Master's thesis

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

Katinka Müller

Optimized Hyperspectral Anomaly Detectors: Improved Performance and Reduced Computational Complexity

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

NTNU
Norwegian University of
Science and Technology

Katinka Müller

Optimized Hyperspectral Anomaly Detectors: Improved Performance and Reduced Computational Complexity

Master's thesis in Electronics Systems 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

Hyperspectral Anomaly Detection (HAD) plays a crucial role in various remote sensing applications, particularly in environmental monitoring. This thesis focuses on developing an HAD model for potential integration into the onboard processing of the Hyper-Spectral Small Satellite for Ocean Observations (HYPSO), designed to detect irregularities in the ocean, such as algae bloom, phytoplankton, and river plumes. The objective is to design a model that achieves high detection performance with regard to other state-of-the-art models while minimizing computational cost to ensure efficient power usage. Three contributions are presented to enhance the Autonomous Hyperspectral Anomaly Detection Network (AUTO-AD), a state-of-the-art Hyperspectral Anomaly Detection (HAD) model. Firstly, AUTO-AD$^+$ incorporates Kernel Principal Component Analysis (KPCA)-based pre-processing to improve detection performance. Secondly, the Lightweight AUTO-AD (LW-AUTO-AD) model proposes a lightweight Autoencoder (AE) architecture to reduce computational cost. Lastly, LW-AUTO-AD$^+$ combines LW-AUTO-AD with a pre-processing block to enhance detection performance and reduce computational cost. Experimental evaluations were conducted on the Airport-Beach-Urban (ABU) dataset, comparing the proposed models with AUTO-AD and other state-of-the-art approaches. The findings revealed that all three configurations effectively reduced the computational cost. Moreover, the results showed that incorporating pre-processing techniques significantly enhanced the detection performance. Overall, LW-AUTO-AD$^+$ demonstrated the highest detection performance among all proposed models, including several state-of-the-art models.

Sammendrag

Hyperspektral anomalideteksjon (HAD) spiller en avgjørende rolle i ulike fjernmålingsapplikasjoner, inkludert miljøovervåking. Denne masteren fokuserer på å utvikle en HAD-modell som kan integreres i prosesseringen om bord på den hyperspektrale småsatellitten for havobservasjoner (HYPSO), som er designet for å oppdage uregelmessigheter i havet, som for eksempel planteplankton og alge oppblomstring. Målet er å designe en modell som oppnår høy deteksjonsytelse sammenlignet med andre topp modeller, samtidig som beregningskostnadene minimeres. Denne masteren presenterer tre bidrag for å forbedre Autonomous Hyperspectral Anomaly Detection Network (AUTO-AD) modellen. Det første bidraget, AUTO-AD$^+$, kombinerer AUTO-AD med en preprosessering metode basert på kernel principal component analysis (KPCA) for å forbedre deteksjonsytelsen. Det andre bidraget er en simplifisert AUTO-AD-modellen (LW-AUTO-AD), som inkorporerer en simplifisert auto enkoder-arkitektur for å redusere beregningskostnadene. Til slutt kombinerer LW-AUTO-AD$^+$, LW-AUTO-AD modellen med en preprosesserings blokk for å forbedre deteksjonsytelsen og redusere beregningskostnadene. Eksperimentelle evalueringer ble utført på Airport-Beach-Urban (ABU) datasettet, der modellene ble sammenlignet med AUTO-AD og andre topp moderne modeller. Funnene viste at alle de tre konfigurasjonene effektivt reduserte beregningskostnadene. Videre viste resultatene at preprosessering forbedret deteksjonsytelsen. Totalt sett viste LW-AUTO-AD$^+$ den høyeste deteksjonsytelsen blant alle foreslåtte modeller, inkludert flere av de topp moderne modellene.

Preface

This master's thesis was conducted at the Norwegian University of Science and Technology (NTNU) for the institute of Electronics Systems (IES). I would like to thank my supervisor Milica Orlandić for the assistance and motivation throughout the master thesis. Her expertise, guidance, and insightful feedback have been instrumental in shaping the direction and quality of this thesis. I would also like to thank my co-supervisor Vinay Chakravarthi Gogineni for the technical advice and feedback. Lastly, I would like to thank Brage Gaasø Samsonsen for helpful discussions, which have provided valuable insights, and feedback for shaping my ideas and enhancing my work.

Table of Contents

List of Acronyms

CNN Convolutional Neural Network 8

HSI Hyperspectral Image 1

HAD Hyperspectral Anomaly Detection 1

HYPSO Hyper-Spectral Small Satellite for Ocean Observations 1

AE Autoencoder 3

GAN Generative Adversarial Network 9

AUTO-AD Autonomous Hyperspectral Anomaly Detection Network 2

LW-AUTO-AD Lightweight AUTO-AD 2

PCA Principal Component Analysis 2

KPCA Kernel PCA 2

RFF-KPCA Random Fourier Features KPCA 2

MKL Multi Kernel Learning 2

MKPCA Multi KPCA 30

KIFD Kernel Isolation Forest Detection 2

RX Reed-Xiaoli 2

CRD Collaborative Representation Detection 8

PTA Prior-based Tensor Approximation 8

SSIIFD Spectral-Spatial Anomaly Detection Algorithm using an Improved iForest Algorithm 2

Chapter 1

Introduction

The introduction chapter will present an overview of this master thesis, providing an insight into the research topic and its significance, in addition to the thesis objective. The thesis objective will describe the main contributions and any limitations or constraints that are applied. Further, the structure of this thesis is presented to give an overview of the different chapters and their contents.

1.1 Motivation

Oceanography is the study of the different features of the ocean, including the past, present and future conditions. Today the ocean is facing multiple threats, including climate change and pollution. Therefore oceanographic phenomena are of great interest to understanding the effects of climate change and environmental effects of human activities [1].

In recent years, remote sensing utilizing Hyperspectral Imaging has become an important research field for environmental monitoring to understand human activities' effects on climate change. Hyperspectral Imaging (HSI) is a technique for generating a spatial map of spectral variation as a three-dimensional hypercube [2]. This technique can take advantage of hundreds of adjacent spectral channels, which results in detailed spectral information, making it possible to distinguish between materials with fine spectral features. By employing small satellites with Hyperspectral Image (HSI), observations of the characteristics of the ocean can be captured and analyzed before communicating the findings to ground stations which can investigate them further. The HSI system can capture different ocean irregularities, such as algae bloom, phytoplankton, and river plumes [2]. The Hyper-Spectral Small Satellite for Ocean Observations (HYPSO), developed by the Norwegian University of Technology, is designed to detect irregularities in the ocean known as anomalies or targets [3]. To optimize the power budget of the satellite, HYPSO utilizes an Field Programmable Gate Arrays (FPGA)-based image processing algorithm. This algorithm enables onboard anomaly detection, eliminating unnecessary data transmission. Designing this target detection block with low computational cost is crucial to ensure efficient power usage on the FPGA.

The target detection block can be designed using an Hyperspectral Anomaly Detection (HAD) algorithm [4, 5, 6]. HAD aims to identify pixels or sub-pixels in a dataset with significantly different spectral characteristics than its neighboring pixels. These pixels are referred to as outliers or anomalies.

1.2 Project Objective and Description

The first HAD algorithms were proposed in the early 1990s through traditional-based methods [7, 8, 9]. There are several categories within the traditional-based methods, such as distribution-based, representation-based, and tensor decomposition-based methods. However, a prevalent problem with traditional methods is generating good feature extraction for various datasets. This problem can be solved by deep learning-based methods, which have demonstrated effective feature extraction [10, 11]. Several deep learning-based models have been proposed using techniques such as Convolutional Neural Networks (CNNs), Autoencoders (AEs), and Generative Adversarial Networks (GANs) [6, 12, 10].

One approach to developing these deep learning-based models is to use them to reconstruct the background of the HSI, while the anomalies appear as reconstruction errors [5, 6, 13, 14, 15]. This approach falls under unsupervised techniques, as it does not rely on labeled datasets. Several AE-based HAD models have been proposed using this concept, such as the Autonomous Hyperspectral Anomaly Detection Network (AUTO-AD) [5]. This model uses the concept of reconstruction to separate the anomalies from the background. AUTO-AD has demonstrated promising results based on its high detection performance with regard to other state-of-the-art models. Further, AUTO-AD does not require manual parameter setting, which makes it possible to use it across various datasets without obtaining new parameter configurations. However, the model is built using a significantly large number of convolutional layers, which increases the computational cost. Additionally, the model does not apply any pre-processing methods, potentially improving the detection performance.

In this thesis, three main contributions are presented with the focus of enhancing the AUTO-AD model with regard to detection performance and computational cost. The AUTO-AD model was selected for improvement due to its sufficient detection performance and the absence of manual parameter tuning. Furthermore, certain aspects of the model, such as the lack of a pre-processing step and the computationally intensive autoencoder architecture, were recognized as potential areas for modification. These factors made the AUTO-AD model suitable for enhancing its capabilities of detecting anomalies and reducing computational complexity.

The first contribution is the AUTO-AD model in conjunction with Kernel PCA (KPCA)-based pre-processing, denoted as the AUTO-AD$^+$ model. KPCA-based pre-processing can represent more of the complex underlying structure of the dataset, which can enhance the separability between anomalous and background pixels. This pre-processing step can contribute to increased detection performance. It is important to note that this contribution was presented and described in a project thesis conducted by the same author as this thesis [16].

The second contribution is the Lightweight AUTO-AD (LW-AUTO-AD) model, which utilizes a new autoencoder architecture that decreases the computational cost by reducing the number of convolutional layers. This architecture also introduces spectral dimensionality reduction, contributing to a lowered computational cost.

The third contribution is to employ a pre-processing method in conjunction with the LW-AUTO-AD model. This model will be denoted as the LW-AUTO-AD$^+$ model. Pre-processing can enhance the accuracy of the detection map by eliminating noise and irrelevant information, the underlying structure of the dataset, revealing the anomalies that may not be apparent in the raw data. Additionally, the effective feature representation can reduce the number of epochs required for the AE to converge, which reduces the computational cost.

The thesis proposed several different pre-processing methods including Principal Component Analysis (PCA), Kernel PCA (KPCA), Random Fourier Features KPCA (RFF-KPCA) and Multi Kernel Learning (MKL) [17, 18, 19]. MKL applies multiple kernel functions which can capture various aspects of the dataset's structure. This can enhance the separability between the background and anomalous pixels, surpassing the performance of using one kernel function, as in KPCA. However, when utilized on large datasets, KPCA requires a high computational cost. In order to minimize the computational cost while preserving the essence of KPCA transformation, an approach called RFF-KPCA pre-processing can be employed as an approximation to KPCA-based pre-processing.

Different parameter configurations of the LW-AUTO-AD and pre-processing methods are tested to optimize the detection performance. However, for simplifications, extensive testing regarding these configurations will not be performed. For comparison, AUTO-AD and three other traditional machine learning-based state-of-the-art models will be implemented. This includes the Reed-Xiaoli (RX) [7], the Kernel Isolation Forest Detection (KIFD) [20] algorithm and Spectral-Spatial Anomaly Detection Algorithm using an Improved iForest Algorithm (SSIIFD) [4].

For simplification, all HAD models will be tested on the Airport-Beach-Urban (ABU) datasets, which is a commonly used dataset within HAD [21]. It is important to note that parts of the data and pre-processing analysis in this thesis were performed in cooperation with Brage Gaasøe Samsonsen, a fellow student at the Norwegian University of Science and Technology working with the same dataset.

1.3 Structure of the Thesis

This thesis is structured into six main chapters. Chapter 1 provides an introduction to HAD, the motivation, and the objective and project description of this thesis.

In Chapter 2, hyperspectral imaging and a review of state-of-the-art HAD models will be introduced. This chapter will also introduce the different pre-processing algorithms and some traditional and deep learning-based methods. The AUTO-AD model is described in depth, including the Autoencoder (AE) architecture and other methods used to optimize the HAD.

Chapter 3 presents the methodology, including all three configurations of the AUTO-AD model described above. The main goal of this chapter will be to present the lightweight autoencoder architecture that will replace the autoencoder in the AUTO-AD model. This chapter will also present the LW-AUTO-AD$^+$ model.

In Chapter 4, the test setup, together with the dataset and dataset analysis, will be presented. An analysis of the pre-processing methods, parameter configuration, and implementation will be briefly presented. Further, a description of the implementation and parameter configuration of the state-of-the-art models, including RX, KIFD, SSIIFD, and AUTO-AD, will be presented. Next, the parameter configuration of the LW-AUTO-AD$^+$ model will be presented through experimental testing. The results from the experimental testing will be used as the default values when evaluating the model. Finally, the results of the contribution of this master will be presented and compared to four other state-of-the-art models. An analysis of the results will also be presented.

The final chapter, chapter 5, provides the conclusion, including a summary of the contributions made by this thesis and a section about future work.

1.4 Publications from this Thesis

C1 K. Müller, **V. C. Gogineni**, M. Orlandic, and S. Werner, "Autoencoder-based hyperspectral anomaly detection using kernel principal component pre-processing," in *Proc. European Conf. Signal Process.*, 2023.

J1 **V. C. Gogineni**, K. Müller, M. Orlandic, and S. Werner, "Light weight autoencoders for timely hyperspectral anomaly detection," *IEEE Geoscience and Remote Sensing Letters*, 2023 (To be Submitted).

Chapter 2

Background and Literature

This chapter provides the necessary background for understanding the concepts behind hyperspectral imaging and hyperspectral anomaly detection.

2.1 Hyperspectral Imaging

Hyperspectral imaging is a technique where a spatial map of spectral variation is generated as a three-dimensional *hypercube* [2]. The hypercube is denoted by a tensor $\mathbf{X} \in \mathbb{R}^{H \times W \times B}$ where H and W, denote the spatial dimensions and B the spectral dimension, also known as the spectral resolution. Each pixel within the hyperspectral image is defined as a vector $\mathbf{x}_{i,j} \in \mathbb{R}^{B \times 1}$ in the spatial position (i, j), consisting of B spectral bands. The number of pixels N within an HSI is given by $N = H \cdot W$. A visualization of a Hyperspectral Image (HSI) is given in Figure 2.1.

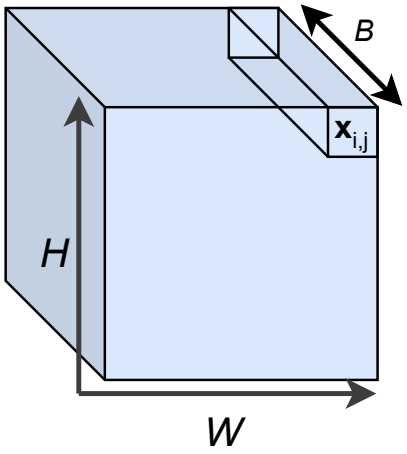


Figure 2.1: Illustration of a hyperspectral image.

Remote sensing involves obtaining information from an object or scene by measuring the reflected and emitted radiation [22]. Hyperspectral imaging has become a fast-growing area in remote sensing, as it can take advantage of hundreds of adjacent spectral channels, which results in detailed spectral information. This detailed information makes distinguishing materials with fine spectral features possible, which may not be distinguishable in RGB or multispectral sensors [23].

A challenge that comes with HSI in remote sensing is the low spatial resolution, which is due to the significant spatial coverage. The spatial resolution refers to the minimum detectable feature in an image [22]. The low spatial resolution can result in pixels that contain more than one material or object. This type of pixel is considered a mixed pixel, further complicating target detection and analysis of the hyperspectral image [23].

2.2 Satellite Technology

Several satellite technologies using hyperspectral imaging systems have been developed for monitoring natural resources, atmospheric characteristics, and environmental characteristics on a global scale. In recent years, low-cost satellites equipped with HSI payloads have been developed to capture daily data from the same geographic location [23]. This section will present some of these satellite technologies and some optical sensors commonly used for HSI.

