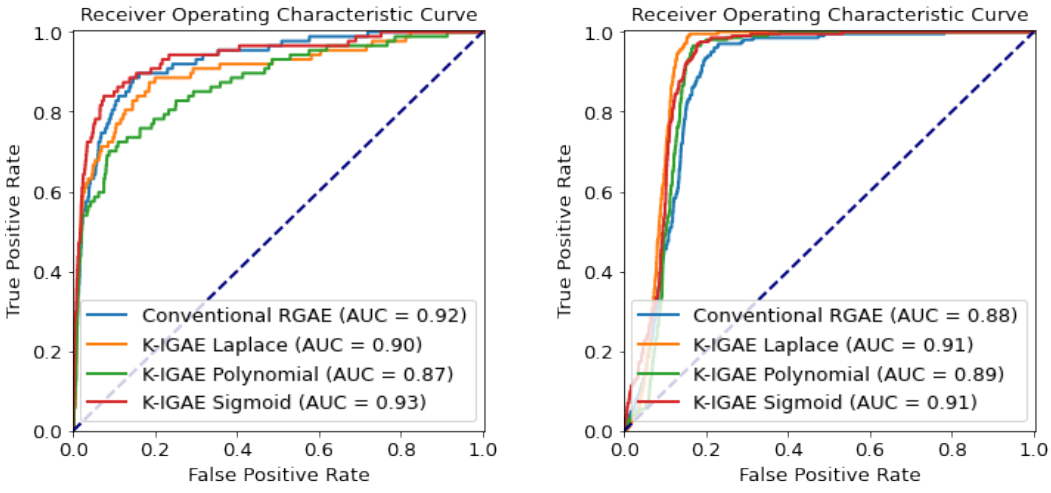
Method	Airport Average	Beach Average	Urban Average	Total Average
Conventional RGAE	0.9237	0.9643	0.9728	0.9551
Kernel Interpolated Graph Autoencoder (Sigm)	**0.9337**	0.9675	0.9834	**0.9632**
Kernel Interpolated Graph Autoencoder (Poly)	0.9121	0.9635	**0.9853**	0.9560
Kernel Interpolated Graph Autoencoder (Lapl)	0.9039	**0.9697**	0.9709	0.9499



(a) Dataset **(b)** Ground Truth **(c)** Default RGAE

(d) K-IGAE (Lapl) **(e)** K-IGAE (Poly) **(f)** K-IGAE (Sigm)

Figure 4.21: Overview of the Detection Maps for the different K-IGAE setups on the abu-airport-2 dataset..



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.22: Overview of the ROC curves for the different K-IGAE setups.

The ROC curves shown in Figure 4.22 show the same indications as seen in the detection maps. Larger anomalies, such as when using the Sigmoid kernel, result in better detection performance,

although not by a large margin in all cases. In the airport scenery, only the K-IGAE using the Sigmoid kernel beats the conventional RGAE, while it is beaten by all of the proposed K-IGAEs on the beach dataset. Each dataset's α values can be seen in Table 4.14.

Table 4.14: Alpha parameter setting for the Interpolated Autoencoder utilizing KPCA for each dataset.

Dataset	**K-IGAE (Sigm)** $\alpha_{\text{K-IGAE,1}}$	**K-IGAE (Poly)** $\alpha_{\text{K-IGAE,2}}$	**K-IGAE (Lapl)** $\alpha_{\text{K-IGAE,3}}$
abu-airport-1	0.5	0.0	0.0
abu-airport-2	0.7	0.0	0.0
abu-airport-3	0.4	1.0	0.2
abu-airport-4	0.4	0.0	0.1
abu-beach-1	0.5	0.0	0.0
abu-beach-2	0.5	1.0	0.1
abu-beach-3	0.4	0.3	0.0
abu-beach-4	0.1	0.0	0.0
abu-urban-1	0.2	0.1	0.0
abu-urban-2	0.4	0.2	0.0
abu-urban-3	0.6	0.0	0.0
abu-urban-4	0.4	0.5	0.2
abu-urban-5	0.8	0.9	0.0

As with the Multikernel RGAE, the Multikernel Interpolated Graph Autoencoder was tested using the Sigmoid kernel in combination with the Polynomial and the Laplacian kernel. The Sigmoid kernel was used in both combinations as it gave the overall best results when just performing KPCA on the input of the RGAE. Another reason for using the Sigmoid kernel for both setups is that it provided the best performance for the Kernel Interpolated Graph Autoencoder. Some results from this testing are shown in Table 4.15. These tests used 50, 100, and 300 spectral dimensions when performing the KPCA. From the table, it is clear that the MK-IGAE utilizing the Polynomial and Sigmoid kernels with 50 and 300 spectral dimensions resulted in the best performing HAD in terms of total average AUC score. The utilization of the Laplacian kernel in combination with the Sigmoid kernel proved to be a bad mix, as seen from the results. In the coming figures, the HAD utilizing the Polynomial and Sigmoid kernels with 300 dimensions will be used for comparison. This is due to it having the best detection performance on the airport datasets and the fact that it beat the conventional RGAE on the total AUC average. The MK-IGAE using the Laplacian and Sigmoid kernels with 300 spectral dimensions will also be used as it was the best-performing method using this kernel combination.

Table 4.15: AUC score for the Multikernel Interpolated Graph Autoencoder in comparison with the conventional RGAE.

Method	**Airport Average**	**Beach Average**	**Urban Average**	**Total Average**
Conventional RGAE	0.9237	**0.9643**	0.9728	0.9551
MKIGAE 300 (Sigm/Poly)	**0.9284**	0.9567	0.9785	**0.9564**
MKIGAE 100 (Sigm/Poly)	0.9247	0.9567	**0.9798**	0.9557
MKIGAE 50 (Sigm/Poly)	0.9270	0.9622	0.9753	**0.9564**
MKIGAE 300 (Sigm/Lapl)	0.9233	0.9548	0.9777	0.9539
MKIGAE 100 (Sigm/Lapl)	0.9154	0.9608	0.9770	0.9531
MKIGAE 50 (Sigm/Lapl)	0.9194	0.9656	0.9699	0.9530

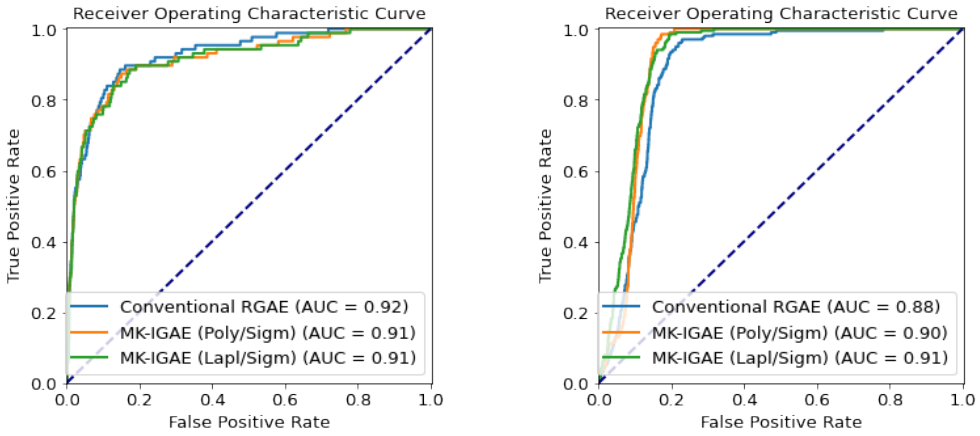
From Figure 4.23, it is evident that the detection maps for the MK-IGAEs are far noisier

than the conventional RGAE. As mentioned several times with other proposed modifications, the good side of this is that more of the anomalies have been marked as anomalous in the detection maps. This time, however, it proved overly noisy for the abu-airport-2 dataset, resulting in worse detection performance for both proposed architectures.

(a) Dataset **(b)** Ground Truth **(c)** Default RGAE **(d)** MK-IGAE 300 (Poly/Sigm) **(e)** MK-IGAE 300 (Lapl/Sigm)

Figure 4.23: Overview of the Detection Maps for the different MK-IGAE setups on the abu-airport-2 dataset.

Figure 4.24 show some of the ROC curves for the conventional RGAE and the proposed MK-IGAEs. For the datasets in question, all the HADs performances are very similar. The RGAE just about beats the proposed MK-IGAEs on the abu-airport-2 dataset while being slightly worse on the abu-beach-2 dataset. Among the two Multikernel Interpolated Graph Autoencoders, the one utilizing the Polynomial and the Sigmoid kernels proved to be best overall. On the other hand, it seemed to under-perform on these datasets compared to its rivaling MK-IGAE setup.



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.24: Overview of the ROC curves for the different MK-IGAE setups.

Table 4.16 show the optimal α parameters for the Multikernel Interpolated Graph Autoencoder utilizing the Polynomial and the Sigmoid kernels as well as the MK-IGAE utilizing the Sigmoid and Laplacian kernels. Here, $\mathbf{X}_1$ is the output when performing KPCA using the Sigmoid kernel, and $\mathbf{X}_2$ is the output when using the Polynomial or Laplacian kernel.

Table 4.16: Alpha parameter setting for the Multikernel Interpolated Graph Autoencoder.

Dataset	MK-IGAE 300 (Sigm/Poly) $\alpha_{\text{MK-IGAE},1}$	MK-IGAE 300 (Sigm/Lapl) $\alpha_{\text{MK-IGAE},2}$
abu-airport-1	0.0	0.0
abu-airport-2	0.0	0.0
abu-airport-3	0.9	0.3
abu-airport-4	0.0	0.1
abu-beach-1	0.0	0.0
abu-beach-2	0.0	0.5
abu-beach-3	0.3	0.1
abu-beach-4	0.9	0.3
abu-urban-1	0.4	0.1
abu-urban-2	0.4	0.1
abu-urban-3	0.0	0.0
abu-urban-4	0.6	0.2
abu-urban-5	0.4	0.0

4.3 Final Performance Comparison of Proposed HADs

This section will compare the different proposed HADs with each other in terms of AUC and time scores. All parameters used for each proposed HAD will be specified in Section 4.3.1. The following subsection, being Section 4.3.2, compares the detection performances in the form of AUC score for all proposed HADs. Detection maps and ROC plots are also presented to compare the different methods. Although not a focus point of the thesis, Section 4.3.3 describes the time scores of all the proposed HADs with the inclusion of the conventional RGAE. Here, scatter plots of the time score and the average AUC scores are shown to help visualize the results. Each subsection compares the preprocessing changes and the layer changes, then the architecture changes. The best performing HAD, concerning AUC score, utilizing a conventional AE will then be compared to the best performing HAD utilizing an IAE.

4.3.1 Parameters of HADs

The results presented in the coming sections are for the specified HADs in the following list. The parameters for each HAD have been optimized to a degree during the experimental testing. For a more detailed explanation of this process, please read Section 4.2 explaining the results from the experimental testing.

- RGAE using KPCA: This setup is the Preprocessing based RGAE using the conventional RGAE network parameters specified in Table 4.5. The KPCA performed on the input of the AE uses the Sigmoid kernel with 300 principal components regarding spectral bands.
- MK-RGAE: As with the RGAE using KPCA, the MK-RGAE utilizes the conventional RGAE parameters specified in Table 4.5. The kernels used are the Sigmoid and Laplacian kernels with 100 spectral dimensions. For the decision fusion, the $\alpha_{\text{MK-RGAE},1}$ parameter shown in Table 4.8 is used.
- RGAE with Additional Layers: The layer setup used with the addition of layers is the ReLU/Sigmoid setup. This implementation utilizes all of the parameters shown in Table 4.5, except for the dimensionality of the layers. For the layer dimensionality, the n_hid_1 and n_hid_2 parameters use the optimized values shown in Table 4.10.
- IGAE: The Interpolated Graph Autoencoder keeps all of the default parameters from the conventional RGAE, shown in Table 4.5, with the addition of the α_{IGAE} shown in Table 4.12.
- K-IGAE: The Kernel Interpolated Graph Autoencoder utilizes the Sigmoid kernel. As with the IGAE, all of the conventional RGAE parameters from Table 4.5 are used with the addition of the $\alpha_{\text{K-IGAE},1}$ shown in Table 4.14.

- MK-IGAE: For the Multikernel Interpolated Graph Autoencoder, the Sigmoid and Polynomial kernels are used with a spectral dimensionality of 300. The default optimized parameters for the conventional RGAE, shown in Table 4.5, are also used. Finally, the parameter used for the decision fusion is the $\alpha_{\text{MK-IGAE},1}$ presented in Table 4.16.

4.3.2 Detection Performance

This section compares the detection performances of all the proposed HADs with the inclusion of the conventional RGAE. Table 4.17 show the results of the most successful implementation among the proposed changes to the RGAE in terms of added preprocessing and changed layer setup. It is clear from the table that the use of multiple KPCA methods for preprocessing, being the MK-RGAE, resulted in a higher detection performance beating the conventional RGAE in total average. For the airport datasets, the increase in average AUC score is almost 0.05, which is an enormous improvement. The only scenario where the average of the Multikernel RGAE was worse than the default is the beach scene. Here, the overall difference is about 0.005 in AUC score. These results show that even though the two preprocessing methods do not improve detection performance when used separately, semantically meaningful data can still be derived from their combined use. This might be because the Laplacian kernel highlighted the anomalies with a clear outline, as noticed in the dataset analysis (Section 4.1.3), as well as the Sigmoid kernel increasing the anomaly size and slightly the anomaly intensity. It is however somewhat surprising that the Laplacian kernel proved to be the best combination with the Sigmoid kernel due to the inversion of spectral intensity.

Table 4.17: Results from changes to the RGAE and the addition of preprocessing compared with the conventional RGAE.

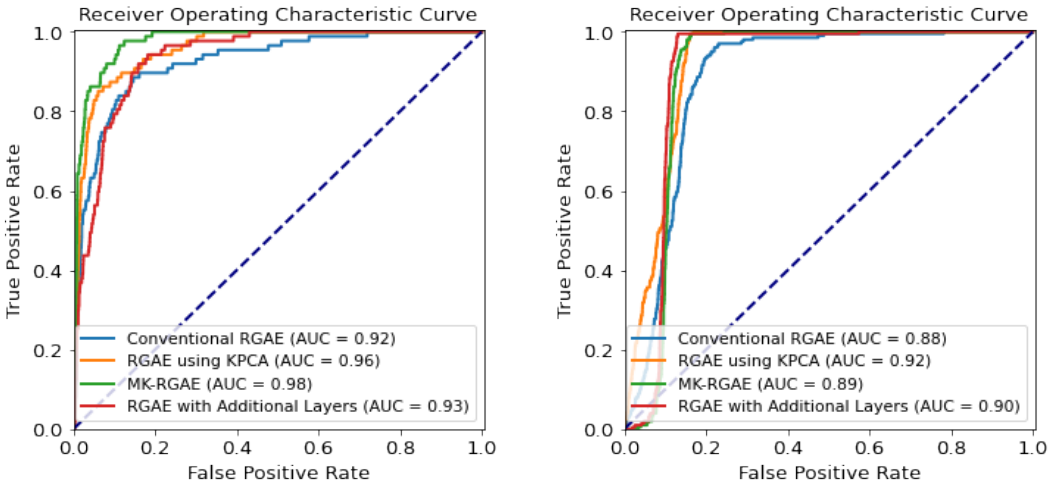
Method	Airport Average	Beach Average	Urban Average	Total Average
Conventional RGAE	0.9237	**0.9643**	0.9728	0.9551
RGAE using KPCA	0.9319	0.9595	0.9466	0.9460
MK-RGAE	**0.9686**	0.9510	**0.9865**	**0.9701**
RGAE with Additional Layers	0.9373	0.9399	0.9812	0.9550

Using one preprocessing technique, such as the RGAE with KPCA, proved to achieve disappointing results except for the airport scene. In that scenery, using a singular KPCA kernel resulted in a detection performance almost 0.01 higher in average AUC score. As already seen in Section 4.1, none of the preprocessing techniques worked that great by themselves, which could be a result of the problems such as clarity, noise, and the inversion of intensity for some of the kernels used with KPCA. Comparing this method to the MK-RGAE, it is truly disappointing.

The addition of more layers was a well-performing method, outperforming the default RGAE in the airport and urban scenario. The most significant improvement of this HAD was for the average AUC score in the airport scenery, with an increase of almost 0.014, which is a significant improvement. Conversely, it proved to be slightly worse in terms of overall detection performance due to the worse performance in the beach scenery. Compared to the MK-RGAE, the addition of layers was beaten in every scenario, with the most significant gap in AUC being for the airport datasets.