2.2.1 HYPSO

Hyper-Spectral Small Satellite for Ocean Observations (HYPSO) is a satellite technology developed by the Norwegian University of Technology, which aims to detect irregularities and characterize ocean colour features like algae bloom, phytoplankton and river plumes [3]. Hyperspectral Anomaly Detection is a method for detecting these irregularities in the HSI. To avoid unnecessary data transmission, HYPSO proposes anomaly detection as part of the onboarding process of the FPGA, as described in Figure 9 in [24] as the Target Detection module. This contributes so that HYPSO can perform analysis and anomaly detection onboard the satellite and alert the ground station if anomalies, like algae bloom, are detected. Given the limited computational power available on the FPGA within the small satellite, it is crucial to design the HAD model that minimizes its computational requirements. By reducing the computational cost, the model can operate efficiently on the FPGA without consuming excessive resources.

2.2.2 EnMAP

Environmental Monitoring and Analysis Program (EnMAP) is a German satellite mission that aims to capture HSI data of the Earth's atmosphere, and surface [25]. This satellite mission's main objective is to provide high-spectral resolution observations of biophysical, biochemical and geochemical variables, which can be used to observe a wide range of ecosystem parameters.

2.2.3 PRISMA

PRecursore IperSpettrale della Missione Applicativa (PRISMA) is a medium-resolution hyperspectral imaging satellite developed by ASI (Agenzia Spaziale Italiana) [26]. The mission objective of this satellite project is to develop a small satellite for monitoring natural resources and atmospheric characteristics, with the overall objective of providing a global observation capability.

2.2.4 AVIRIS

Airborne Visible Infrared Imaging Spectrometer (AVIRIS) is a well-established instrument used in remote sensing. It captures calibrated images of the spectral radiance in 224 consecutive spectral channels, covering wavelengths from 400 to 2500 nanometers. AVIRIS has been deployed on four aircraft platforms and utilized in various regions, including North America, Europe, South America, and Argentina. The primary goal of the AVIRIS project is to identify, measure, and monitor components of the Earth's surface and atmosphere [27].

2.3 Hyperspectral Anomaly Detection

Anomaly detection refers to a branch within machine learning that aims to identify patterns in the dataset that differ from the rest of the data [28]. These are often referred to as outliers or anomalies. Anomaly detection has a wide variety of applications like fraud detection for credit cards, insurance and health care, and within target detection in HSI, referred to as Hyperspectral Anomaly Detection [28].

Hyperspectral Anomaly Detection has become an important research field in remote sensing due to its usage within mine detection, environmental monitoring, and search-and-rescue [2]. Satellite projects, such as HYPSO, PRISMA and EnMAP, can use hyperspectral imaging for environmental monitoring of natural resources, atmospheric characteristics, and environmental characteristics on a global scale [3, 26, 25]. Within hyperspectral imaging, anomaly detection is used to

find pixels or sub-pixels with spectral characteristics that differ significantly from the background pixels [23]. Hence, HAD can be used as target detection to identify smaller objects [4, 7].

Figure 2.2 is an example of a plot of the spectral signatures for the anomaly and background pixels in a Hyperspectral Image. In this case, the spectral signature is based on the intensity or the amount of radiation received by the hyperspectral imaging system, which again indicates the pixel's brightness. The example is from the ABU dataset [29] for the 1st beach scene. The mean and standard deviation are displayed in the figure for both the anomaly and background pixels. The separability properties of the HSI can be indicated by looking at the overlap or lack of overlap between the mean and standard deviation values. For example, the figure shows no overlap between the mean values of the anomalies and background. However, the standard deviation range for the anomalies is extensive, which causes there to be an overlap between the background and anomaly standard deviation, which can result in some misclassification.

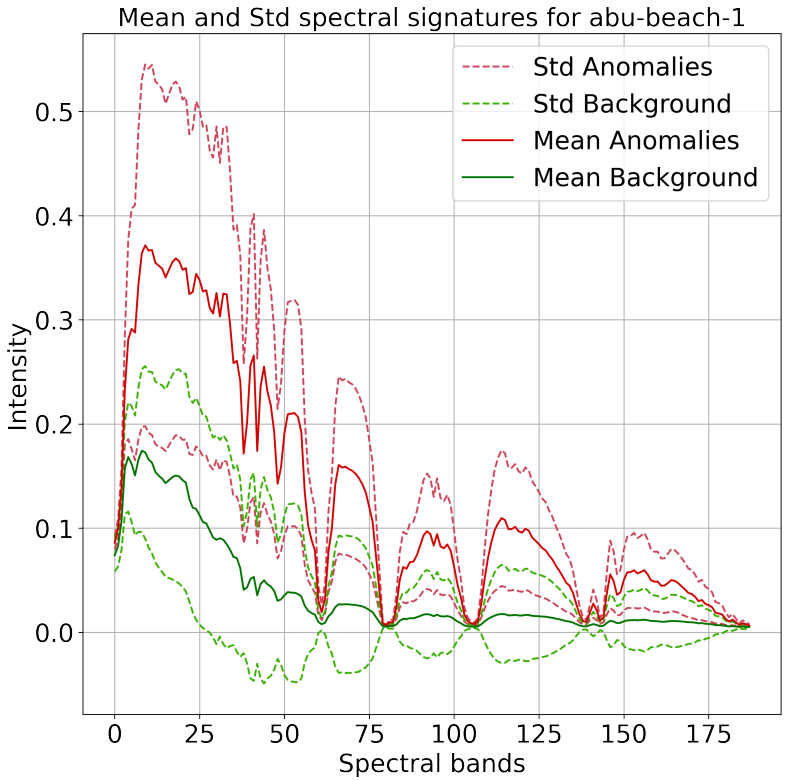


Figure 2.2: Plot illustrating the different spectral characteristics of anomaly pixels and background pixels.

HAD algorithms commonly follow the basic structure presented in Figure 2.3, which consists of an input HSI, an anomaly detection algorithm which identifies the anomalies in the image, and the resulting detection map containing the detected anomalies [4, 6]. Some examples of these algorithms will be presented in the next section.

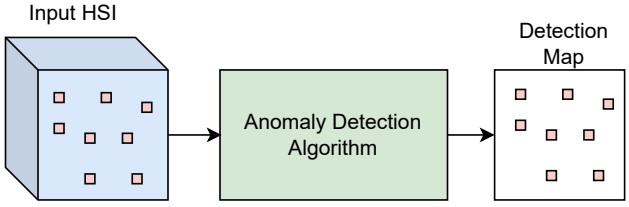


Figure 2.3: Framework for a basic hyperspectral anomaly detection model.

Another commonly used structure for HAD models is presented in Figure 2.4, where an additional pre-processing block has been introduced between the input HSI and anomaly detection

algorithm. The pre-processing block is an essential step in machine learning algorithms. It can handle noise and reduce the computational complexity in the HSI by performing dimensionality reduction, feature scaling or feature extraction [30]. Dimensionality reduction is a technique that aims to reduce the number of features in a dataset while preserving crucial information. Within HSI, dimensionality reduction can reduce the number of spectral bands while preserving the essential information [4, 6]. On the other hand, feature scaling aims to normalize the scale of the features in the dataset to ensure that the features have similar ranges or distributions. In contrast, feature extraction aims to identify the features that capture the dataset's most relevant and discriminant information. The extracted features can provide a more compact and informative representation of the dataset, which in some cases can be used to reduce the dimensionality.

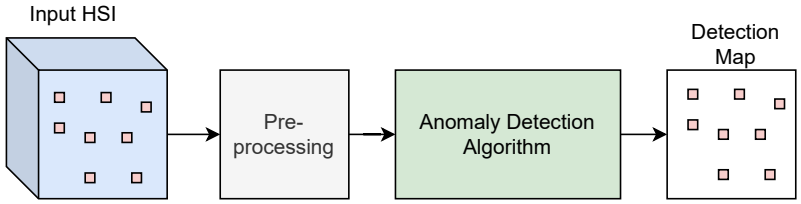


Figure 2.4: Framework for a basic hyperspectral anomaly detection model.

It should be noted that in some cases, the anomaly detection algorithm may only consist of a pre-processing algorithm. In this case, the pre-processing block serves as the anomaly detection algorithm block.

2.3.1 Methods of Hyperspectral Anomaly Detection

As presented above, HAD methods can be categorized into traditional - and deep learning-based machine learning models. A timeline of some of these HAD methods, which have shown some of the best performance in terms of detection performance and computational cost, is presented in Figure 2.5. The blue arrows represent traditional-based machine learning methods, and the orange represents deep learning-based methods. A brief introduction to each model is provided below. The HAD methods discussed are all unsupervised techniques, which have the advantage of not requiring labelled datasets. Unsupervised techniques are particularly beneficial in HAD, as labelled datasets are limited.

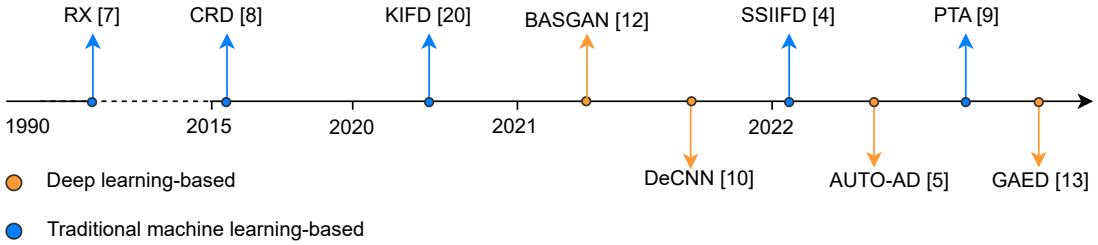


Figure 2.5: Timeline of some of the state-of-the-art HAD models. This figure is inspired by [21].

Various traditional machine-learning-based HAD methods have been proposed, such as statistical model-based, distance-based and prior-based methods [21]. The first proposed statistical-based model is the Reed-Xiaoli (RX) [7], which assumes that the background follows a multidimensional Gaussian distribution. Anomalies are found by using the Mahalanobis distance [31] to measure the distance to the Gaussian distribution as follows:

$$D_{RX}(\mathbf{x}) = (\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu) \tag{2.1}$$

where $\mathbf{x}$ is a pixel in the HSI, and μ and $\mathbf{\Sigma}$ are the estimated background mean and covariance matrix. Due to the complex structure of an HSI, the Gaussian distribution has shown to lack the ability to model the background distribution sufficiently [21] .

The representation-based methods include sparse representation, low-rank representation and collaborative representation. Among these is the Collaborative Representation Detection (CRD) model [8]. This algorithm assumes that background pixels can be modelled as a linear combination of the surrounding pixels, while the anomalies cannot. The CRD has outperformed several state-of-the-art models with regard to detection performance [6].

The Prior-based Tensor Approximation (PTA), [9], is a decomposition-based method that utilizes tensor representations. This method takes advantage of the 3D structure of the HSI, which has also shown high detection performance compared to other state-of-the-art models [21]. However, the CRD and PTA models require manual parameter setting, making it difficult to generalize them to new datasets. It is worth noting that the RX, CRD and PTA algorithms are commonly used as a comparison for HAD algorithms [5, 6, 21].

Another traditional-based method recently introduced to HAD is the iForest algorithm [32], which isolates the anomalies from the background using iTrees. The Kernel Isolation Forest Detection (KIFD) algorithm was the first HAD algorithm to incorporate the iForest together with the KPCA - based pre-processing [20]. This model utilizes KPCA to transform the input data into a higher-dimensional feature space where the iForest algorithm can extract more complex features. These complex features allow better distinguishing between anomalies and background pixels in the transformed space. KIFD has shown to outperform several state-of-the-art models. However, KPCA-based pre-processing requires the computation of both the kernel matrix and singular value decomposition (SVD) [33]. Both computations increase the computational cost by $O(N^2)$ and $O(N^3)$, respectively. Additionally, the results indicate that the model struggles to utilize local spatial information effectively.

The Spectral-Spatial Anomaly Detection Algorithm using an Improved iForest Algorithm (SSIIFD) was proposed to improve the struggles of the KIFD model [4]. SSIIFD exploits spectral and spatial information by dividing the model into two parts, each generating a detection map. These detection maps are then fused together to produce the final detection map. The SSIIFD has outperformed both the KIFD and other state-of-the-art models in terms of detection performance.

One common challenge traditional methods pose is achieving effective feature extraction for various data types [21]. Deep learning-based methods can overcome this automatic feature extraction [10, 11]. The deep learning-based models include Convolutional Neural Networks (CNNs), Autoencoders (AEs) and Generative Adversarial Networks (GANs).

Convolutional Neural Networks (CNNs) can extract data features from previous convolutional layers and learn higher-level feature representation from lower-level feature information using convolutional filters, which can be of great significance in HAD. The Convolutional Neural Network (CNN) has the additional benefit of having a strong self-learning ability and automated feature extraction [34]. The Deep Plug-and-Play Denoising CNN (DeCNN) [10] is a HAD that has shown promising detection performance. This model uses a low-rank representation (LRR) based anomaly detection to represent the background, which can be used to identify anomalies which are not well-represented by the LRR. Further, the model applies the CNN to detect the anomalies.

Similar to traditional based methods, deep learning-based methods can be categorized into different types, among which residual error-based approaches are one [5, 6, 13, 14, 15]. The residual error-based approach assumes that anomalies are minor compared to the background and occur less frequently. These characteristics make the anomalies more challenging to reconstruct. Hence they will appear as reconstruction errors. The reconstruction error is found by comparing the reconstructed HSI created by the HAD with the original HSI. As a result, the anomalies are detected based on locations in the image with high reconstruction error. Figure 2.6 illustrates this, where the HSI is processed using a HAD, resulting in a reconstructed background. A detection map is also generated based on the residual error, which are the areas of the image with high reconstruction error.

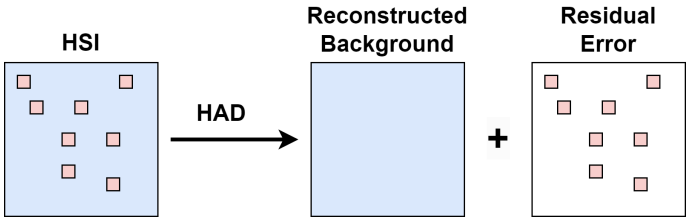


Figure 2.6: Illustration of the residual error-based approach for HAD.

Autoencoders are often employed as residual error-based methods for anomaly detection. One such model is the Autonomous Hyperspectral Anomaly Detection Network (AUTO-AD), which utilizes a fully convolutional AE with skip connections to reconstruct the background and separate the anomalies [5]. This model has demonstrated superior detection performance compared to several state-of-the-art models and does not require any parameter tuning. However, this model consists of many convolutional layers, which increases the computational cost. Additionally, the model does not apply pre-processing, which may enhance the detection performance.

Another such model is the guided autoencoder (GAED), which outperforms several state-of-the-art models with regard to detection performance [13]. The proposed model includes a multilayer AE network with skip connections and a guided module designed to suppress anomalies' representations and enhance the background features' representation.

Generative Adversarial Network (GAN)s are generative models that capture the distribution of the dataset to learn new data points [35]. The GAN consists of two models: a generative model (G) and a discriminator model (D). The generator learns the features of the input data to produce new data while tricking the discriminator into detecting the new data as "real" data. In contrast, the discriminator is trained to correctly classify the generated and real data as real or fake.

Recently, the background anomaly separable feature method based on GAN (BASGAN) was proposed [12]. This model has demonstrated promising results for HAD. BASGAN constructs a pseudo-background to separate anomalies from the background, thus overcoming the need for manual and accurate labelling. The model also includes background suppression, which enhances the contrast between the anomalies and the background.

2.4 Pre-processing Algorithms

Data pre-processing is an essential step in machine learning for handling noisy data, feature scaling and extraction, dimensionality reduction and reducing computational complexity [30]. Within Hyperspectral Anomaly Detection, data pre-processing has proven to give improved accuracy by removing noise and irrelevant information from the data and improved computational efficiency by introducing dimensionality reduction [36, 20, 37]. In this section,Principal Component Analysis (PCA), Kernel PCA (KPCA), Multi Kernel Learning (MKL) and Random Fourier Features KPCA (RFF-KPCA) will be presented.