Figure 4.25 shows some ROC plots for the different preprocessing setups and the addition of layers. The datasets used for this were the abu-airport-2 and the abu-beach-2 datasets. From the curves, all of the proposed implementations seem to outperform the conventional RGAE with the MK-RGAE achieving an impressive AUC score of approximately 0.98. The other proposed implementations, such as using a singular kernel for preprocessing, are a couple of steps behind while still achieving satisfactory results. It must be noted that the AUC scores for the airport dataset is not surprising as all of these setups proved to beat the conventional RGAE in overall AUC average for the airport datasets. When looking at the ROC curves for the abu-beach-2

dataset, the MK-RGAE shows its weakness that was spotted by the average AUC score on the beach datasets. Although achieving a better AUC score than the conventional RGAE, it still underperforms compared to the less complex implementations. The use of KPCA for preprocessing shows that it can sometimes be beneficial even though it mostly underperforms compared to the conventional RGAE and the other proposed HADs.



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.25: Overview of the ROC curves for the proposed changes in preprocessing and layer setups for a set of selected datasets.

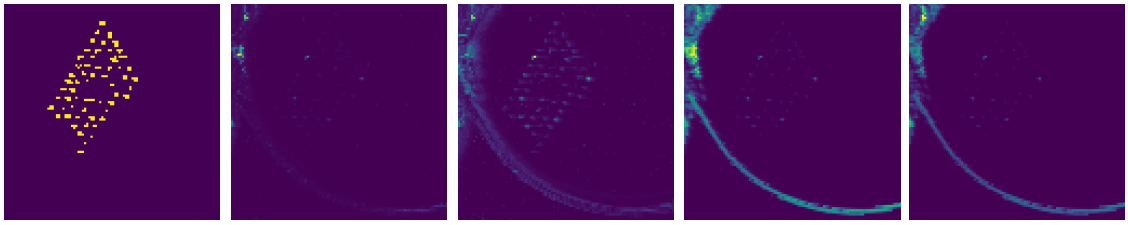
The detection maps for all of the proposed additions of preprocessing and in the layer setup for the abu-airport-2 dataset can be seen in Figure 4.26. Here, the use of KPCA shows that the anomalies increase in size whilst adding a small quantity of noise. The broader coverage of the anomalies is likely the reason for the higher detection performance. When looking at the Multikernel RGAE detection map, the anomaly coverage is much better than for the RGAE, and the intensity of the anomalies has also increased. This comes at the cost of more noise, but the higher intensity is probably the reason for the impressive AUC score seen from the ROC curve. All of this was somewhat expected as the Sigmoid kernel increased the size and intensity of the anomalies at the cost of increasing the occurrence of noise within the HSI. For the Preprocessing based RGAE utilizing the Sigmoid kernel for KPCA, we also see that the object in the lower left has decreased in intensity. There has also been an increase in the occurrence of different noise, as predicted in the dataset analysis. Finally, the addition of layers seems to have lowered the intensity of noise whilst slightly increasing the intensity of the anomalies. An exception to the decrease in noise is the noisy box in the lower left of the detection map. Among all proposed HADs, it is far easier to see the anomalies in the MK-RGAEs detection map.

(a) Ground Truth **(b)** RGAE **(c)** RGAE using KPCA **(d)** MK-RGAE **(e)** RGAE with Additional Layers

Figure 4.26: Overview of the Detection Maps for abu-airport-2 for each of the models in Table 4.17.

The detection maps for the abu-beach-2 datasets can be seen in Figure 4.27. For the RGAE utilizing KPCA and the MK-RGAE, their detection maps show similar habits as with the abu-

airport-2 dataset. This refers to the larger anomaly size with the addition of more noise. The anomalies for the MK-RGAEs detection map are very hard to locate, likely due to the Laplacian kernel inverting the intensity of the anomalies. Adding more layers increased the noise, which is the opposite of what happened with the airport dataset. On the other hand, the anomalies intensity slightly increased, which is likely the reason for the higher AUC score than that of the conventional RGAE.



(a) Ground Truth **(b)** RGAE **(c)** RGAE using KPCA **(d)** MK-RGAE **(e)** RGAE with Additional Layers

Figure 4.27: Overview of the Detection Maps for abu-beach-2 for each of the models in Table 4.17.

In Table 4.18, all of the best results from using the Interpolated Autoencoder can be seen. From the table, it is clear that the Kernel Interpolated Graph Autoencoder results in a better HAD in terms of detection performance in comparison to the conventional RGAE and the other proposed HADs. It performs the best on 6 out of the 13 datasets when all proposed architectures are considered. The most significant leap in AUC score is with the urban datasets, where the total average increased by slightly more than 0.01. In terms of overall AUC average, it increased by almost 0.01.

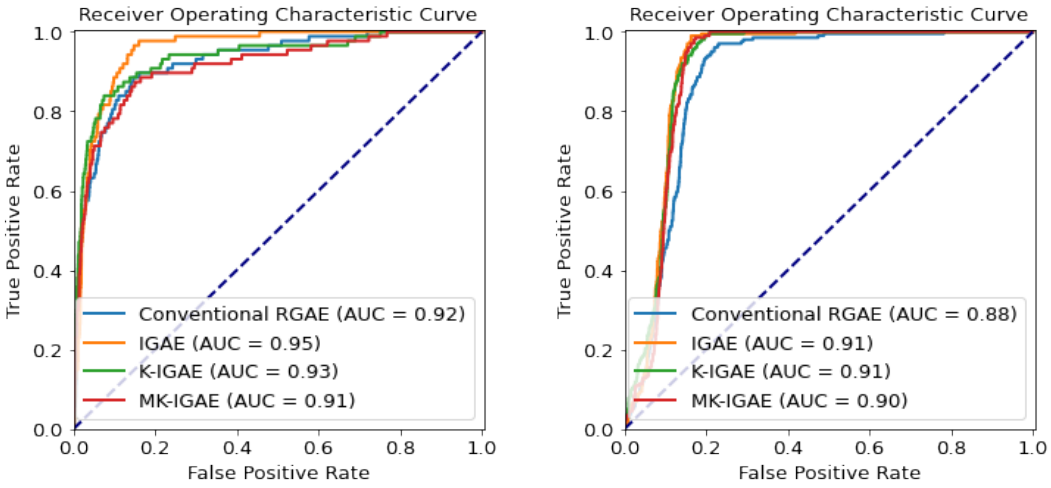
Table 4.18: Results different implementations of the Interpolated Autoencoder compared with the conventional RGAE.

Method	Airport Average	Beach Average	Urban Average	Total Average
Conventional RGAE	0.9237	0.9643	0.9728	0.9551
IGAE	0.9174	0.9642	0.9708	0.9523
K-IGAE	**0.9337**	**0.9675**	**0.9834**	**0.9632**
MK-IGAE	0.9284	0.9567	0.9785	0.9564

With the Interpolated Graph Autoencoder, the performance is slightly worse than that of the conventional RGAE. Although almost matching the conventional RGAE on the beach scene, the addition of the Interpolated Autoencoder without preprocessing proved to be underwhelming on the airport and urban datasets. The overall AUC score fell by about 0.03.

Finally, the Multikernel Interpolated Graph Autoencoder did increase the overall detection performance. The total average AUC score increased with slightly more than 0.001 compared to the conventional RGAE. While this is not a significant increase, the increase of AUC score for the airport and urban datasets is about 0.005. This came at the cost of worse performance on the beach scenery. On the other hand, the MK-IGAE underperformed compared to the slightly less complex K-IGAE, which is an unsatisfactory result.

Some ROC curves are shown in Figure 4.28, illustrating the performance difference between the different HADs utilizing the Interpolated Autoencoder. Surprisingly, given the lower average AUC score for the airport scene, the IGAE is the best-performing model on the abu-airport-2 scene, with the K-IGAE a solid step behind. However, the MK-IGAE proved to under-perform on this dataset, being beaten by the conventional RGAE. For the abu-beach-2 dataset, all of the proposed models using the Interpolated Autoencoder proved to result in a higher AUC score. The ROC curves for these models are also steeper than that of the conventional RGAE, indicating a higher sensitivity.



(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.28: Overview of the ROC curves for the proposed changes in architecture for a set of selected datasets.

The detection maps of the abu-airport-2 dataset from the proposed new architectures can be seen in Figure 4.29. As we see from the conventional RGAE, there is some noise in the bottom right, as predicted to be a problem in the dataset analysis. The anomalies are small but still possible to identify by the human eye. In terms of noise, the IGAE has slightly improved upon the issue at the cost of the anomalies being slightly lowered in intensity. Their overall size, however, has increased, which is likely the reason for the higher detection performance for that specific dataset.