2.4.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a mathematical algorithm used in data analysis for multivariate data [38]. PCA can be used for dimensionality reduction and feature selection within HAD [20, 4]. This pre-processing method performs an orthogonal linear transformation of the dataset, resulting in a set of Principal Component (PC)s. PCA aims to maintain most of the variation in the dataset to preserve important information in the data by maximizing the variance along the axis of the PCs. The figure below provides a visual representation of this concept, where a two-dimensional dataset is processed using two PCs, which maximize the variance along the axis, denoted by the red lines.

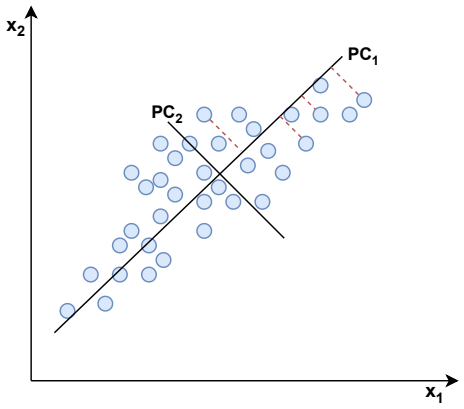


Figure 2.7: Illustration of PCA-based pre-processing on a two-dimensional dataset.

The covariance matrix $\mathbf{C}$ of the dataset $\mathbf{X}$ is exploited to find the PCs since it provides an estimate of the variability of the dataset. Hence, by performing the eigenvalue decomposition on $\mathbf{C}$, a set of eigenvectors and eigenvalues representing the PCs and variance along each PC can be obtained. The first step in PCA is to find the covariance matrix $\mathbf{C}$ given as

$$\mathbf{C} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{x}_k \mathbf{x}_k^{\mathrm{T}} \tag{2.2}$$

where $\mathbf{x}_k$ is a pixel from the dataset $\mathbf{X}$, and N is the number of pixels present in the dataset. For PCA, the dataset is assumed to be centralized, meaning that the dataset has zero mean as follows:

$$\sum_{i=1}^{N} \mathbf{x}_i = 0 \tag{2.3}$$

The PCs are, as mentioned, found using the eigenvalue decomposition expressed as

$$\mathbf{CP} = \mathbf{\Lambda P} \tag{2.4}$$

where $\mathbf{P}$ contains the eigenvectors of $\mathbf{C}$, also referred to as the PCs, and $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues of $\mathbf{C}$ in its diagonal. The principal components are arranged based on the highest eigenvalues, which indicates the amount of variance.

For dimensionality reduction, only the first ξ principal components are used to create a new feature matrix $\mathbf{F} \in \mathbb{R}^{H \times W \times \xi}$, and used to transform the dataset $\mathbf{X} \in \mathbb{R}^{H \times W \times B}$ using

$$\widetilde{\mathbf{X}} = \mathbf{F}^T \times \mathbf{X} \tag{2.5}$$

where $\widetilde{\mathbf{X}} \in \mathbb{R}^{H \times W \times \xi}$ is a tensor containing the new transformed dataset.

Feature scaling is an essential step in PCA, for ensuring that the variance is calculated using the same scale [30]. A commonly used method is the min-max normalization, which scales the values of $\mathbf{X}$ between $[0, 1]$ using

$$\bar{\mathbf{X}} = \frac{\mathbf{X} - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})} \tag{2.6}$$

2.4.2 Kernel Principal Component Analysis (KPCA)

Kernel PCA (KPCA) is a technique used for implementing a nonlinear version of PCA that can be utilized for non-linearly separable data. By applying kernels, the data $\mathbf{X}$ can be transformed to a higher dimensional feature space, where the data is linearly separable. This transformation can be found using the transformation function, $\phi(\mathbf{x})$. The concept of KPCA is illustrated in Figure 2.8. The Figure visualizes a non-linearly separable dataset, consisting of two classes, red and blue, in a 2-dimensional feature space (x1, x2) before and after transformation using the function $\phi(\mathbf{x})$ to a 3-dimensional feature space (x1, x2, x3), where the dataset becomes linearly separable.

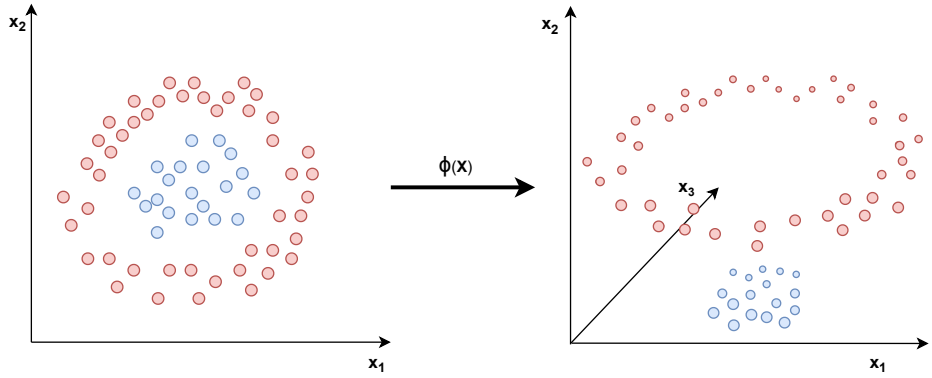


Figure 2.8: Illustration of the concept of KPCA-based pre-processing.

To avoid explicitly transforming the data, KPCA utilizes the kernel trick to compute the kernel matrix [17]. The basic idea of the kernel trick is to apply a kernel function that can compute the inner product between two vectors $\mathbf{x}$ and $\mathbf{x}'$ in the higher dimensional feature space. The kernel function $\kappa(\cdot)$ needs to be a continuous, symmetric, positive definite function that satisfies Mercer's condition. It is defined as

$$\kappa(\mathbf{x}, \mathbf{x}') = \phi^{\mathrm{T}}(\mathbf{x})\phi(\mathbf{x}') \tag{2.7}$$

Some commonly used kernels are the Radial Basis Function (RBF), sigmoid and Laplacian, given in (2.8), (2.9),(2.10),respectively [17].

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma||\mathbf{x} - \mathbf{x}'||^2\right), \tag{2.8}$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \tanh(\gamma \mathbf{x}^{\mathrm{T}}\mathbf{x}' + r), \tag{2.9}$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||}{\sigma}\right), \tag{2.10}$$

where σ, γ and ,r are hyperparameters representing the variance, inverse of variance and offset, respectively.

The first step in KPCA is to express the covariance matrix $\hat{\mathbf{C}}$ as an inner product between two transformed vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$ as follows

$$\hat{\mathbf{C}} = \frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x})\phi^{\mathrm{T}}(\mathbf{x}') \tag{2.11}$$

Further, the covariance matrix $\hat{\mathbf{C}}$ is denoted as the kernel matrix $\mathbf{K} = \{\kappa_i\}_{i=1}^{N}$. This gives a new eigenvalue decomposition problem expressed as

$$\mathbf{KP} = \mathbf{\Lambda P} \tag{2.12}$$

where $\mathbf{P}$ and λ represent the same as in PCA.

It is important to note that the equations above assume a centred data that follows (2.3). To ensure this, the centered covariance matrix $\mathbf{K}_c$ must be computed as

$$\mathbf{K}_c = \mathbf{K}_c - I_N\mathbf{K}_c - \mathbf{K}_c I_N + I_N\mathbf{K}_c I_N \tag{2.13}$$

where $\mathbf{I}_N$ denotes the $N \times N$ matrix where each element takes the value $1/N$. When employing KPCA, it is possible to capture complex underlying structures of the data, which is useful for identifying anomalies which may not be apparent in the raw data [39]. However, the computation of the kernel matrix comes with a high computational cost when N is large [40].

2.4.3 KPCA using Random Fourier Features (RFF-KPCA)

Random Fourier Features KPCA (RFF-KPCA) can be used to accelerate the computation of the kernel function in (2.7) when the amount of data samples, N, is large [19]. RFF-KPCA

approximates the kernel function as an inner product in the D dimensional RFF space [41]. This approximation avoids the high computation of the kernel function because it becomes a finite-dimensional linear filtering problem. RFF-KPCA approximates the feature transformation $\phi(\cdot) \approx z(\cdot)$, which is used to compute the kernel function $\kappa(\cdot)$. The new kernel function is expressed as $\kappa(\mathbf{x}, \mathbf{x}') = \phi^{\mathrm{T}}(\mathbf{x})\phi(\mathbf{x}') \approx z^{\mathrm{T}}(\mathbf{x})z(\mathbf{x}')$, where $z(\mathbf{x})$ is given as

$$z(\mathbf{x}) = \sqrt{\frac{1}{D}} \left[\cos(\mathbf{w}_1\mathbf{x} + b_1) ... \cos(\mathbf{w}_D\mathbf{x} + b_D) \sin(\mathbf{w}_1\mathbf{x} + b_1) ... \sin(\mathbf{w}_D\mathbf{x} + b_D)\right] \tag{2.14}$$

The phase terms $\{b_i\}_{i=1}^{D}$ are drawn from the uniform distribution $\mathcal{U} \in [0, 2\pi]$,and the RFF vectors $\{\mathbf{w}_i\}_{i=1}^{D}$ are drawn from a distribution $p(\mathbf{w})$ such that

$$\kappa(\mathbf{x} - \mathbf{x}') = \int p(\mathbf{w}) \exp(j\mathbf{w}^{\mathrm{T}}(\mathbf{x} - \mathbf{x}')\, d\mathbf{w} \tag{2.15}$$

where $j^2 = -1$. The RFF vectors can for example be the Gaussian or Laplacian distribution [19]. Some commonly used distribution function $p(\mathbf{w})$ are given in Table 2.1.

Table 2.1: Possible distribution functions $p(\mathbf{w})$ used within RFF-KPCA.

Kernel	$p(\mathbf{w})$
Gaussian	$(2\pi)^{\frac{-D}{2}} \exp\left(\frac{\|\|\mathbf{w}\|\|_2^2}{2}\right)$
Laplacian	$\prod_N \left(\pi(1 + \mathbf{w}_N^2)\right)^{-1}$
Cauchy	$\exp\left(\|\|\Delta\|\|_1\right)$

The computational cost of calculation $\phi(\mathbf{x})$, when $\mathbf{X} \in \mathbb{R}^{N \times B}$, is $O(BN)$ where N is the amount of samples and B is the dimensionality of $\mathbf{X}$ before the transformation. For $z(\mathbf{x})$, the computational cost is $O(D + B)$, which has a significant impact when the number of samples N in the dataset is large.

2.4.4 Multi Kernel Learning (MKL)

Multi Kernel Learning (MKL) combines multiple kernel functions to capture various aspects of the intricate data structure [18]. This technique can contribute to enhancing the background and anomaly separability.

One MKL method is to replace the kernel matrix $\mathbf{K} = \{\kappa_i\}_{i=1}^{N}$ in KPCA, with the kernel matrix $\mathbf{K}_\eta = \{\kappa_i^\eta\}_{i=1}^{N}$, where κ_η consists of a linear combination of M different kernel functions κ, given as

$$\kappa_\eta(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{M} \eta_i \kappa_i(\mathbf{x}, \mathbf{x}') \tag{2.16}$$

where η_i denotes the weights associated to the kernel functions $\kappa_i(\mathbf{x}, \mathbf{x}')$.

Within hyperspectral anomaly detection, this pre-processing method replaces the pre-processing block or the anomaly detection algorithm, visualized in Figures 2.3 and 2.4, respectively. Furthermore, the proposed method offers the flexibility to employ the RFF-KPCA-based pre-processing technique using various distribution functions $p(\mathbf{w})$, as outlined in (2.15). Table 2.1 presents a range of potential distribution functions that can be utilized for this purpose.

An alternative way to incorporate MKL within HAD is by simultaneously performing multiple KPCA-based or RFF-KPCA-based pre-processing algorithms This can be achieved using multiple numbers of the HAD framework illustrated in Figure 2.3 or Figure 2.4 in parallel. The resulting output is multiple detection maps. The final detection map is found by combining these multiple maps using a decision fusion technique. If M kernel functions are utilized, the final detection map $\mathbf{D}$ is found using

$$\mathbf{D} = \frac{v_1\mathbf{D}_1 + v_2\mathbf{D}_2 ... + v_M\mathbf{D}_M}{v_1 + v_2 + ... + v_M} \tag{2.17}$$

where $\{\mathbf{D}_i\}_{i=1}^{M}$ are the detection map retrieved from the M kernel functions, and $\{v_i\}_{i=1}^{M}$ are the fusion weights found using a grid search.

Figure 2.9 illustrates the modified basic framework of HAD with multiple pre-processing blocks, anomaly detection algorithm blocks, and detection map blocks. Each pre-processing block

represents one KPCA-based or RFF-KPCA-based pre-processing algorithm. The diagram presents an example using two pre-processing blocks, but this approach can easily be scaled up to accommodate more pre-processing blocks.

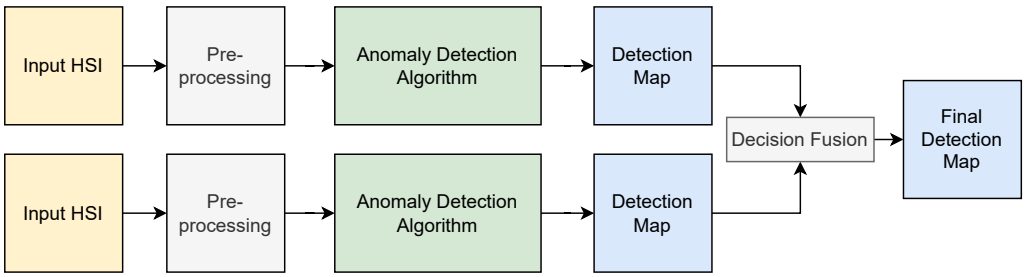


Figure 2.9: Visual representation of MKL.

2.5 Traditional Machine Learning-Based Methods

This section will present two traditional based machine learning models, the KIFD [4] and the SSIIFD [20]. Both models are based on the isolation forest (iForest) algorithm and are commonly used as a comparison to measure the performance of other HAD models [21, 6, 4].

2.5.1 Isolation Forest Algorithm (iForest)

The Isolation Forest (iForest) algorithm is used for anomaly detection [32]. This algorithm uses binary trees to detect anomalies by taking advantage of two properties with anomalies: their feature values differ from the normal instances, and they are the minority within the dataset [32]. As a result, it is easier to distinguish anomalies from the background pixels since they exhibit dissimilarities with the majority of the remaining pixels, commonly denoted as normal instances or background instances. Figure 2.10 shows an example of isolating a non-anomalous point, denoted in blue, and isolating an anomalous point in red. The figure demonstrates that fewer splits are needed to isolate the anomalous point than the non-anomalous one.

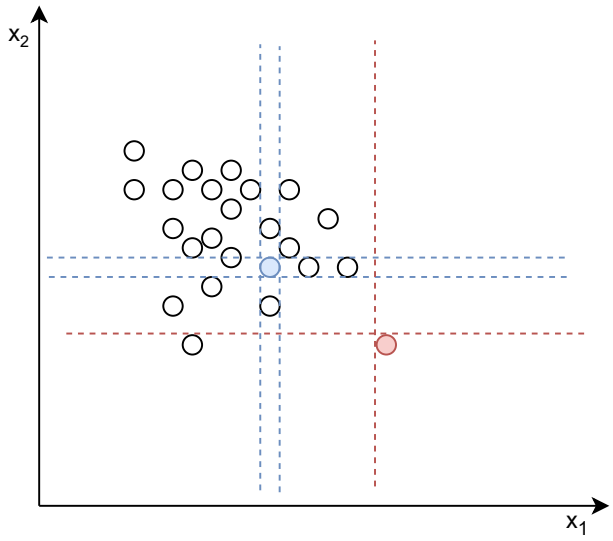


Figure 2.10: The figure illustrates isolation of a non-anomalous point (blue) and an anomalous point (red).

The iForest algorithm is divided into two stages. First, the isolation Trees (iTrees) are constructed using sub-samples of the training set $\mathbf{X}$. The tree structure is visualized in Figure 2.11, where the root node is the first and main node. The subsequent nodes are constructed by dividing

the data at each node based on a randomly selected feature and threshold. The isolation depth, which denotes the path length for a given instance $\mathbf{x}$, is calculated from the root node to the node where the instance is placed. Since anomalous instances require fewer splits than normal instances, they are isolated much closer to the root note. This concept is visualized in the figure, where the anomalous instance is denoted in red, and the normal instances are denoted in blue.