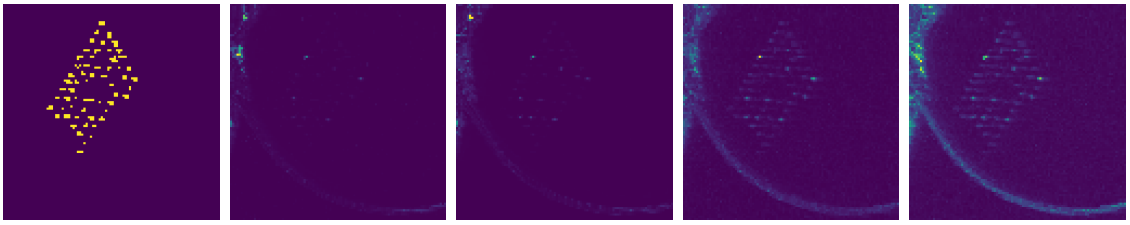
(a) Ground Truth **(b)** RGAE **(c)** IGAE **(d)** K-IGAE **(e)** MK-IGAE

Figure 4.29: Overview of the Detection Maps for abu-airport-2 for each of the models in Table 4.18.

With the Kernel Interpolated Graph Autoencoder, there is a lot of added noise. On the other hand, the anomalies have both increased in size and intensity. This most likely compensates for the added noise, leading to a higher AUC score than the conventional RGAE. It is also possible to see that the object in the lower left has decreased in intensity for the K-IGAEs detection map. As mentioned in the dataset analysis (Section 4.1.3), the Sigmoid kernel was predicted to result in such a detection map. The final detection map, being for the MK-IGAE, shows the same tendencies as with the K-IGAE, although this time not resulting in a higher detection performance than the conventional RGAE. Most likely, this is due to the addition of even more noise within the detection map than with the K-IGAE.

Figure 4.30 shows the detection maps of the abu-beach-2 dataset. As with the abu-airport-2 dataset, there is some noise in the conventional RGAE's detection map. This is again improved with the IGAE, which lowers the intensity of the noise. However, the most significant difference is that this does not come at the cost of the anomalies' intensity, as the Interpolated Graph Autoencoder highlights more of the anomalies than the RGAE. The K-IGAE highlights even more anomalies than that of the IGAE. In terms of noise, it is a fair bit worse than both the conventional RGAE and the IGAE, but the added visibility of the anomalies compensates for this resulting in

the better AUC score. The detection map for the MK-IGAE shows the same tendencies as with the abu-airport-2 dataset, being the addition of even more noise than the other HADs. Even though the Multikernel Interpolated Graph Autoencoder beats the RGAE on this dataset, the noise issue is still a substantial problem that causes it to underperform in comparison to the IGAE and the K-IGAE.



(a) Ground Truth **(b)** RGAE **(c)** IGAE **(d)** K-IGAE **(e)** MK-IGAE

Figure 4.30: Overview of the Detection Maps for abu-beach-2 for each of the models in Table 4.18.

Table 4.19 shows the best results from the addition of preprocessing to the conventional RGAE and the change of neural network architecture, being the MK-RGAE and the K-IGAE respectively. As the table show, the overall accuracy for the MK-RGAE beats the K-IGAE by approximately 0.007 in average AUC score. This is primarily due to the vast difference in the airport datasets. Here, the difference in average AUC score is almost 0.035 in favor of the MK-RGAE. With the beach datasets, however, the K-IGAE comfortably beats the MK-RGAE with a AUC score difference of about 0.017. In the urban scenery, the average AUC scores for the two proposed HADs are a lot closer, only being separated by approximately 0.003 in AUC score. On a final note, the K-IGAE is the only setup beating the conventional RGAE in every scenery, which is also something to be considered.

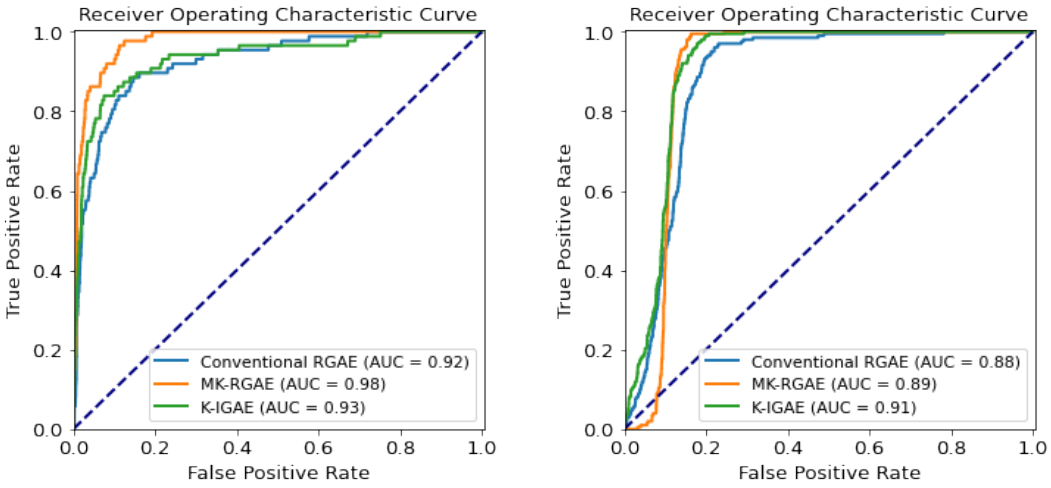
Table 4.19: Top AUC scores from the addition of preprocessing and the new proposed architecture.

Method	Airport Average	Beach Average	Urban Average	Total Average
MK-RGAE	**0.9686**	0.9510	**0.9865**	**0.9701**
K-IGAE	0.9337	**0.9675**	0.9834	0.9632

A comparison of the ROC curves for the MK-RGAE, the K-IGAE, and the RGAE can be seen in Figure 4.31. Again, the chosen datasets are the abu-airport-2 and the abu-beach-2 datasets. As also seen in Table 4.19, the MK-RGAE is far greater than the K-IGAE on the airport dataset with an AUC score difference of approximately 0.07. The ROC curve for the Multikernel RGAE also seems to be steeper, indicating slightly more sensitivity. On the beach dataset, however, this is not the case as the Kernel Interpolated Graph Autoencoder outperforms the MK-RGAE by 0.02 in terms of AUC score. Regarding steepness, the MK-RGAEs ROC curve still indicates a higher sensitivity in the beach scene.

The detection maps of the MK-RGAE and the K-IGAE for the abu-airport-2 datasets has been shown for comparison in Figure 4.32. The figures show that the detection maps of the two proposed enhancements to the RGAE have similar properties. These are broader anomaly coverage than for the conventional RGAE at the cost of added noise. The main difference between the two HADs is that the Multikernel RGAE increases the anomalies intensity significantly in comparison to both the RGAE and the K-IGAE. This is presumably the reason for the much higher AUC score for this dataset and the other airport datasets. It is also worth noting that the noisy block in the lower left of the detection maps is far less noticeable for the K-IGAE compared to the MK-RGAE.

Figure 4.33 shows the detection maps for the abu-beach-2 datset. Like seen from the previous detection maps, the MK-RGAE and the K-IGAE both increase the anomaly coverage at the cost of added noise when compared to the conventional RGAE. The most significant difference between the



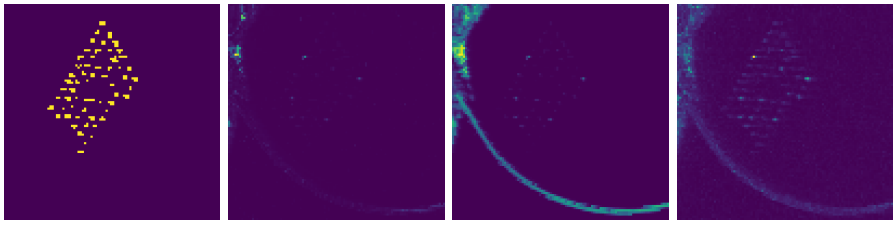
(a) abu-airport-2 **(b)** abu-beach-2

Figure 4.31: Overview of some ROC curves for the best preprocessing addition and the best architecture modification to the conventional RGAE.

(a) Ground Truth **(b)** RGAE **(c)** MK-RGAE **(d)** K-IGAE

Figure 4.32: Overview of the Detection Maps for abu-airport-2 for the best performing.

two HADs for this dataset is that the MK-RGAE struggles with highlighting all of the anomalies, like with the RGAE. In the detection map of the Kernel Interpolated Graph Autoencoder, however, practically all anomalies are highlighted. Regarding noise, the MK-RGAE is far worse than the K-IGAE. This is seen by the high intensity of the non-anomalous area within the detection map of the Multikernel RGAE. With the detection map of the K-IGAE, most of the noise has a lower intensity than the anomalies, with some exceptions. Given the noise, it is conceivable that this is why the Kernel Interpolated Graph Autoencoder outperforms the MK-RGAE on the abu-beach-2 datasets and the beach datasets in general.



(a) Ground Truth **(b)** RGAE **(c)** MK-RGAE **(d)** K-IGAE

Figure 4.33: Overview of the Detection Maps for abu-beach-2 for the best performing.

4.3.3 Time Performance

This section will compare the different proposed HADs time performances. Firstly, the time to perform the preprocessing methods will be described. Then, all HADs utilizing the conventional AE will be compared, followed up by a comparison of the HADs utilizing the IAE. In the final comparison of time scores, there will be a focus on the MK-RGAE and the K-IGAE. This is due to the two HADs demonstrating the greatest improvement in detection performance compared to the RGAE, which was the main objective of the thesis.

In terms of time performance, several implementations and changes to the conventional RGAE utilize some preprocessing. Table 4.20 is added to give an indication of this cost for the KPCA kernels used for the proposed models. These times were achieved by running KPCA with the specified kernels on all datasets except the abu-beach-1 and the abu-beach-4 since these were calculated on different machinery due to the high spatial resolution. These datasets were predicted to have a higher time cost in the dataset analysis (Section 4.1.1). As the table indicates, the appliance of the Laplacian kernel proved to be the most costly in every scenery, with the Polynomial kernel being the most efficient in terms of time score. The Sigmoid kernel was close to the Polynomial one, only a few seconds slower on average.

Table 4.20: Time scores for the use of KPCA with different kernels.

Method	Airport Average	Beach Average	Urban Average	Total Average
Sigmoid	134.5	137.9	137.3	136.4
Polynomial	**131.9**	**132.5**	**133.8**	**132.9**
Laplacian	274.8	275.7	281.9	278.2

All of the total time scores for the additions of preprocessing and changes in layer setup can be seen in Table 4.21. The preprocessing times are included for the HADs utilizing preprocessing. It is not a surprise that the inclusion of preprocessing increased the time score for the RGAE using KPCA and the MK-RGAE, as clearly seen from the table. For the MK-RGAE, the added time score is the cost of the substantial increase in detection performance. The average time cost increase for the MK-RGAE was about 2.5 times that of the conventional RGAE. It could be worse however, as the MK-RGAE only utilized 100 spectral dimensions and not 300, which would increase the time score even more. For the RGAE using KPCA, it was slightly quicker than that of the RGAE in the urban scenery, indicating a quicker convergence in terms of learning. The overall time cost, however, increased by approximately 30%.

Table 4.21: Time scores from added preprocessing and changes in layer setup compared with the conventional RGAE.

Method	Airport Average	Beach Average	Urban Average	Total Average
Conventional RGAE	**80.9**	226.3	233.3	184.2
RGAE using KPCA	222.5	287.2	205.7	235.7
MK-RGAE	454.0	515.3	489.2	486.5
RGAE with Additional Layers	98.6	**67.8**	**37.6**	**65.7**

Something slightly surprising is the fact that the RGAE with additional layers results in a lower time score than that of the conventional RGAE. Although the addition of more layers added complexity to the RGAE, the Autoencoder converged faster on average, except for the airport scenery. In total, the average time cost for the conventional RGAE is almost 3 times as large as using the modified layers. For the urban scenery, the additional layers shortened the average time

cost by almost 85%, which is astonishing. Compared to the other proposed HADs, the RGAE using a new layer setup has a clear advantage in terms of the time score.

The scatter plot in 4.34 displays the average time scores achieved for each scenery using the different setups and their corresponding average AUC scores. In this figure, the x-axis (representing the AUC score) is constrained within the range of 0.9 to 1.0. By analyzing the plot, one could argue that the addition of layers is the more efficient among the changes done to the conventional RGAE if one were to exclude the detection performance on the beach datasets. This is due to the relative detection performance increase to the time cost increase.

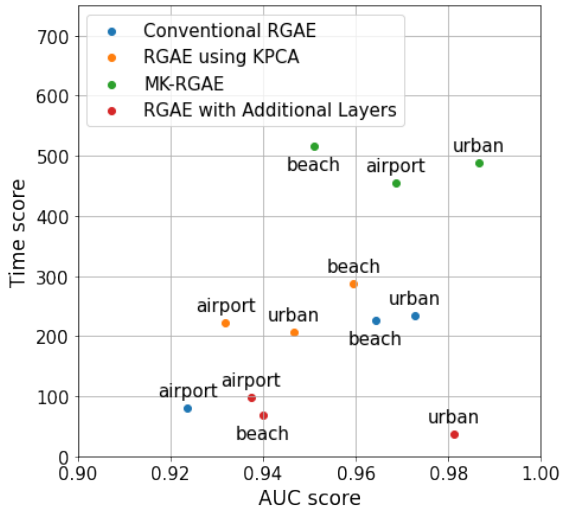


Figure 4.34: A scatter plot illustrating the time score and the AUC score for each scenery for every proposed change in either preprocessing or layer setup.

Looking at the airport scenery for the RGAE utilizing KPCA and the RGAE with additional layers, it is evident that the latter proved to be more efficient as both the time score is lower and the increase in detection performance is larger. For the RGAE utilizing KPCA, Figure 4.34 only highlights the poor performances, such as the fact that the beach average time score increased while decreasing the performance. In the MK-RGAEs case, this plot illustrates that using multiple RGAEs in parallel, although adding performance, is not the most efficient implementation if time or processing is a limiting factor. This is especially noticeable for the beach datasets.

The time scores for the changes in network architecture can be seen in Table 4.22. In terms of overall average, the conventional RGAE beat every proposed architecture change. Time time scores were terrible for the Interpolated Graph Autoencoder not even utilizing any preprocessing techniques. On the other hand, it is not that surprising considering that the amount of data being handled by the encoders is twice the amount compared to the conventional RGAE.

Table 4.22: Time scores from the different implementations of the Interpolated Autoencoder compared with the conventional RGAE.

Method	**Airport Average**	**Beach Average**	**Urban Average**	**Total Average**
Conventional RGAE	**80.9**	**226.3**	**233.3**	**184.2**
IGAE	632.7	230.0	528.9	468.8
K-IGAE	374.7	737.7	665.9	598.1
MK-IGAE	352.0	479.5	473.5	437.8

For the Kernel Interpolated Graph Autoencoder, the time score is not all that bad compared to the IGAE, considering the added preprocessing time. However, the time score on the airport datasets is a bit peculiar, as it is almost half the score of the IGAE. This is not the case for the

average time scores on the other datasets. This indicates a difference in how fast the Interpolated Autoencoder converges for the different scenery. A slight surprise is the fact that the MK-IGAE is the quickest of the proposed HADs using a new architecture even though it utilizes twice the amount of preprocessing than the K-IGAE. This indicates that the MK-IGAE converges faster than the proposed HADs that utilize the Interpolated Autoencoder.

Figure 4.35 shows a scatter plot of the time and AUC scores for the proposed HADs utilizing the Interpolated Autoencoder. The first thing to look at is the fact that the IGAE is neither efficient nor effective in terms of the detection of anomalies. This highlights that the introduction of the IAE without preprocessing is not the way to go if one wishes for a better performing Robust Graph Autoencoder. When looking at the beach scenery for the K-IGAE, it is evident that this method is not the most efficient either. On the other hand, it still increases the AUC score and is more efficient for the airport and urban datasets. The Multikernel Interpolated Graph Autoencoder is not as efficient, nor as effective, as the K-IGAE in the airport scenery. In terms of the urban datasets, the increase in AUC score compared to the increase in time score is very similar between the two mentioned HADs.

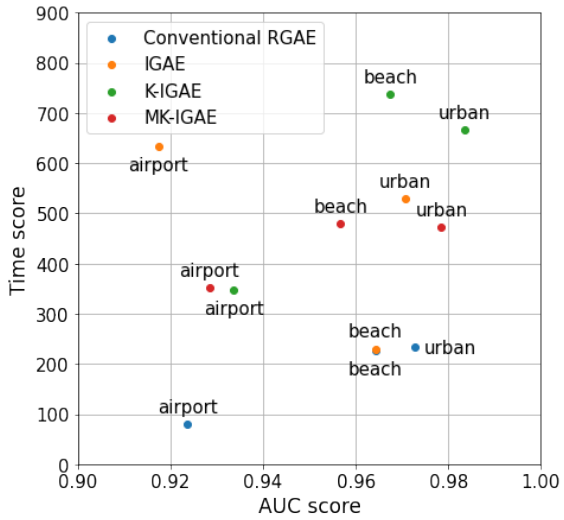


Figure 4.35: A scatter plot illustrating the time score and the AUC score for each scenery for every proposed change in neural network architecture.