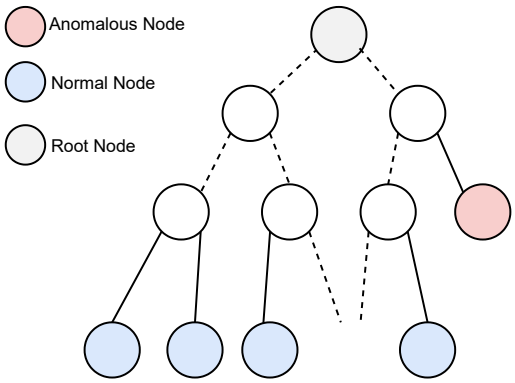


Figure 2.11: Figure illustrating the structure of an iTree.

The iForest is constructed from q iTrees, which are randomly generated by M randomly selected instances of the dataset. Each iTree is divided based on a specific feature and threshold θ. When using hyperspectral dataset $\mathbf{X}^{N\times B}$, the spectral dimensions are denoted as the different features. Hence a randomly selected feature is denoted as $\mathbf{X}_s^m$, where $s \in [1, B]$, and m is the given pixel selected from the M randomly selected instances. The iteratively splitting within an iTree stops if the tree reaches a specific height limit H_{max}, the number of pixels in each node is 1, or if the pixels in each node have the same value.

The second step of the iForest algorithm is finding the anomalous score. This step requires calculating $h(\mathbf{x})$, which represents the height of a test instance $\mathbf{x}$. The height, is found by counting the number of splits which are required to isolate the test instance. First the average path is calculating by passing test instances trough the iTrees, and calculating the length $h(\mathbf{x})$ for the test instance $\mathbf{x}$ over q iTrees:

$$E(h(\mathbf{x})) = \frac{1}{q}\sum_{i=1}^{q} h_i(\mathbf{x}) \tag{2.18}$$

where $h_i(\mathbf{x})$ denotes the length for the test instance for a given iTree. The anomaly score $s(\mathbf{x})$ is obtained using

$$s(\mathbf{x}) = 2^{-\frac{E(h(\mathbf{x}))}{K}} \tag{2.19}$$

where K is a constant, and $s \in (0, 1]$. The anomaly score and average path $E(h(\mathbf{x}))$ exhibit an inverse relationship, implying that a higher average path length results in a lower anomaly score, and conversely, a lower average path length will leads to a higher anomaly score.

2.5.2 KIFD

Kernel Isolation Forest Detection (KIFD) [20], was the first HAD to introduce the iForest algorithm together with KPCA-based pre-processing [39]. By introducing KPCA, the HSI is transformed into a higher dimensional feature space, where the dataset is linearly separable, making it easier to distinguish the anomalies. Figure 2.12 depicts the architecture of the KIFD model.

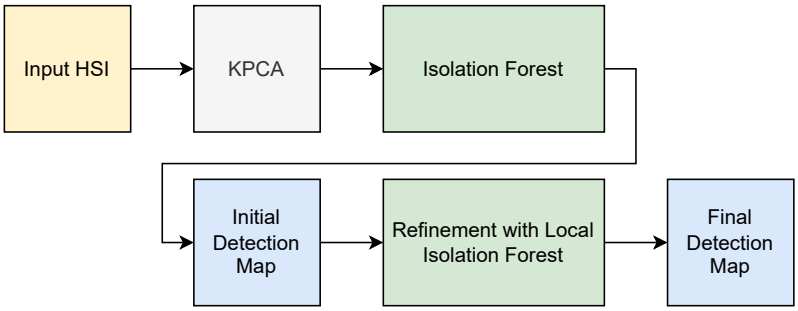


Figure 2.12: Architecture of the KIFD model, based on the architecture in [20].

As shown in Figure 2.12, the HSI $\mathbf{X}$ is first pre-processed using KPCA, which results in a new data matrix $\mathbf{X}_{KPCA} \in \mathbb{R}^{H \times W \times \xi}$, where ξ denotes the number of PCs. After the pre-processing, the new data matrix is passed as input to the iForest algorithm, which outputs a detection map of dimension $H \times W$, based on the anomaly score in (2.19). The iForest algorithm outputs the initial anomaly detection map.

To further suppress the representation of anomalies, a local iForest is applied to refine the detection map further [20]. The local iForest algorithm is based on anomalies appearing in small areas. Hence, if the detected anomalies exceed a specific threshold value α, they can be reevaluated by constructing an iForest using only the pixels within the detected anomaly region. This method is iteratively applied until all anomalies within the detection map are below the threshold value α. The model has demonstrated high detection performance on several standard datasets. However, it has shown difficulty exploiting the local spatial information in addition to an increased computational cost when introducing KPCA-based pre-processing [20].

2.5.3 SSIIFD

In [4], the Spectral-Spatial Anomaly Detection Algorithm using an Improved iForest Algorithm (SSIIFD) was proposed to improve the KIFD model. The SSIIFD model makes full use of both the spectral and spatial information, in addition to the global and local information within the hyperspectral image. The SSIIFD architecture is shown in Figure 2.13.

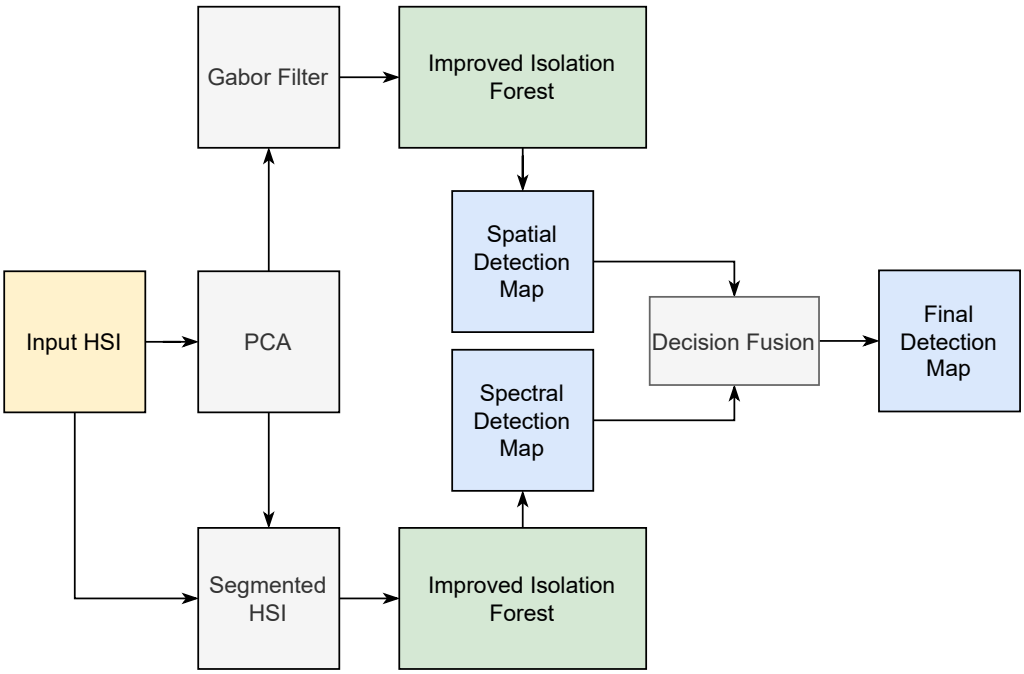


Figure 2.13: Architecture of the SSIIFD, based on the architecture in [4].

The model is divided into two sections, a spectral and spatial anomaly detection part. Each

section computes a detection map, denoted as the spectral and spatial detection map. The spectral detection map is generated by leveraging the spectral components of the model, incorporating all the spectral information available. In contrast, the spatial map is computed using the spatial part of the model, which focuses solely on the first principal components derived from the PCA-based pre-processing method. These detection maps are fused together to obtain the final detection map. Both detection maps are obtained using an improved iForest algorithm (IIF), which aims to improve some of the key challenges of the standard iForest algorithm: detection of anomalies masked by normal instances and selecting more separable bands during iTree construction. In this setting, bands denote the selected spectral dimension.

The input to the model is the HSI $\mathbf{X}$ which is passed through a pre-processed block that performs PCA based pre-processing using only the first principal component, resulting in a new data tensor $\mathbf{X}_{PCA} \in \mathbb{R}^{H \times W \times 1}$. For the spectral anomaly detection section, a segmentation algorithm divides the HSI into η homogeneous regions. The IIFD is then applied to each region separately to obtain the spectral anomaly detection map. In this way, the isolation is based on local areas rather than the entire image. This approach allows anomalies to be compared to nearby pixels, which can be advantageous when anomalies have similar values to normal instances in the surrounding area. The segmentation algorithm used in the model is the Entropy Rate Superpixel (ERS) segmentation [42] algorithm. The ERS algorithm is applied to the two-dimensional pre-processed matrix $\mathbf{X}_{PCA}$, to obtain the η regions. Further, the HSI $\mathbf{X}$ is divided using the regions defined by the ERS algorithm.

For the spatial anomaly detection part, $\mathbf{X}_{PCA}$ is used as input to a Gabor filter bank. Gabor filters are linear bandpass filters that can be used for feature extraction within computer vision and image processing, as they have been shown to capture both low- and high-level features such as edges, lines, texture and shapes of images [43]. The filter is constructed from complex sinusoids that are modulated by the Gaussian function, resulting in a two-dimension filter g with a real and imaginary component defined by

$$g(a, b; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{a'^2 + \gamma^2 b'^2}{2\sigma^2}\right) \times \exp\left(i\left(2\pi\frac{a'}{\lambda} + \psi\right)\right) \tag{2.20}$$

where

$$a' = a\cos(\theta) + b\sin(\theta) \tag{2.21}$$

$$b' = -a\sin(\theta) + b\cos(\theta) \tag{2.22}$$

and a, b are the coordinates, λ denotes the wavelength of the sinusoidal factor, θ the angel in the xy- plane, ψ is the phase offset, γ is the spatial aspect ratio and σ denotes the standard deviation of the Gaussian envelope. In image processing application, it is common practice to use a filter bank of Gabor filters for feature extraction [4, 44]. This makes it possible to extract spatial frequency structures from the image [43]. The filters are created with different orientations, by adjusting θ and ω as follows

$$\omega_u = \frac{\pi}{2}\sqrt{2}^{-(u-1)} \quad u = 1, 2, ..., U \tag{2.23}$$

$$\theta_v = \frac{\pi}{8}(v-1) \quad v = 1, 2, ..., V \tag{2.24}$$

where U denotes the number of scales and V denotes the number of orientations, $D = U \times V$ denotes the amount of Gabor filters in the filter bank. A Gabor filter bank with $U = 5$ number of scales and $V = 8$ number of orientations is visualized in the Figure 2.14.

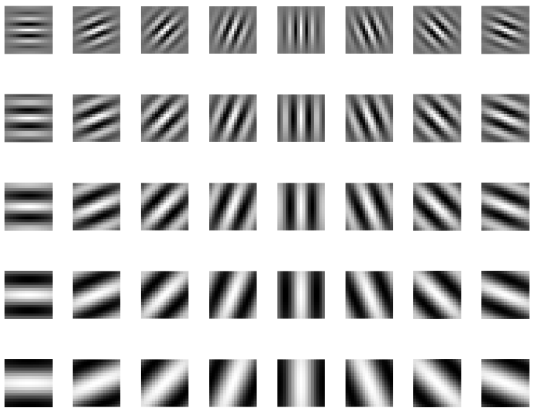


Figure 2.14: Visualization of a Gabor filter bank.

The output of the Gabor filter bank is a data matrix $\mathbf{X}_{Gab} \in \mathbb{R}^{H \times W \times D}$. Further, the IIFD is applied to $\mathbf{X}_{Gab}$ to obtain the spatial detection map. The final detection map D_0 is found using

$$D_0 = \omega D_{spectral} + (1 - \omega) D_{spatial} \tag{2.25}$$

where $D_{spectral}$ and $D_{spatial}$ denote the spatial and spectral detection maps, and ω denotes the fusion weight.

2.6 Deep Learning based methods

Deep learning is a machine learning method based on Artificial Neural Network (ANN)s, which automates the feature extraction process, making it easier to perform tasks such as detection and classification [45]. As previously mentioned, numerous hyperspectral anomaly detection methods have focused on deep learning-based methods because of their capability to extract deep features[21].

2.6.1 Feedforward Neural Network

The feedforward neural network, also known as the Multilayer Perceptrons (MLPs), is denoted as the classical deep learning model [46]. Figure 2.15 shows an example of a feedforward neural network with multiple layers consisting of perceptrons. The first layer is denoted as the input layer, while the subsequent layers are called the hidden layer. The last layer is the output layer, which outputs the prediction made by the Neural Network (NN).

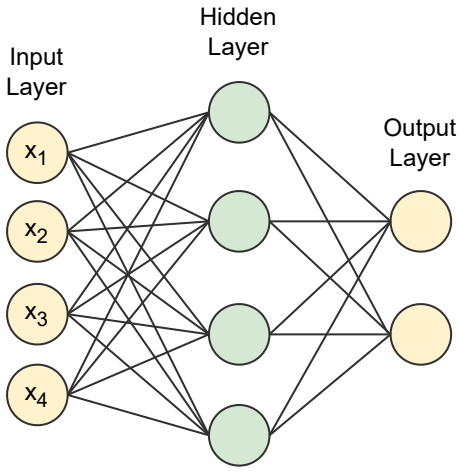


Figure 2.15: Example of a feedforward neural network consisting.

An example of a basic perceptron is illustrated in Figure 2.16.

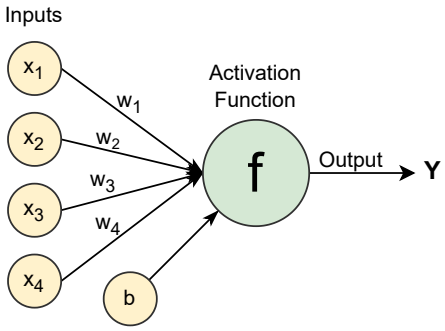


Figure 2.16: Example of a basic perceptron.

Each perceptron takes an input vector of size N $\mathbf{x} = \{x\}_{i=1}^{N}$ and uses a weight $\mathbf{w} = \{w\}_{i=1}^{N}$ and bias b together with an activation function $f(\cdot)$ to compute the output Y as follows:

$$Y = f(\mathbf{w} \cdot \mathbf{x} + b) = \sum_{i=1}^{N} f(w_i \cdot x_i + b) \tag{2.26}$$

The activation function determines if the neuron should be active by applying a non-linear function, making it possible for the network to learn complex patterns [47]. Two common activation functions, the Sigmoid and LeakyReLU are presented below:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{2.27}$$

$$LeakyReLU(a) = \begin{cases} a & \text{if } a > 0, \\ 0.01 \cdot a & \text{otherwise.} \end{cases} \tag{2.28}$$

The Sigmoid activation function ensures that the input a is transformed to a value between $[0, 1]$, while the LeakyReLU is a threshold activation function [48].

The network weights and biases are trained using backpropagation until the difference from the produced output is similar to the desired output, referred to as the loss. Each training iteration is known as an epoch. The adjustments of the weights w_i and biases b are calculated using:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \lambda \frac{dL(\mathbf{x})}{d\mathbf{x}} \tag{2.29}$$

$$b_{k+1} = b_k - \lambda \frac{dL(\mathbf{x})}{d\mathbf{x}} \tag{2.30}$$

where k is the previous epoch, $\mathbf{w}_k$ is the weight from the previous epoch, $\mathbf{w}_{k+1}$ is the updated weight, $\frac{dL(\mathbf{x})}{d\mathbf{x}}$ is the gradient of the loss function $L\mathbf{x}$) and λ is the learning rate.