The time score for the two best performing HADs based on AUC score, being the MK-RGAE and the K-IGAE, are shown in Table 4.23. Neither of these HADs improves upon the time score of the conventional RGAE. On total average, the MK-RGAE is about 100 seconds quicker than that of the K-IGAE. As seen from the table, the Kernel Interpolated Graph Autoencoder is considerably slower than the MK-RGAE in the beach and urban scenery while being slightly quicker on the airport scenery. It is, however, worth noting that the purpose of this thesis is not to lower the time cost, but to improve upon the detection performance of the RGAE. That being said, the MK-RGAE still beats the K-IGAE in every scenario except for the time score of the airport scenery and the AUC score on the beach scenery.

Table 4.23: Time scores from the best performing preprocessing implementation and the best performing implementation of the Interpolated Autoencoder based on AUC score.

Method	Airport Average	Beach Average	Urban Average	Total Average
MK-RGAE	454.0	**515.3**	**489.2**	**486.5**
K-IGAE	**374.7**	737.7	665.9	598.1

The scatter plot in Figure 4.36 shows the time scores against the AUC scores for the MK-RGAE

and the K-IGAE, with the inclusion of the conventional RGAE. Focusing on the airport scenery, the use of the K-IGAE compared to the MK-RGAE costs more in terms of increased time to increased AUC score. The same can be said for the urban scenery. The only true advantage the K-IGAE has to that of the MK-RGAE is the fact that it can achieve a higher AUC score on the beach datasets. On the other hand, the increased time is far higher than for the MK-RGAE, making this advantage questionable. Therefore, the question of what method is best boils down to what scenery is of the question for the user and the processing power available.

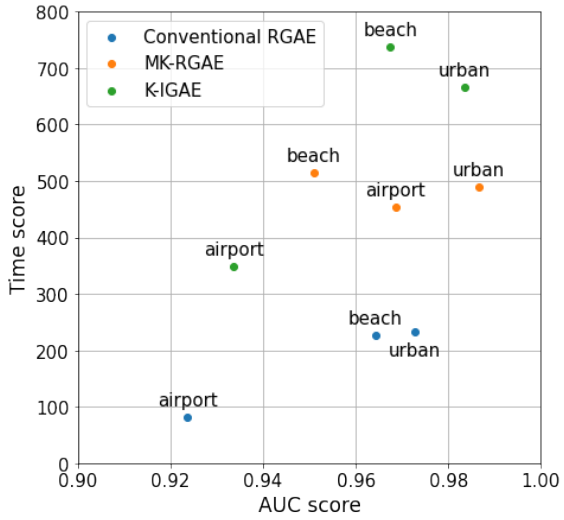


Figure 4.36: A scatter plot illustrating the time score and the AUC score for the best proposed models.

Chapter 5

Conclusions and Future Directions

This chapter provides an overview of the most important findings of the Results chapter (Chapter 4). The summary and conclusion of the entire thesis will be presented in Section 5.1. Following up on the summary and conclusion, there will be presented several topics that needs to be further investigated. These topics focus on new areas that could uncover a performance boost for some of the proposed HADs, as well as making the work of the thesis more relevant for the HYPSO mission.

5.1 Summary of Findings and Conclusion

The objective of this Master's thesis was to improve upon a preexisting state-of-the-art HAD to help the HYPSO mission to detect anomalous objects and occurrences. In this thesis, various different HADs have been proposed, which all use the Robust Graph Autoencoder as a backbone. These HADs have brought the additions of preprocessing, layer setups, and new neural network architectures. Preprocessing methods that have been tested are the PCA, KPCA, and RPCA using various number of spectral dimensions. Among the HADs utilizing these changes were the preprocessing based RGAE and the MK-RGAE. Adding more layers to the encoder and decoder was also tested. The new architecture implemented to replace the Autoencoder was the Interpolated Autoencoder. This neural network was implemented for the IGAE, K-IGAE, and the MK-IGAE where the last two mentioned HADs utilized preprocessing.

From the comparison of the proposed Hyperspectral Anomaly Detectors, it is evident that there were two proposed HADs that outshine the others. These were the Multikernel RGAE and the Kernel Interpolated Graph Autoencoder. The two HADs achieved the goal of enhancing the preexisting Robust Graph Autoencoder as their total average AUC score increased significantly. For the MK-RGAE, a total average AUC score of 0.9701 was achieved, beating the RGAE, wich had a score of 0.9551. Although not beating the MK-RGAE, the K-IGAE achieved an impressive AUC score of 0.9632 in total average. On the other hand, the K-IGAE had the upper hand on the beach datasets. It is also worth noting that both of the these proposed HADs outperformed the RGAE on the airport and urban datasets, while only the K-IGAE improved upon the detection performance for the beach datasets. A similarity between these two HADs in terms of the detection maps was that the anomaly coverage was larger and the intensities of the anomalies were stronger than that of the conventional RGAE. This came at the cost of added noise in both of the proposed HADs' detection maps.

The other HADs, being the Preprocessing based RGAE, the RGAE with modified layers, the IGAE and the MK-IGAE, showed disappointing results. None of the methods, except for the MK-IGAE, proved to result in overall higher detection performance. However, since the MK-IGAE added such a slight increase in performance at such high computational cost, it is not considered an excellent upgrade to the preexisting RGAE. An exception to the bad results from these HADs is the time score of RGAE with additional layers. This time score is considered substantially lower than that of the other proposed HADs and the conventional RGAE, making it a viable implementation to reduce time cost.

Since the primary objective of the thesis revolved around enhancing the detection performance

of the Robust Graph Autoencoder, several of the proposed HADs failed to achieve the wanted results. Given that the MK-RGAE and the K-IGAE emerged as the methodologies that genuinely accomplished the objective of the thesis, it is inherent that these approaches warrant further refinement and advancement. Adding preprocessing and utilizing the Interpolated Autoencoder is, therefore, viable methods to increase the detection performance of the Robust Graph Autoencoder at the expense of added time cost.

5.2 Future Work

This section will go through the future work of the thesis regarding areas such as new configurations of the neural network (Section 5.2.1), optimization (Section 5.2.2), and testing on other datasets more relevant to the HYPSO project (Section 5.2.3).

5.2.1 New Configurations

As we saw from the results in Chapter 4, the modification of layer setup proved to increase detection performance in the airport and urban scenery and reduce the time cost drastically in terms of total average on all datasets. Implementing that layer setup with other HADs, such as with the Multikernel RGAE, could further boost detection performance. This could also be done for the Kernel Interpolated Graph Autoencoder. An exploration of new and more complex layer setups could also be performed to investigate if such changes could lead to better detection performance or a lower time score.

Although highly computationally heavy, checking the use of multiple K-IGAEs in parallel using different preprocessing techniques, such as with the MK-RGAE, could also be explored. This would likely achieve time scores around double that of the standard K-IGAE, but it would be interesting to see if such a setup would obtain a higher detection performance than that of the MK-RGAE.

Utilizing optimizers is a well-known method to boost neural networks learning and convergence speed, as explained in Section 2.7.1. Optimizers such as the Alternating Direction Method of Multipliers (ADMM) optimizer could potentially further boost an Autoencoders performance and should therefore be considered for further investigation [59].

5.2.2 Optimization

In terms of optimization, several topics should be investigated further. One of these is the spectral dimensionality of and preprocessing methods output. For the optimization in this thesis, the dimensionalities were only tested using 50, 100 and 300 spectral bands. A more detailed range of dimensionalities should therefore be tested to solidify the results achieved in this thesis.

Ideally, the parameters optimized for the conventional RGAE and used by other modifications to the HAD should be optimized again for the new setups. In the case of the RGAE using a new layer setup, there is still much optimization to do. For simplicity's sake, the n_hid_2 was chosen to be half the size of n_hid_1. Other relationships could be tested, such as n_hid_2 being 3/4 the size of n_hid_1. The values chosen for the grid search of n_hid_1 were amongst the following values [50, 100, 200]. These values can be increased to the same list of values used for the dimensionality of the conventional RGAE's parameter n_hid. If this improves the performance, it can be further tested on the MK-RGAE and the K-IGAE.

Finally, the decision fusion variables, being the $\alpha_{...}$ parameters, for the HADs utilizing the Interpolated Autoencoder could be tested for the same range as with the similar parameter for the MK-RGAE. To do this, though, using a powerful computer is imperative as HADs such as the K-IGAE needs to be trained for each of the following parameters.

5.2.3 Datasets

To check whether the MK-RGAE or the K-IGAE works well for the HYPSO satellite, it would be beneficial to test the HADs on some of the HSIs taken by the satellite. To do this, a couple of requirements need to be fulfilled. The first requirement is that the data must be labeled, which can be a tedious task. A powerful computer would also be needed to handle the extensive data for the Hyperspectral Images taken by the HYPSO satellite as the spatial resolution is much larger

than that of the ABU datasets. A different approach could be to segment each HSI into smaller images containing less data.

References

[1] Pedram Ghamisi et al. 'Advances in Hyperspectral Image and Signal Processing: A Comprehensive Overview of the State of the Art'. In: *IEEE Geoscience and Remote Sensing Magazine* 5.4 (2017), pp. 37–78. DOI: 10.1109/MGRS.2017.2762087.

[2] Hongjun Su et al. 'Hyperspectral Anomaly Detection: A survey'. In: *IEEE Geoscience and Remote Sensing Magazine* 10.1 (2022), pp. 64–90. DOI: 10.1109/MGRS.2021.3105440.

[3] Stefania Matteoli, Marco Diani and James Theiler. 'An Overview of Background Modeling for Detection of Targets and Anomalies in Hyperspectral Remotely Sensed Imagery'. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.6 (2014), pp. 2317–2336. DOI: 10.1109/JSTARS.2014.2315772.

[4] *HYPSO (HYPerspectral Smallsat for Ocean Observation)*. 2023. URL: https://www.eoportal.org/satellite-missions/hypso#eop-quick-facts-section.

[5] Xing Hu et al. 'Hyperspectral Anomaly Detection Using Deep Learning: A Review'. In: *Remote Sensing* 14.9 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14091973.

[6] Ganghui Fan et al. 'Hyperspectral Anomaly Detection With Robust Graph Autoencoders'. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–14. DOI: 10.1109/TGRS.2021.3097097.

[7] Md. Palash Uddin, Md. Al Mamun and Md. Ali Hossain. 'PCA-based Feature Reduction for Hyperspectral Remote Sensing Image Classification'. In: *IETE Technical Review* 38.4 (2021), pp. 377–396. DOI: 10.1080/02564602.2020.1740615.

[8] David Berthelot et al. *Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer*. 2018. arXiv: 1807.07543 [cs.LG].

[9] *Data sets*. URL: http://xudongkang.weebly.com/data-sets.html (visited on 23rd Jan. 2023).

[10] Brage Samsonsen. *Anomaly Detection in Hyperspectral Images Using Machine Learning*. Tech. rep. Trondheim: Norwegian University of Science and Technology, Dec. 2022.

[11] Pedram Ghamisi et al. 'New Frontiers in Spectral-Spatial Hyperspectral Image Classification: The Latest Advances Based on Mathematical Morphology, Markov Random Fields, Segmentation, Sparse Representation, and Deep Learning'. In: *IEEE Geoscience and Remote Sensing Magazine* 6.3 (2018), pp. 10–43. DOI: 10.1109/MGRS.2018.2854840.

[12] Gamal ElMasry and Da-Wen Sun. 'Principles of hyperspectral imaging technology'. In: *Hyperspectral imaging for food quality analysis and control*. Elsevier, 2010, pp. 3–43.

[13] Bing Lu et al. 'Recent Advances of Hyperspectral Imaging Technology and Applications in Agriculture'. In: *Remote Sensing* 12.16 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12162659.

[14] Mihaela Antonina Calin et al. 'Hyperspectral Imaging in the Medical Field: Present and Future'. In: *Applied Spectroscopy Reviews* 49.6 (2014), pp. 435–447. DOI: 10.1080/05704928.2013.838678.

[15] X. Briottet et al. 'Military applications of hyperspectral imagery'. In: *Targets and Backgrounds XII: Characterization and Representation*. Ed. by Wendell R. Watkins and Dieter Clement. Vol. 6239. International Society for Optics and Photonics. SPIE, 2006, 62390B. DOI: 10.1117/12.672030.

[16] James B. Campbell and Randolph H. Wynne. *Introduction to Remote Sensing, Fifth Edition*. en. Guilford Press, June 2011. ISBN: 978-1-60918-177-2.

[17] *ASI — Agenzia Spaziale Italiana*. URL: https://www.asi.it/en/earth-science/prisma/ (visited on 25th Apr. 2023).

[18] *EnMAP*. URL: https://www.enmap.org/?fbclid=IwAR2S7Z4BcdbuDofH5NjpnfVY3ehJ3T_12vHLL-zc5Z1sSuXnAro-oaDLBNw (visited on 25th Apr. 2023).

[19] Robert O Green et al. 'Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)'. In: *Remote sensing of environment* 65.3 (1998), pp. 227–248.

[20] Ferdi Andika, Mia Rizkinia and Masahiro Okuda. 'A Hyperspectral Anomaly Detection Algorithm Based on Morphological Profile and Attribute Filter with Band Selection and Automatic Determination of Maximum Area'. In: *Remote Sensing* 12.20 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12203387.

[21] Shaoyu Wang et al. 'Auto-AD: Autonomous Hyperspectral Anomaly Detection Network Based on Fully Convolutional Autoencoder'. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–14. DOI: 10.1109/TGRS.2021.3057721.

[22] Andrew P. Bradley. 'The use of the area under the ROC curve in the evaluation of machine learning algorithms'. In: *Pattern Recognition* 30.7 (1997), pp. 1145–1159. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(96)00142-2.

[23] José Manuel Amigo, Hamid Babamoradi and Saioa Elcoroaristizabal. 'Hyperspectral image analysis. A tutorial'. In: *Analytica Chimica Acta* 896 (2015), pp. 34–51. ISSN: 0003-2670. DOI: https://doi.org/10.1016/j.aca.2015.09.030.

[24] Radhakrishna Achanta et al. 'SLIC Superpixels Compared to State-of-the-Art Superpixel Methods'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/TPAMI.2012.120.

[25] Jonathon Shlens. 'A Tutorial on Principal Component Analysis'. In: *CoRR* abs/1404.1100 (2014). arXiv: 1404.1100. URL: http://arxiv.org/abs/1404.1100.

[26] Ganesh R Naik. *Advances in principal component analysis: research and development*. Springer, 2017.

[27] René Vidal, Yi Ma and S. Shankar Sastry. 'Principal Component Analysis'. In: *Generalized Principal Component Analysis*. New York, NY: Springer New York, 2016, pp. 25–62. ISBN: 978-0-387-87811-9. DOI: 10.1007/978-0-387-87811-9_2.

[28] L.J. Cao et al. 'A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine'. In: *Neurocomputing* 55.1 (2003). Support Vector Machines, pp. 321–336. ISSN: 0925-2312. DOI: https://doi.org/10.1016/S0925-2312(03)00433-8.

[29] Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller. 'Kernel principal component analysis'. In: *Artificial Neural Networks — ICANN'97*. Ed. by Wulfram Gerstner et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 583–588. ISBN: 978-3-540-69620-9.

[30] Yen-Yu Lin, Tyng-Luh Liu and Chiou-Shann Fuh. 'Multiple Kernel Learning for Dimensionality Reduction'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.6 (2011), pp. 1147–1160. DOI: 10.1109/TPAMI.2010.183.

[31] Lingxia Mu, Biyu Lei and Ding Liu. 'A Multi-Kernel Principal Component Analysis Method for Quality-Related Fault Detection'. In: *2022 37th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. 2022, pp. 1–5. DOI: 10.1109/YAC57282.2022.10023777.

[32] Emmanuel J. Candès et al. 'Robust Principal Component Analysis?' In: *J. ACM* 58.3 (June 2011). ISSN: 0004-5411. DOI: 10.1145/1970392.1970395.

[33] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[34] Sefa Küçük and Seniha Esen Yüksel. 'Comparison of RX-based anomaly detectors on synthetic and real hyperspectral data'. In: *2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. 2015, pp. 1–4. DOI: 10.1109/WHISPERS.2015.8075504.

[35] Bing Tu et al. 'Hyperspectral anomaly detection via density peak clustering'. In: *Pattern Recognition Letters* 129 (2020), pp. 144–149. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2019.11.022.

[36] Patrick C. Hytla et al. 'Anomaly detection in hyperspectral imagery: comparison of methods using diurnal and seasonal data'. In: *Journal of Applied Remote Sensing* 3.1 (2009), p. 033546. DOI: 10.1117/1.3236689.