The loss function can take many forms, but some of the most common are the L1 and L2 loss given in equations (2.32) and (2.31), where $\mathbf{x}_{i,j} \in \mathbb{R}^{B\times 1}$ is a pixel in the spatial position (i,j) of an HSI $\mathbf{X} \in \mathbb{R}^{H\times W\times B}$ and $\hat{\mathbf{x}}_{i,j} \in \mathbb{R}^{B\times 1}$ be the corresponding output of the network [46].

$$L1 = \sum_{i=1}^{H}\sum_{i=j}^{W} ||\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}||_1 \tag{2.31}$$

$$L2 = \sum_{i=1}^{H}\sum_{i=j}^{W} ||\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}||_2^2 \tag{2.32}$$

A standard optimization method for backpropagation is the Adaptive Moment Estimation (ADAM) optimization algorithm [49], which considers both the first and second momentum when setting the learning rate. ADAM is given by

$$\mathbf{w}_k = \mathbf{w}_{k+1} - \lambda \frac{\hat{\mathbf{m}}_k}{\sqrt{\hat{\mathbf{v}}_k} + \epsilon} \tag{2.33}$$

where λ denotes the learning rate, ϵ a small constant to avoid dividing by zero, and $\hat{\mathbf{m}}_k$ is the average momentum, and $\hat{v_k}$ is the average sum of past gradients. These are given by

$$\hat{\mathbf{m}}_k = \frac{\mathbf{m}_t}{1+\beta_1} \tag{2.34}$$

$$\hat{\mathbf{v}}_k = \frac{\mathbf{v}_t}{1+\beta_2} \tag{2.35}$$

where β_1 and β_2 are constants denoting the decay rates of the average of the gradients, and $\mathbf{m}_k$ and $\mathbf{v}_k$ are given as

$$\mathbf{m}_k = \beta_1 \mathbf{m}_{k-1} + (1-\beta_1)\left[\frac{\delta L}{\delta \mathbf{w}_k}\right] \tag{2.36}$$

$$\mathbf{v}_k = \beta_1 \mathbf{v}_{k-1} + (1-\beta_2)\left[\frac{\delta L}{\delta \mathbf{w}_k}\right]^2 \tag{2.37}$$

Batch normalization is also commonly used to optimize training by introducing a smoother optimization landscape, enabling faster and more stable training [50, 51]. This technique is commonly used between the layers in the NN to re-centre and re-scale the input. The basic formula for batch normalization is given as

$$\mathbf{x}_{new} = \gamma \frac{\mathbf{x}_B - E[\mathbf{x}]}{\sqrt{Var[\mathbf{x}] + \epsilon}} + \beta \tag{2.38}$$

where $\mathbf{x}_{new}$ is the new value of the output batch $\mathbf{x}_B$ of the hidden layer, $E[\mathbf{x}_B]$ and $Var[\mathbf{x}_B]$ are the mean and variance of the batch $\mathbf{x}_B$. The learnable parameters,γ and β control the scale and shift to the output of the layer.

Another optimization technique is early stopping, where the training stops after reaching a specific criterion. This technique has been shown to increase the performance and computational cost [52]. The early stopping criterion can be set to a particular epoch L_{max} or to when the average variation in loss $\mathcal{L}$ has reached a threshold σ within the last I epochs as follows

$$\frac{1}{I}\sum_{i=k-I}^{k} |\mathcal{L}^{i+1} - \mathcal{L}^i| < \sigma. \tag{2.39}$$

where k signifies the number of epochs.

2.6.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a special kind of feedforward network that use convolutional filters, allowing the network to learn more abstract features with fewer parameters [46].

Convolutional layers introduce sparse connectivity, parameter sharing and equivalent representation [46]. Sparse connectivity is introduced using convolutional kernels that only connect some neurons between each layer, providing indirect connections between the input and output layers. Both parameter sharing and sparse connectivity reduce the complexity and computational cost because they result in fewer parameters. Parameter sharing also allows the network to be equivariant to the translation of the input since the same features can be learned at different locations.

The convolutional filters can reduce spatial dimensionality using the kernel filters defined by kernel size, padding and stride. Given a 2D image of dimension $H \times W$ and a kernel filter with kernel size k, padding p and stride s, the output dimension after the convolutional filter is given by

$$H_{out} = \frac{H + 2 \cdot p - k}{s + 1} \tag{2.40}$$

$$W_{out} = \frac{W + 2 \cdot p - k}{s + 1} \tag{2.41}$$

where H_{out} and W_{out} are the new dimensions of the 2D image.

Additionally, the convolutional filter is defined by the amount of input and output channels, which indicate the input and output spectral dimension [46]. Hence, the CNN can be used to reduce the spectral dimensionality of a hyperspectral image. Reducing the number of channels within a CNN reduces the computational complexity since this reduces the number of filters to be learned. Hence there are fewer parameters to train [46]. This reduction can also help with generalization, removing redundant information and avoiding overfitting. However, it can also lead to a loss of information, which can decrease performance.

Skip connections were introduced by the residual neural network (ResNet) [53], where the output of a layer is concatenated with the output of a layer much deeper in the network. This technique has demonstrated improved performance of CNNs by addressing the issue of vanishing gradients and enhancing the information flow through the network. Figure 2.17 shows an example of a skip connection, where the input $\mathbf{x}$ is concatenated with the output of a layer deeper into the network.

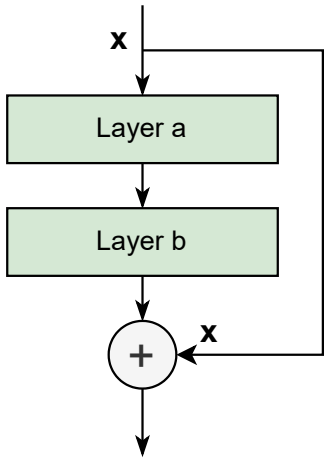


Figure 2.17: Example of the skip connection method.

2.6.3 Convolutional Autoencoder

An Autoencoder (AE) is an unsupervised learning technique that has shown promising results within anomaly detection [54, 5, 13]. AEs consists of an encoder, decoder and a dimensional bottleneck, forcing a compressed representation of the encoder's input. This bottleneck is often referred to as the image code or latent code [55]. The AE learns to encode the most critical information in the image code so that the decoder is able to reconstruct the input data from the image code efficiently. AEs can be used for many applications, such as image compression, denoising and dimensionality reduction. A basic architecture of an AE is shown in Figure 2.18, consisting of an encoder $\mathbf{X}$, decoder $\mathbf{Y}$ and image code $\mathbf{Z}$.

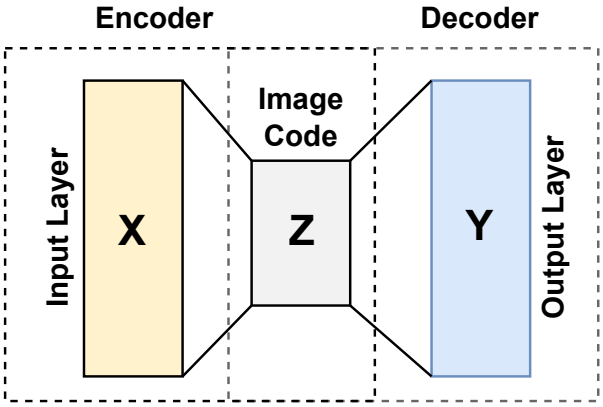


Figure 2.18: Basic architecture of an autoencoder.

The encoder and decoder of the AE are constructed using neural networks, where a possible solution is to use convolutional neural networks [34, 5], which results in a convolutional AE. Introducing convolutional layers into the AE allows the network to learn more abstract features. Further, autoencoders are trained using loss functions like the feedforward NN. However, it is important to emphasize that autoencoders operate in an unsupervised manner, where the loss is computed by comparing the input of the network $\mathbf{X}$ with its corresponding output $\hat{\mathbf{X}}$, denoted as the reconstructed background.

As mentioned previously, the kernel filters in the CNN can be used to reduce the spatial dimensionality of the input. When applied to the encoder, the decoder has to increase the dimensionality back to the original input dimension. Image interpolation can be used for this up-sampling in the decoder [56]. Nearest neighbor interpolation is one such technique, where the output pixel is assigned the value of the nearest sample point in the input image [57]. This method ensures that the new values are already in the dataset.

2.6.4 AUTO-AD

The Autonomous Hyperspectral Anomaly Detection Network (AUTO-AD) is a fully convolutional AE that utilizes skip connections to reconstruct the background to separate the anomalies. This unsupervised model uses a residual error-based method as described in Figure 2.6 [5]. Figure 2.19 visualizes the AUTO-AD framework.

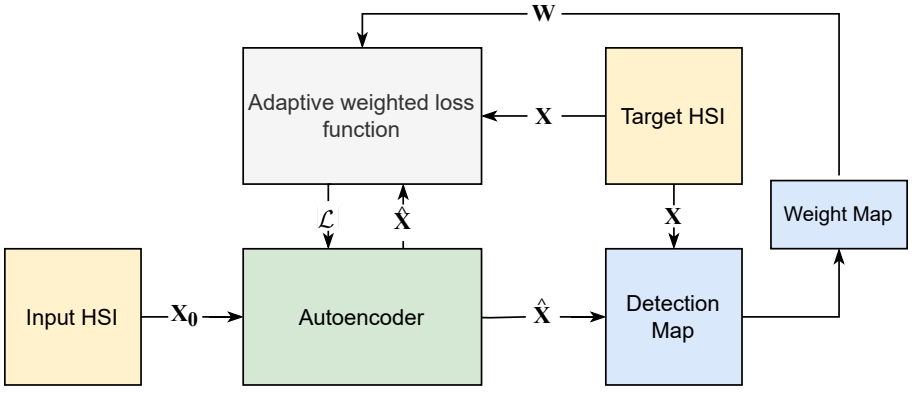


Figure 2.19: Architecture of the AUTO-AD model, based on the architecture in [5].

The AUTO-AD model uses uniform noise as input to the autoencoder instead of using the dataset $\mathbf{X}$. Hence the input to the AE denoted as the input HSI $\mathbf{X}_0 \in \mathbb{R}^{H \times W \times B}$, is a hypercube filled with uniform noise $\mathcal{U} \in [0, 1]$. This means that the AE learns to reconstruct the background of the HSI using uniform noise as input. The output of the autoencoder is the reconstructed

background $\widehat{\mathbf{X}}$, which, together with the dataset, denoted as the target HSI $\mathbf{X}$, is used to calculate the loss and detection map.

Research has shown that the AE learns to reconstruct the anomalies after many epochs. To address this issue, an adaptive-weighted loss function is applied to reduce the feature representation of the anomalies further [5]. This loss function uses the reconstruction errors as the adaptive weight.

The reconstructed error $e_{i,j}$ in the spatial position (i, j), is measured between the reconstructed background $\hat{\mathbf{x}}_{i,j}$ and the HSI $\mathbf{x}_{i,j}$ as follows:

$$e_{i,j} = ||\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}||_2^2. \tag{2.42}$$

The adaptive weight-loss function is then given as

$$\mathcal{L} = \sum_{i=1}^{H} \sum_{i=j}^{W} ||(\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}) w_{i,j}||_2^2 \tag{2.43}$$

where $w_{i,j}$ is the weight that can be calculated as

$$w_{i,j} = \max(\text{vec}(\mathbf{E})) - e_{i,j}, \tag{2.44}$$

where $e_{i,j}$ denotes the error between the input and output of the network, also called the reconstruction loss, and $\text{vec}(\cdot)$ denotes the vectorization of the argument matrix, and

$$\mathbf{E} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,W} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,W} \\ \vdots & \vdots & \ddots & \vdots \\ e_{H,1} & e_{H,2} & \cdots & e_{H,W} \end{bmatrix}. \tag{2.45}$$

The weights $w_{i,j}$ are first initialized to one, and updated after a give amount of epochs I. These weights form the weight map denoted as $\mathbf{W}$ in Figure 2.19.

Additionally, the AUTO-AD model employs the early stopping method in (2.39), which guarantees that the training stops after reaching one of two criteria. The first criterion is reaching a particular epoch L_{max}, and the second is to stop the training after the average variation in loss has reached a threshold σ. This method has been shown to increase detection performance and reduce the computational cost. The ADAM optimizer described in (2.33) is utilized for backpropagation to optimize the training process further.

The structure of the autoencoder of the AUTO-AD is presented in Figure 2.20.

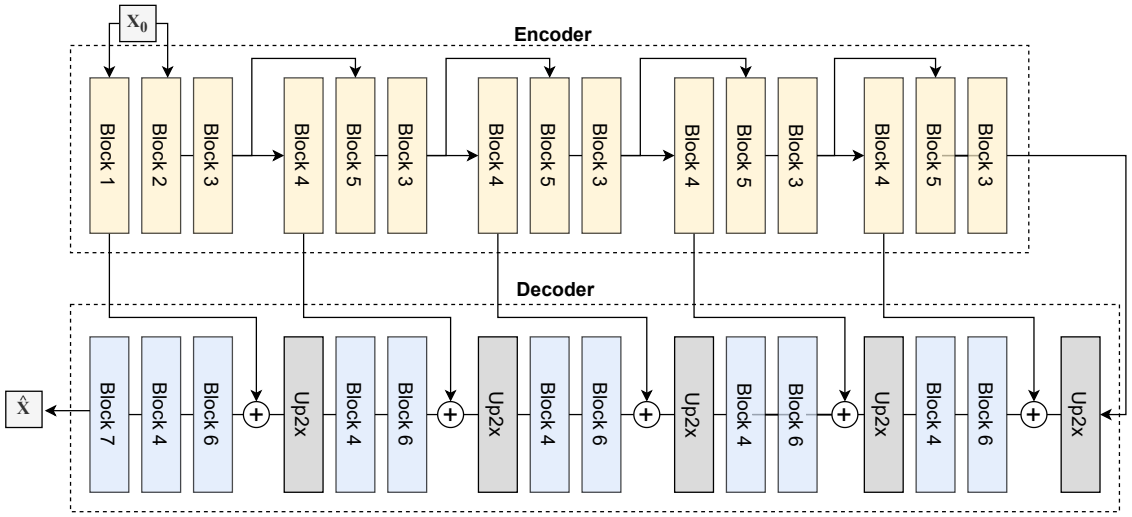


Figure 2.20: Block diagram visualizing the AE structure of the AUTO-AD. The Figure is inspired by [5].

The AE structure consists of 26 blocks and four up-sampling blocks utilizing the nearest neighbour interpolation technique. Each block contains a convolutional layer and an activation

function. Additionally, every block applies a batch normalization block as described in (2.38, except for block 7. Batch normalization enhances the stability of the training process by ensuring that the input to the convolutional layer is normalized. The LeakyReLU activation function in (2.28), is utilized in every layer block except for block seven which utilizes the Sigmoid activation function in (2.27).

In total there are seven different types of blocks which are presented in Figure 2.21. The convolutional layer is denoted by *conv*, consisting of number of input and output channels, kernel size, stride and padding.

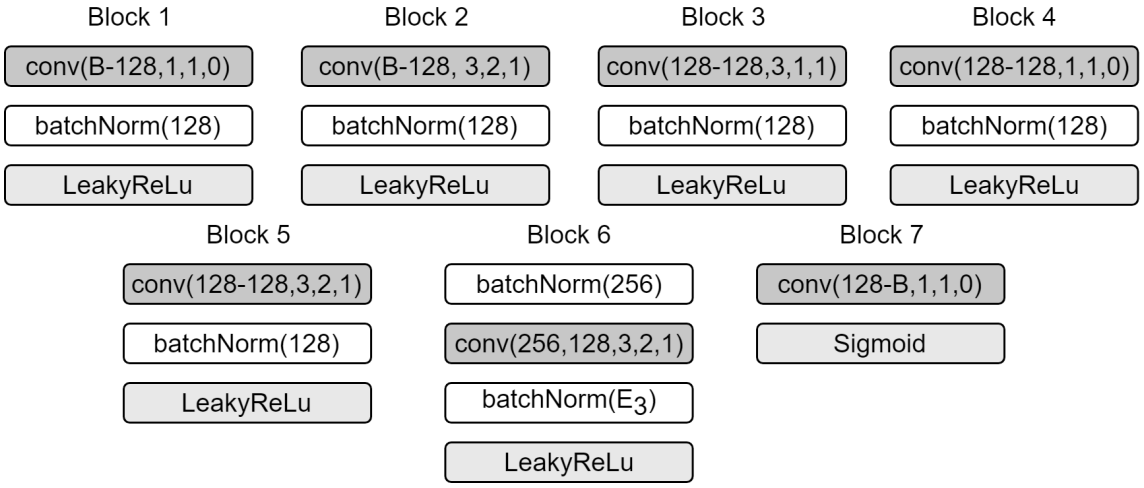


Figure 2.21: Figure presenting the contents of each block in the autoencoder of the AUTO-AD model.

The encoder consists of 15 blocks that are split into five sections. The input HSI $\mathbf{X}_0$ is passed as the input to the first layers of the encoder, blocks 1 and 2, with a spectral dimension of B and reduced to 128 by the convolutional layers. It should be emphasized that the reduction of spectral dimensionality is only carried out in the initial two blocks of the encoder. In contrast, the subsequent blocks in the encoder maintain the same number of input and output channels.

In block 1, a kernel of size 1×1 with a stride of 1 and zero padding is applied, maintaining the spatial dimension. The resulting output is then forwarded via skip connections to the decoder. Block 2 employs a kernel of size 3×3 with a stride of 2 and padding of 1, reducing the spatial dimension according to (2.40) and (2.41). The output is passed on to block 3, consisting of a convolutional layer with the same kernel parameters as block 2 but with a stride of 1, ensuring that the spatial dimension remains the same.

The output of block 3 is used as input to blocks 4 and 5 in the first four sections and as input to the first up-sampling block in the decoder in the last section. In block 4 the convolutional layer consists of a 1×1 kernel with a stride of 1 and padding of 0, while block 5 consists of the same kernel parameters as block 2. In the encoder, the output of block 4 is propagated through skip connections to the decoder, while in the decoder, it is utilized as input to the up-sampling block, which is subsequently concatenated with the skip connections from the encoder The output of block 5 is sent as input to block 3.

The decoder consists of 11 blocks with an additional four up-sampling blocks, which are used to ensure the same spatial dimension between the concatenation. After each concatenation, the spectral dimension is doubled to 256 and passed as input to block 6 where a kernel of size 3×3 with a padding of 1 and stride of 2 is applied. These kernel parameters reduce the spatial dimension. Additionally, the spectral dimension is reduced from 256 to 128. The output of block 6 is used as input to block 4, which applies the same convolutional layer as in the encoder. Block 7 is the final block in the decoder, which includes a convolutional layer that preserves the spatial dimension and expands the spectral dimension from 128 to the original size B. The output of this layer is then passed through a Sigmoid activation function.

2.7 Evaluation Metric

The performance of the anomaly detection models is assessed using a 2D Receiver Operating Characteristic (ROC) curve and the corresponding Area Under Curve (AUC) values [58]. The ROC curve presents the trade-off between the true positive rate, also defined as the detection probability P_D and false positive rate, known as the false alarm probability P_F. The optimal outcome is to achieve $P_D = 1$ and $P_F = 0$. Figure 2.22 presents a graphical representation of an ROC curve, where two models are compared. Model 1 shows a higher P_D rate and a lower P_F rate than Model 2, resulting in a higher AUC score.

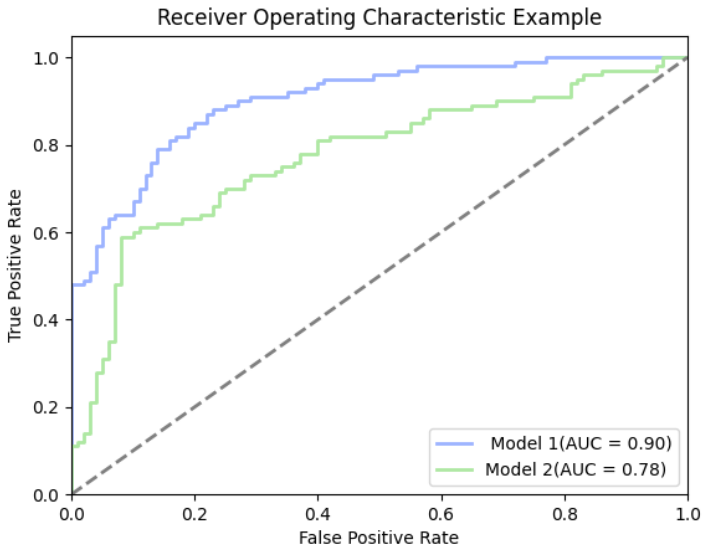


Figure 2.22: Illustration of a basic ROC.

Within anomaly detection, the false alarm probability is defined as the number of wrongly classified anomalous pixels, and the detection probability is defined as the number of correctly classified anomalous pixels. P_D and P_F are defined as:

$$P_D = \frac{N_D}{N_O} \tag{2.46}$$

$$P_F = \frac{N_F}{N} \tag{2.47}$$

N_D represents the number of correctly classified anomaly pixels, N_O represents the total number of pixels classified as anomalies, N_F represents the number of falsely classified anomaly pixels, and N represents the total number of pixels in the hyperspectral image. The AUC is calculated by taking the area under the ROC curve.

Chapter 3

Methodology

This chapter provides the methodology of the contributions to this master thesis. As stated in the introduction, the main contribution to this thesis is to enhance the Autonomous Hyperspectral Anomaly Detection Network (AUTO-AD) model with regards to detection performance and computational cost. As described in the previous chapter, the model consists of a considerably large number of convolutional layers which increases the computational cost. Additionally, the model does not apply any pre-processing methods, which could potentially enhance both the detection performance and the computational cost. Based on these observations, three contributions are proposed. This chapter will give a comprehensive description of all three proposed contributions, including the motivation behind the proposed changes.

3.1 AUTO-AD with Pre-processing (AUTO-AD$^+$)

The first contribution of this master's thesis aims to enhance the detection performance of the AUTO-AD by introducing Kernel PCA-based pre-processing. This proposed model is denoted as the AUTO-AD$^+$ model. KPCA transforms the HSI into a higher dimensional feature space, which can enhance the representation of the the anomalies that are not apparent in the raw data of the HSI. This enhanced representation can result in a higher separability between the anomalies and background pixels, leading to improved detection performance.

A basic structure of the proposed AUTO-AD$^+$ is presented in Figure 3.1, where the red block illustrates the KPCA-based pre-processing block. Furthermore, the AE is constructed according to the structure presented by the AUTO-AD in Section 2.6.4.

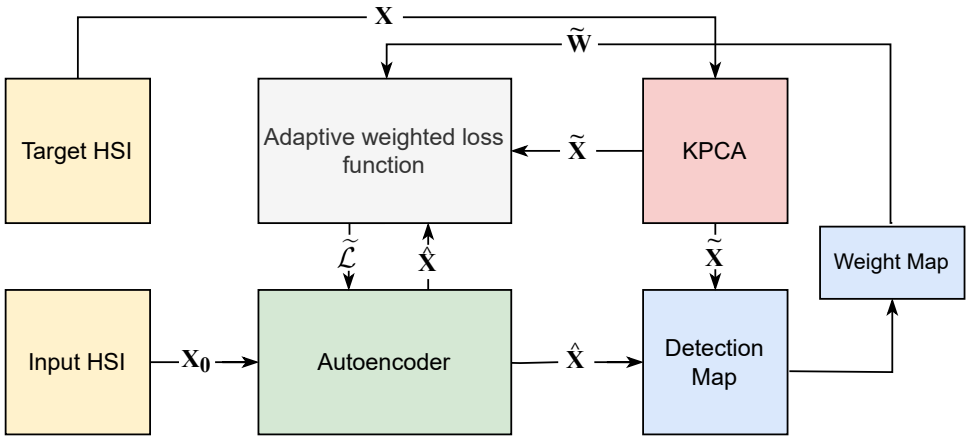


Figure 3.1: Figure illustrating the framework of the AUTO-AD$^+$ model.

The figure illustrates that the target HSI $\mathbf{X} \in \mathbb{R}^{H \times W \times B}$ is passed as input to the KPCA-based pre-processing block. This results in a transformed HSI $\widetilde{\mathbf{X}} \in \mathbb{R}^{H \times W \times \xi}$, where ξ represents the number of PCs. Further, the input of the autoencoder is the input HSI $\mathbf{X}_0 \in \mathbb{R}^{H \times W \times \xi}$. The

output of the AE is the reconstructed background $\widehat{\mathbf{X}} \in \mathbb{R}^{H \times W \times \xi}$, which is used to compute the detection map and the loss.

For the AUTO-AD, the detection map is computed using the matrix $\mathbf{E} = \{e_{i,j}\}_{i,j}^{H,W}$ in (2.45). However, in this configuration, the pre-processed HSI, $\widetilde{\mathbf{X}}$, replaces the target HSI, $\mathbf{X}$. Therefore the new reconstruction error $\widetilde{e}_{i,j}$ is computed as follows

$$\widetilde{e}_{i,j} = ||\widetilde{\mathbf{x}}_{i,j} - \hat{\mathbf{x}}_{i,j}||_2^2 \tag{3.1}$$

where $\widetilde{\mathbf{x}}_{i,j}$ and $\hat{\mathbf{x}}_{i,j}$ are of dimension $\xi \times 1$. This results in a new weight matrix $\widetilde{\mathbf{W}} = \{\widetilde{w}_{i,j}\}_{i,j}^{H,W}$, where $\widetilde{w}_{i,j} = \max(\text{vec}(\widetilde{\mathbf{E}})) - \widetilde{e}_{i,j}$, and $\widetilde{\mathbf{E}}$ is the matrix containing the reconstruction errors $\widetilde{e}_{i,j}$. The adaptive-weighted loss function is then given by

$$\widetilde{\mathcal{L}} = \sum_{i=1}^{H} \sum_{i=j}^{W} ||(\widetilde{\mathbf{x}}_{i,j} - \hat{\mathbf{x}}_{i,j})\widetilde{w}_{i,j}||_2^2 \tag{3.2}$$

The AUTO-AD$^+$, like the AUTO-AD, utilizes the early stopping method and the ADAM optimizer. These models optimize the training process of the AE and have been shown to increase detection performance and reduce computational cost.

3.2 Lightweight AUTO-AD (LW-AUTO-AD)

The second contribution of this master's thesis focuses on reducing the computational complexity of the AUTO-AD by introducing a lightweight autoencoder architecture. This model is denoted as the Lightweight AUTO-AD (LW-AUTO-AD). The main objective is to design an AE that requires fewer computational resources while maintaining or increasing the detection performance. This requirement becomes particularly crucial if the HAD model is intended for potential integration into the onboard processing of a satellite where the system has limited resources.

Figure 3.2 presents the new framework of the LW-AUTO-AD, where the proposed lightweight autoencoder is represented by the red block. This HAD model also utilizes the early stopping method and the ADAM.

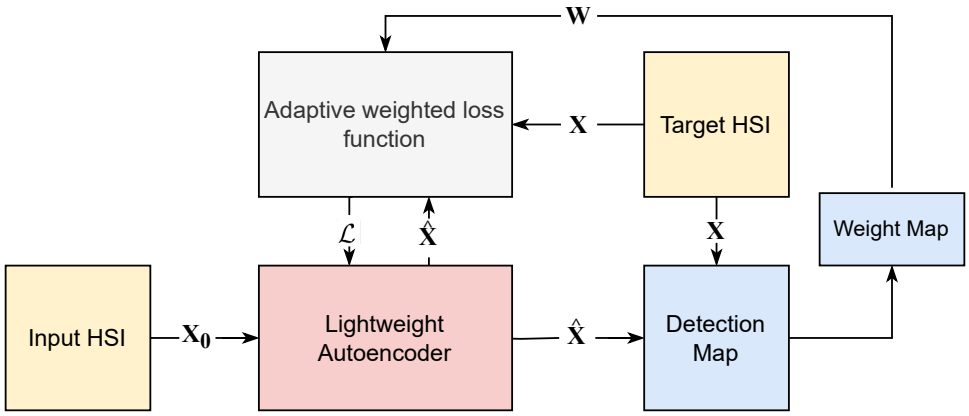


Figure 3.2: The figure illustrates the structure of proposed LW-AUTO-AD model.

The framework consists of an input HSI $\mathbf{X}_0 \in \mathbb{R}^{H \times W \times B}$ which consists of a hypercube with uniform noise $\mathcal{U} \in [0, 0.1]$. The target HSI is the dataset $\mathbf{X} \in \mathbb{R}^{H \times W \times B}$. Additionally, the model incorporates the adaptive-weighted loss function $\mathcal{L}$ defined in (2.43), where the loss is computed between the reconstructed background $\widehat{\mathbf{X}} \in \mathbb{R}^{H \times W \times B}$ and the target HSI $\mathbf{X}$, using the weight matrix $\mathbf{W} \in \mathbb{R}^{H \times W}$.

The LW-AUTO-AD consists of a fully convolutional autoencoder with skip connections, consisting of 10 convolutional layers. This results in a reduction of more than half the number of convolutional layers compared to the AUTO-AD model, which consists of 26 convolutional layers. Furthermore, the encoder's convolutional layers are employed to decrease the number of spectral

bands of the HSI. This reduction is achieved by decreasing the number of channels from the input to the output of the convolutional layer. These measures simplify the network and reduce the number of learnable parameters in the AE, resulting in lower computational costs.

The architecture of the lightweight autoencoder is described in the block diagram in Figure 3.3. The AE consists of six blocks in the encoder, three blocks in the decoder, and three upsampling blocks. The figure provides a visual representation of the flow within the AE, where the input HSI $\mathbf{X}_0 \in \mathbb{R}^{H \times W \times B}$ is used as input to the 1st and 2nd block of the decoder. The decoder output is the reconstructed background $\widehat{\mathbf{X}}$.

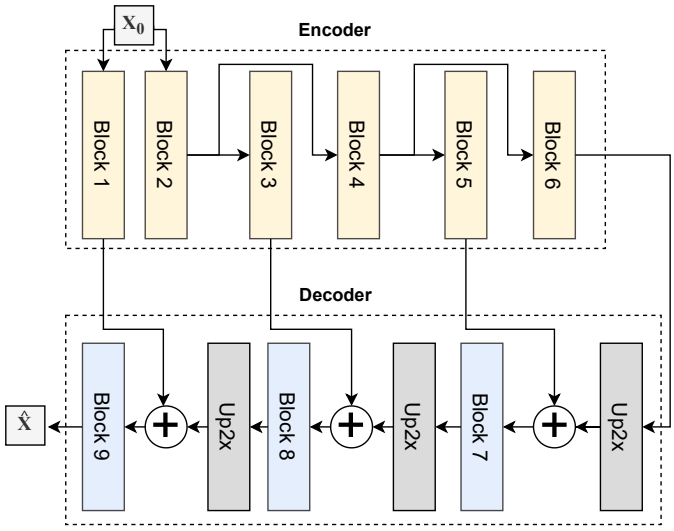


Figure 3.3: Block diagram illustrating the lightweight AE architecture.

Each block in the encoder contains a convolutional layer that reduces the number of channels from the input to the output of the layer. The number of input channels in the six blocks is given by $[B, B, E_1, E_1, E_2, E_2]$, while the number of output channels is given by $[E_1, E_1, E_2, E_2, E_3, E_3]$, respectively. These dimensions are visualized in Figure 3.4, which illustrates the contents of the blocks in the encoder. The convolutional layer is denoted as *conv*, consisting of the nr. input channels, output channels, kernel size, stride, and padding. For instance, referring to the first convolutional layer in block 1, $conv(B - E_1, 1, 1, 0)$ indicates that the layer has B input channels, E_1 output channels and applies a 3×3 kernel with a stride of 1 and padding of 0.

The number of input and output channels will be determined through experimental testing, where the optimal values will be selected based on the detection performance and computational cost. However, it is worth noting that these values will follow the condition $B \geq E_1 \geq E_2 \geq E_3$.