[37] A. Banerjee, P. Burlina and C. Diehl. 'A support vector method for anomaly detection in hyperspectral imagery'. In: *IEEE Transactions on Geoscience and Remote Sensing* 44.8 (2006), pp. 2282–2291. DOI: 10.1109/TGRS.2006.873019.

[38] Prudhvi Gurram and Heesung Kwon. 'Support-Vector-Based Hyperspectral Anomaly Detection Using Optimized Kernel Parameters'. In: *IEEE Geoscience and Remote Sensing Letters* 8.6 (2011), pp. 1060–1064. DOI: 10.1109/LGRS.2011.2155030.

[39] Xiyou Fu et al. 'Hyperspectral Anomaly Detection via Deep Plug-and-Play Denoising CNN Regularization'. In: *IEEE Transactions on Geoscience and Remote Sensing* 59.11 (2021), pp. 9553–9568. DOI: 10.1109/TGRS.2021.3049224.

[40] Susmita Ray. 'A Quick Review of Machine Learning Algorithms'. In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019, pp. 35–39. DOI: 10.1109/COMITCon.2019.8862451.

[41] Pai-Hsuen Chen, Chih-Jen Lin and Bernhard Schölkopf. 'A tutorial on ν-support vector machines'. In: *Applied Stochastic Models in Business and Industry* 21.2 (2005), pp. 111–136.

[42] Li Han. 'Using a Dynamic K-means Algorithm to Detect Anomaly Activities'. In: *2011 Seventh International Conference on Computational Intelligence and Security*. 2011, pp. 1049–1052. DOI: 10.1109/CIS.2011.233.

[43] Moisés F. Lima et al. 'Anomaly detection using baseline and K-means clustering'. In: *SoftCOM 2010, 18th International Conference on Software, Telecommunications and Computer Networks*. 2010, pp. 305–309.

[44] Aristidis Likas, Nikos Vlassis and Jakob J. Verbeek. 'The global k-means clustering algorithm'. In: *Pattern Recognition* 36.2 (2003). Biometrics, pp. 451–461. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(02)00060-2.

[45] Ben Krose and Patrick van der Smagt. *An introduction to neural networks*. The University of Amsterdam, 1996.

[46] Sagar Sharma, Simone Sharma and Anidhya Athaiya. 'Activation functions in neural networks'. In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[47] Qi Wang et al. 'A comprehensive survey of loss functions in machine learning'. In: *Annals of Data Science* (2020), pp. 1–26.

[48] Usha Ruby and Vamsidhar Yendapalli. 'Binary cross entropy with deep learning technique for image classification'. In: *Int. J. Adv. Trends Comput. Sci. Eng* 9.10 (2020).

[49] Otto Fabius and Joost R. van Amersfoort. *Variational Recurrent Auto-Encoders*. 2015. DOI: 10.48550/arXiv.1412.6581. arXiv: 1412.6581 [stat.ML].

[50] H Jabbar and Rafiqul Zaman Khan. 'Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)'. In: *Computer Science, Communication and Instrumentation Devices* 70 (2015), pp. 163–172.

[51] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. DOI: 10.48550/arXiv.1609.04747. arXiv: 1609.04747 [cs.LG].

[52] Dor Bank, Noam Koenigstein and Raja Giryes. *Autoencoders*. 2021. DOI: 10.48550/arXiv.2003.05991. arXiv: 2003.05991 [cs.LG].

[53] Ganghui Fan et al. 'Robust Graph Autoencoder for Hyperspectral Anomaly Detection'. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 1830–1834. DOI: 10.1109/ICASSP39728.2021.9414767.

[54] Xiangyu Song et al. 'Spectral–Spatial Anomaly Detection of Hyperspectral Data Based on Improved Isolation Forest'. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–16. DOI: 10.1109/TGRS.2021.3104998.

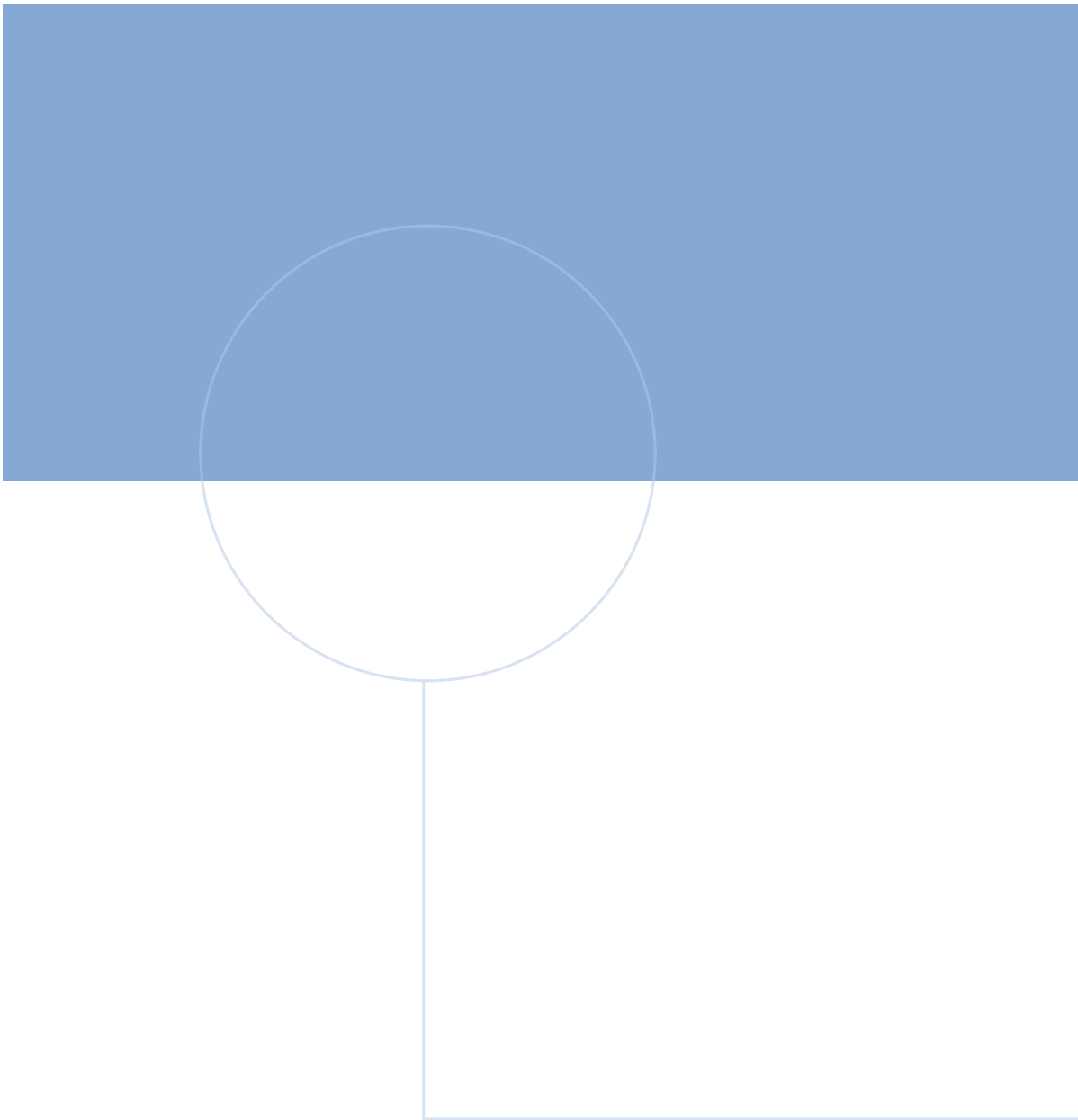
[55] *scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation*. URL: https://scikit-learn.org/stable/?fbclid=IwAR1UJEF15G7twNAA1jz17fpG0Jq5L6dlkufO_CZnCwNSO5fBXpidHlEgv28 (visited on 11th May 2023).

[56] Jürgen Schmidhuber. 'Deep learning in neural networks: An overview'. In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003.

[57] *PyTorch*. en. URL: https://www.pytorch.org (visited on 11th May 2023).

[58] Ferdi Andika, Mia Rizkinia and Masahiro Okuda. 'A Hyperspectral Anomaly Detection Algorithm Based on Morphological Profile and Attribute Filter with Band Selection and Automatic Determination of Maximum Area'. In: *Remote Sensing* 12 (Oct. 2020), p. 3387. DOI: 10.3390/rs12203387.

[59] Shaokai Ye et al. *Progressive Weight Pruning of Deep Neural Networks using ADMM*. 2018. arXiv: 1810.07378 [cs.LG].



NTNU

Norwegian University of
Science and Technology