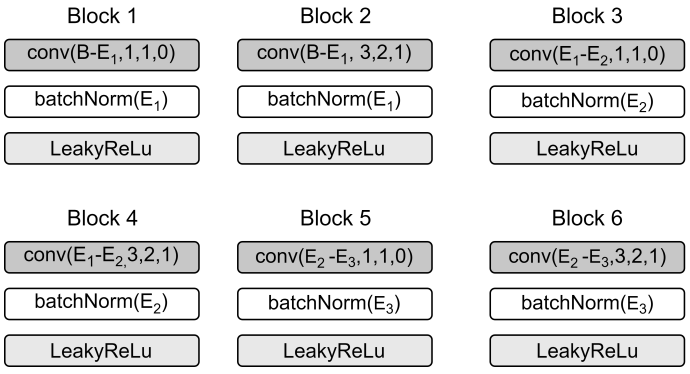


Figure 3.4: Figure presenting the contents of the blocks in the encoder.

The autoencoder incorporates three skip connections by concatenating the encoder and decoder outputs. These allow the encoder to complement the decoder with spatial information. Figure 3.3 emphasizes that the encoder's 1st, 3rd, and 5th blocks are involved in the concatenation

process. The resulting dimensions after concatenation are given by

$$F_i = \begin{cases} E_3 + E_3 & \text{i} = 1, \\ E_2 + D_1 & \text{i} = 2, \\ E_1 + D_2 & \text{i} = 3. \end{cases} \tag{3.3}$$

where F_1, F_2, F_3 denotes the number of input channels to the three blocks in the decoder, 7,8 and 9, respectively. D_1 and D_2 represents the number of output channels in block 7 and 8, whereas the number of output channels in the fist convolutional layer in block 9 is denoted by D_3.

Block 9 contains a second convolutional layer which ensures that the reconstructed background $\widehat{\mathbf{X}}$ contains the same number of spectral bands as the input $\mathbf{X}_0$ and target HSI $\mathbf{X}$. Hence, this last layer increases the number of input channels D_3 to B.

The contents of each block in the decoder are presented in Figure 3.5. The number of input and output channels in the decoder will also be determined through experimental testing, however, following the condition $B \geq D_3 \geq D_2 \geq D_1$.

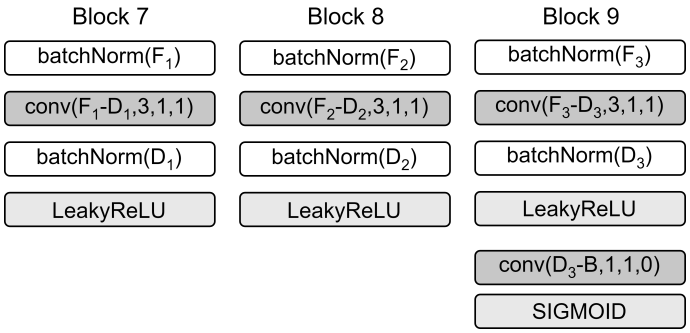


Figure 3.5: Figure presenting the contents of the blocks in the decoder.

As illustrated in Figure 3.3, the decoder consists of three blocks containing convolutional layers. In addition to these blocks, the decoder consists of three blocks that perform spatial up-sampling using nearest neighbour interpolation. This up-sampling is crucial for the decoder as it serves two essential purposes. Firstly, to ensure that the decoder's spatial dimension matches those of the encoder's corresponding spatial dimensions, which are utilized in the skip-connections. Secondly, the up-sampling operation guarantees that the reconstructed output of the decoder contains the same spatial dimension as the input of the AE.

Further, as described in Figure 3.4 each block in the encoder contains batch normalization and the LeakyReLU activation function. Batch normalization stabilizes the training process, while the LeakyReLU ensures that non-linearity is introduced to the network.

Blocks 1,3 and 5 complement the decoder with spatial details through skip connections. Hence, the spatial dimension must remain the same. By applying a convolutional filter with a 1×1 kernel, with a stride of 1 and padding of 0, the resulting spatial dimension remains the same as the layer's input.

On the contrary, blocks 2,4, and 6 all perform spatial dimensionality reduction by applying convolutional filters with a 3×3 kernel with a stride of 2 and a padding of 1 in both height and width dimensions. The new spatial dimension H_{out} and W_{out} can be found using (2.40) and (2.41). Spatial dimensionality reduction reduces the number of parameters of the network, which reduces the computational cost. Additionally, spatial dimensionality reduction improves the network's ability to extract more compact and informative representations [59]. This reduction can enhance the anomalies' representation and increase detection performance.

Blocks 7 and 8 contain two batch normalization steps, one convolutional layer and the LeakyReLU activation function, whereas block 9 contains an extra convolutional filter with the Sigmoid activation function. The Sigmoid activation function ensures that the network's output, the reconstructed background, is between $[0, 1]$.

Each block in the decoder applies a convolutional filter with a 3×3 kernel, a stride and padding of 1, which ensures that the spatial dimension is unchanged. To ensure that the input to the convolutional layers is normalized, batch normalization is applied before the convolutional layer in each block. Normalization has been shown to stabilize the input, which can lead to a more

efficient and effective model. Batch normalization is also applied to the convolutional layer output with the same reasoning. As described above, the LeakyReLU activation function is employed after batch normalization.

Block 9 consists of two convolutional layers. The first layer uses the same kernel parameters as in blocks 7 and 8, applying batch normalization to the input and output of the convolutional layer and the LeakyReLU activation function. The second convolutional layer of block 9 uses a 1×1 kernel, with a stride of 1 and padding of 0, followed by the Sigmoid activation.

3.3 LW-AUTO-AD with Pre-processing (LW-AUTO-AD$^+$)

The third and final contribution of this master's thesis integrates the Lightweight AUTO-AD model with a pre-processing block to enhance detection performance while reducing computational complexity. This combined model is referred to as the LW-AUTO-AD$^+$ model. Pre-processing can enhance the accuracy of the detection map by eliminating noise and irrelevant information and revealing the intricate underlying structure of the data. These factors can benefit the AE by making anomalies more noticeable in the dataset, thereby making the anomalies easier to differentiate from the background. As a result, the autoencoder can better prevent reconstructing anomalies, as it will not perceive them as part of the background.

The LW-AUTO-AD$^+$ model is presented in Figure 3.6, where the contributions of this thesis are represented by red blocks, namely the lightweight autoencoder and the pre-processing block. LW-AUTO-AD$^+$, like the AUTO-AD and LW-AUTO-AD model, applies the early stopping scheme and ADAM for optimizing the training.

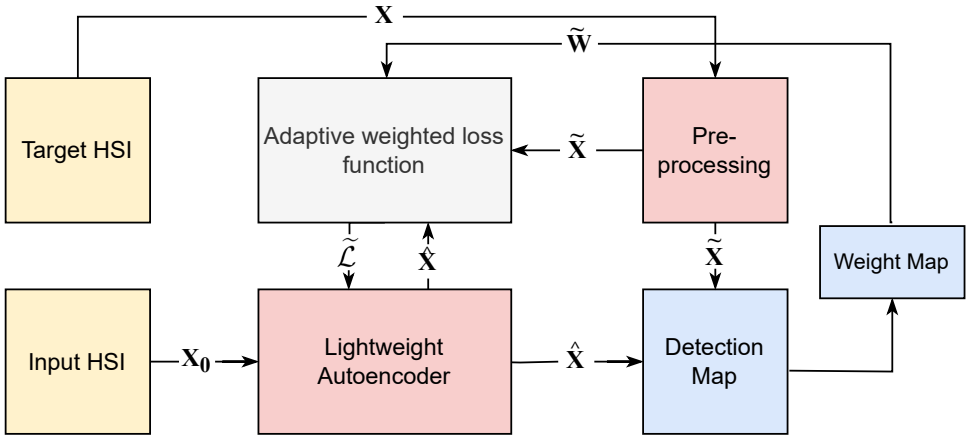


Figure 3.6: Figure illustrating the proposed LW-AUTO-AD$^+$ model.

Like in the proposed AUTO-AD$^+$, the target HSI $\mathbf{X} \in \mathbb{R}^{H \times W \times B}$ is passed as input to the pre-processing block, resulting in a new data tensor $\widetilde{\mathbf{X}} \in \mathbb{R}^{H \times W \times \xi}$, where ξ is determined by the number of PCs utilized by the pre-processing method. The input of the network is the input HSI$\mathbf{X}_0 \in \mathbb{R}^{H \times W \times \xi}$ and the output is the reconstructed background $\widehat{\mathbf{X}} \in \mathbb{R}^{H \times W \times \xi}$.

Further, the LW-AUTO-AD$^+$ applies the adaptive-weighted loss function $\widetilde{\mathcal{L}}$ described in (3.2). The loss function determines the loss between the target HSI $\widetilde{\mathbf{X}}$ and reconstructed background $\widehat{\mathbf{X}}$, and uses the weight $\widetilde{\mathbf{W}} \in \mathbb{R}^{H \times W}$ to reduce the feature representation of the anomalies.

Three pre-processing methods are tested: Principal Component Analysis, Kernel PCA, and Random Fourier Features KPCA. Each method has advantages and disadvantages regarding detection performance and computational cost. PCA is the simplest method among these techniques since it does not necessitate the computation of the kernel matrix, which adds an extra $O(N^2)$ computations, where N denotes the number of pixels. However, PCA is designed for linearly separable datasets and may be less effective on hyperspectral images.

KPCA has the advantage of capturing non-linear relationships between the features that may not be present in the raw data. These relations can help reveal information about the complex underlying structure of the dataset. However, as previously discussed, KPCA comes at the expense of a high computational cost, mainly when dealing with large datasets where the number of pixels

N is substantial.

The RFF-KPCA -based pre-processing method can be considered a compromise between PCA and KPCA, as it aims to approximate the KPCA while avoiding the computational cost of computing the kernel matrix.

In addition, this thesis proposes to apply an MKL-based pre-processing. MKL-based pre-processing applies multiple kernel functions to capture various aspects of the complex structure of the dataset. This can enhance the separation between background and anomalies to a greater extent than standard KPCA or RFF-KPCA -based pre-processing methods, improving detection performance.

The MKL-based pre-processing method follows the framework described in Figure 2.9, where multiple pre-processing and anomaly detection blocks are employed in parallel. This results in multiple detection maps fused using the decision fusion in (2.17) with the fusion weight v_1 to obtain the final detection map.

Figure 3.7 illustrates how the MKL-based pre-processing method is applied to LW-AUTO-AD$^+$ when utilizing two pre-processing approaches, LW-AUTO-AD$^+$ (1) and LW-AUTO-AD$^+$ (2). To make the figure more concise, the LW-AUTO-AD$^+$ models are presented in one block, where the contents of these blocks are described in Figure 3.6.

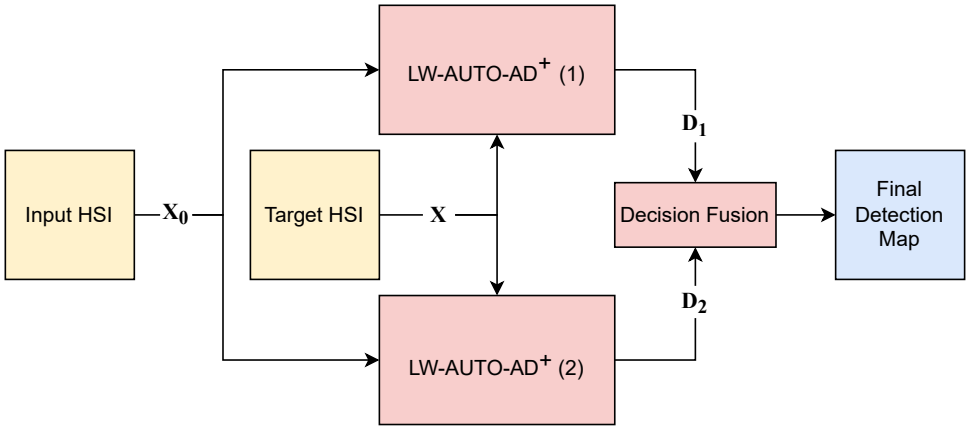


Figure 3.7: Figure illustrating the LW-AUTO-AD$^+$ together with the MKL-based pre-processing method.

The input HSI $\mathbf{X}_0$ and target HSI $\mathbf{X}$ are the same as in the LW-AUTO-AD$^+$. Each LW-AUTO-AD$^+$ model generates a detection map, $\mathbf{D}_1 \in \mathbb{R}^{H \times W}$ and $\mathbf{D}_2 \in \mathbb{R}^{H \times W}$, determined by the pre-processing method applied in the model. The final detection map $\mathbf{D} \in \mathbb{R}^{H \times W}$ is, as mentioned, found using a decision fusion.

The pre-processing blocks within the LW-AUTO-AD$^+$ models can employ KPCA-based methods that utilize various kernel functions, including the RBF and Laplacian kernels described in (2.8) and (2.10). This approach is referred to as Multi KPCA (MKPCA)-based pre-processing. However, the computation of two kernel matrices in the LW-AUTO-AD$^+$ model increases the computational cost significantly. Therefore, an alternative approach is utilizing the RFF-KPCA-based pre-processing method using multiple distribution functions outlined in Table 2.1. This approach is denoted as RFF-MKPCA-based pre-processing.

Chapter 4

Results

This chapter presents the results for the proposed contributions of this master thesis. The results will be assessed based on two primary factors: the detection performance, which will be measured using the AUC (Area Under the Curve) score, and the time cost. It is important to note that the time cost serves as an indicator of the computational cost. By considering both the detection performance and the time cost, a comprehensive evaluation of the proposed approach can be obtained.

4.1 Test Setup

Table 4.1 describes the setup used in this masters thesis.

Table 4.1: Configurations of the test setup.

Item	**Specification**
Processor	12th Gen Intel(R) Core(TM) i5-12500H, 2500 Mhz, 12 Core(s), 16 Logical Processor(s)
RAM	16 GB
OS	Microsoft Windows 11
Matlab Version	MATLAB R2022b
Python Version	Python 3.9.12

4.2 Datasets

The Airport-Beach-Urban (ABU) dataset is used for testing the HAD models. This dataset was made available by [29]. The features of the datasets, including the capture place, resolution (in pixels) and the number of spectral bands, are presented in Table 4.2. All datasets, except beach scene 4, have been manually extracted from the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) . Beach scene 4 has been taken from the Reflective Optical System Imaging Spectrometer (ROSIS-03).

Table 4.2: Airport-Beach-Urban dataset features. The information is taken from [60].

Airport scene	**Captured Place**	**Resolution**	**Band**
1	Los Angelas	100 × 100	205
2	Los Angelas	100 × 100	205
3	Los Angelas	100 × 100	205
4	Gulfport	100 × 100	191
Beach scene	**Captured Place**	**Resolution**	**Band**
1	Cat Island	150 × 150	188
2	San Diego	100 × 100	193
3	Bay Champagne	100 × 100	188
4	Pavia	150 × 150	102
Urban scene	**Captured Place**	**Resolution**	**Band**
1	Texas Coast	100 × 100	204
2	Texas Coast	100 × 100	207
3	Gainesville	100 × 100	191
4	Los Angelas	100 × 100	205
5	Los Angelas	100 × 100	205

4.3 Dataset Analysis

A research analysis is conducted in order to gain a more thorough comprehension of the ABU datasets by observing the anomaly-to-background pixel ratio, number of anomalous pixels and number of anomalies This analysis aims to understand better the distribution of the anomalous and background pixels, which may be used to analyse the results of the HAD models. In addition, a comparison of the spectral signatures between the anomalies and background pixels is conducted. The spectral signatures are presented using a plot which displays the mean and standard deviation of the anomalies and background pixels over all spectral bands.

An overview of the results from the first part of the analysis is shown in Table 4.3. The first column displays the dataset scenes, the second column gives the anomaly-to-background pixel ratio, the third column shows the number of anomalous pixels present in the dataset, and the last column shows the number of anomaly objects within the dataset.

Table 4.3: Data set analysis of the ABU dataset from Table 4.2.

Airport Scenes	**Anomaly Ratio**	**Number of Anomalous Pixels**	**Number of anomalies**
1	0.0144	144	13
2	0.0087	87	2
3	0.0170	170	16
4	0.0060	60	3
Beach scenes			
1	0.0008	19	1
2	0.0202	202	58
3	0.0011	11	1
4	0.0030	68	7
Urban scenes			
1	0.0067	67	9
2	0.0155	155	20
3	0.0052	52	11
4	0.0272	272	25
5	0.0232	232	30

The ground truth plots of the airport scenes are presented in Figure 4.1, where the white pixels represent the anomalies, while the black areas represent the background. Table 4.3 demonstrates

that the 1st and 2nd scenes have similar values for all three categories. Both scenes have an anomaly-to-background ratio of around 1.5%, which is around 150 anomaly pixels. The 1st scene has 13 anomaly objects, while the 3rd has 16, which can be seen in the ground truth plots. Both scenes contain varying sizes of anomalies, with the highest concentration around the centre of the image.

The 2nd and 4th scenes contain fewer anomaly pixels, with a ratio of less than 1%, and only 2 and 3 anomaly objects, respectively. The plots show that the anomaly objects in all the airport scenes are aeroplanes with varying sizes.

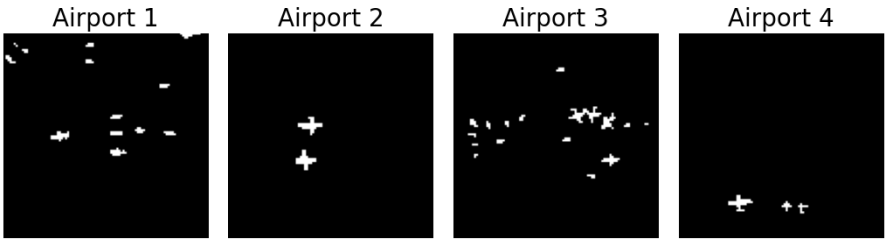


Figure 4.1: Ground truth plots of the Airport scenes.

Figure 4.2 presents the ground truth plots for the beach scenes. Results in Table 4.3 show that the 2nd beach scene differs the most from the other beach scenes with 58 anomaly objects, compared to the 1st and 3rd scene, which contains one anomaly object, and the 4th scene containing seven objects. Compared to the airport and urban scenes, the 2nd beach scenes contain the highest number of anomalies.

The ground truth plots for the beach scenes show that the anomaly objects are all of a considerably small size, which coincides with the low anomaly-to-background ratio. However, this is different for the 2nd scene, which has a ratio of 2%

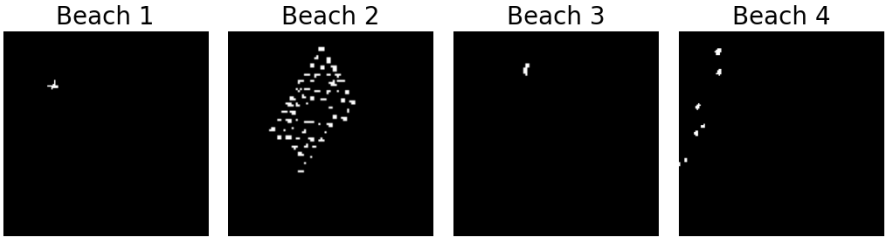


Figure 4.2: Ground truth plots of the Beach scenes.

Figure 4.2 presents the ground truth plots for the urban scenes. Based on the results in Table 4.3, it can be seen that the 1st and 3rd urban scenes are similar in both ratio and amount of anomaly objects, with 9 and 11 objects, respectively. However, the anomaly objects in the 3rd scene are smaller than in the 1st scene. Both the 4th and 5th scene contains a considerably large number of anomaly objects, which coincides with the high anomaly-to-background ratio of 2.7% and 2.3%, respectively. The second scene has fewer anomaly objects than the 4th and 5th scenes, with 20 anomaly objects and a 0.5% ratio.

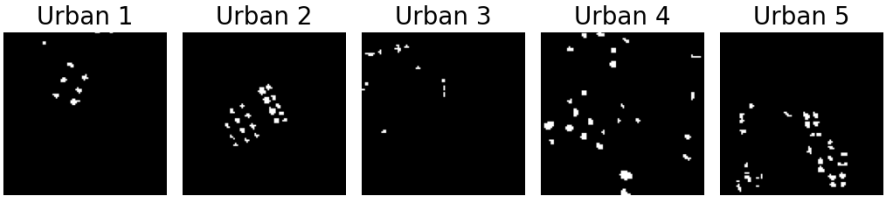


Figure 4.3: Ground truth plots of the Urban scenes.

Plots of the spectral signatures for some ABU datasets are presented for further analysis. These plots consist of the mean and standard deviation of the anomaly and background pixels. The separability properties of the dataset are indicated by looking at the overlap between the mean and standard deviation in the plots.

The first plot is of the 1st and 4th airport scenes, presented in Figure 4.4.

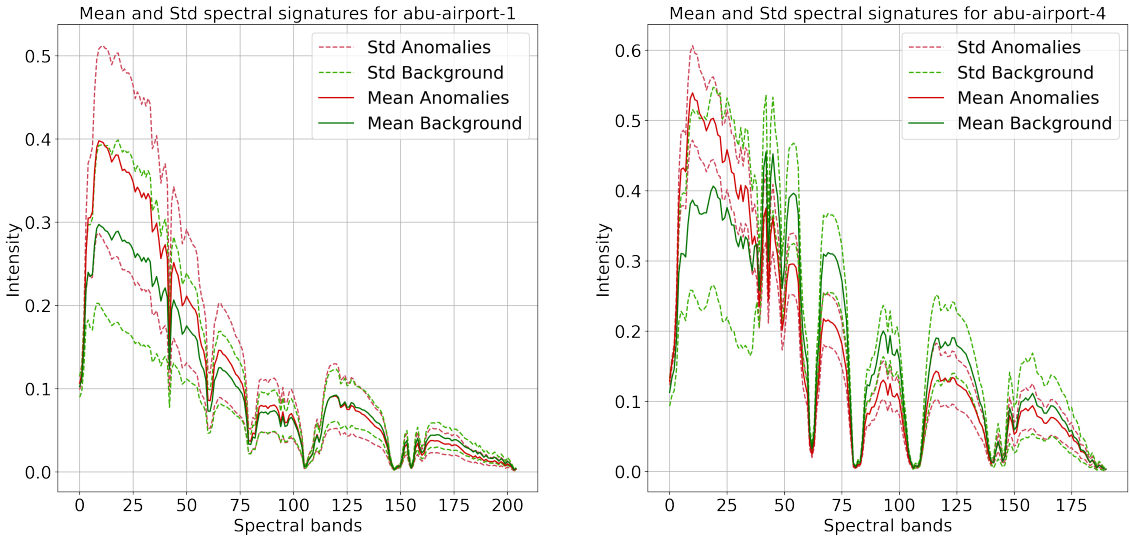


Figure 4.4: Spectral signatures for the 1st and 4th airport scenes.

The plots show that both spectral signatures follow a similar structure, with the highest standard deviation and mean values in the first 50 bands and peaks and valleys around the same areas. It is worth noting that the 1st scene has six more spectral bands than the 4th scene.

For the 1st scene, the mean and standard deviation of the anomalies and background have a high degree of overlap after the 50th band. For the 4th scene, there is a lower degree of overlap, both before and after the 50th band. The mean of the background pixels is mainly outside the standard deviation of the anomalies, which indicates that this scene will have a higher degree of separability.

The spectral signatures for the 1st and 2nd beach scenes is shown in Figure 4.5.

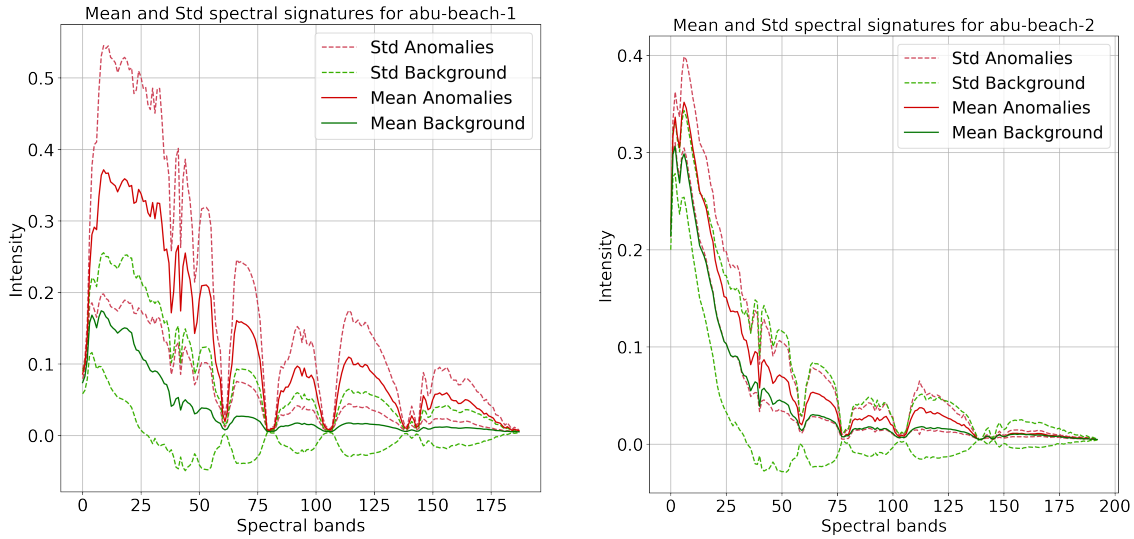


Figure 4.5: Spectral signatures for the 1st and 2nd beach scene.

The plots show that the 1st scene has a higher range of values and a low degree of overlap between the mean and standard deviation. Similar to the airport scenes, the first 50 bands contain the highest mean and standard deviation values. As discussed previously, the 2nd beach scene contains 58 anomaly objects, and by looking a the spectral signature, it can be seen that there is a high degree of overlap in almost all bands. This overlap indicates that the anomalies will be challenging to separate from the background pixels since the spectral signature of the pixels is difficult to distinguish. For the 1st scene, there is a low degree of overlap in all bands, meaning that the mean of the background and anomalies are not within the standard deviation of each other.

The spectral signatures for the 1st and 4th urban scenes is shown in Figure 4.6.

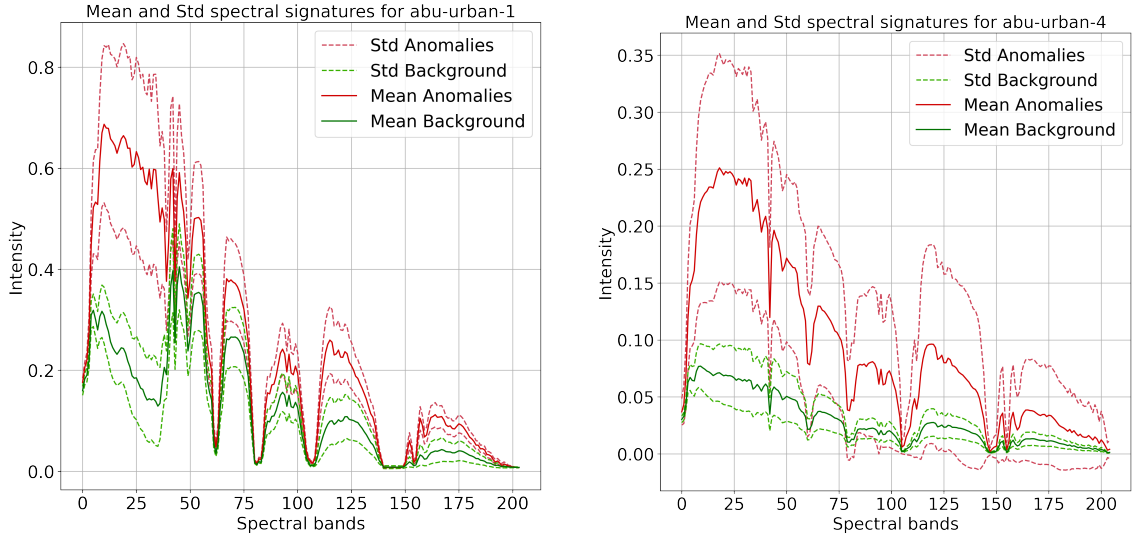


Figure 4.6: Spectral signatures for the 1st and 4th urban scene..

The plots for the urban scenes show that 1st scene has a much higher range of values than the 4th scene. The 1st scene has a low degree of overlap in the first 40 spectral bands and after about the 75th spectral bands. Compared to the previous plots of the airport and beach scenes, the urban scenes have some of the lowest degrees of overlap.

4.4 Implementation of Pre-processing Methods

PCA, and KPCA-based pre-processing are implemented using the code from the official implementation of the SSIIFD model [61]. KPCA is implemented using the RBF and Laplace kernel function with the parameters given in Table 4.4, where the number of PCs will be determined based on the experimental testing

Table 4.4: Parameter configuration for the RBF and Laplace kernel functions.

Parameter	Definition	Value
γ	Variance of RBF kernel.	0.5
σ	Inverse of variance for Laplace kernel.	2
ξ	Number of principle components.	100

The RFF-KPCA-based pre-processing is implemented in Python using the mathematical description given in Section 2.4.3, with the Gaussian and Laplacian distribution given in Table 2.1. All three pre-processing methods will be implemented using 100 PCs, as this was the value that gave the best detection performance.

4.5 Pre-processing Analysis

As presented in the implementation section, different pre-processing methods are applied to the ABU dataset to increase the separability between the anomalies and background pixels by enhancing the representation of anomalies. Enhanced separability can increase the detection performance and improve the computational efficiency of the HAD.

This section will present an analysis of the different pre-processing methods. The analysis is based on plots for three different ABU datasets, including the 1st airport scene, 2nd beach scene and 3rd urban scene. These plots provide insight into how the pre-processing methods influence the datasets and indicate their potential impact on the detection performance. For comparison, plots of these datasets before pre-processing are illustrated in Figure 4.7, in addition to the ground truth plots in Figure 4.8.

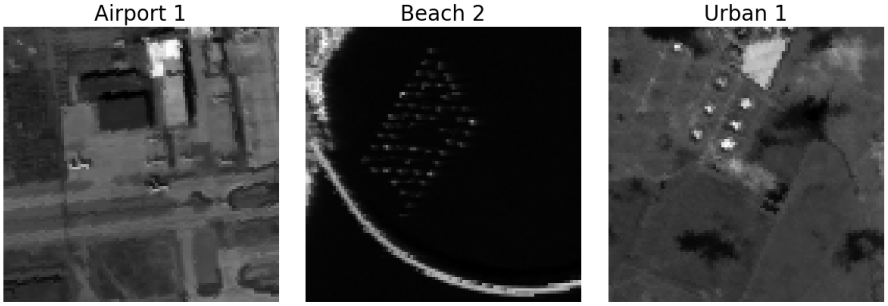


Figure 4.7: Plot of the 1st airport and urban scene, and 2nd beach scene from the ABU dataset.

Several observations can be made by comparing the plots in Figure 4.7 to the ground truth plots in Figure 4.8. The airport and urban scenes contain the least homogeneous background compared to the beach scene, which primarily consists of a dark background. For the airport scene, the background makes the anomalies blend in with the background. However, the anomalies remain relatively apparent in the urban scene despite the heterogeneous background. It is worth noting that a prominent object is present in the beach scene. Consequently, this object draws more attention than the anomalies, which are relatively difficult to perceive.

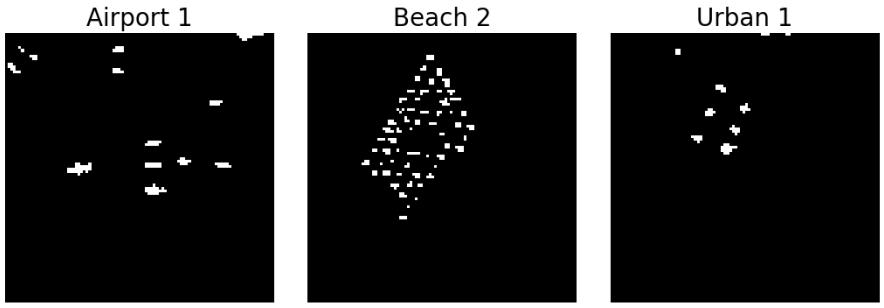


Figure 4.8: Ground truth plots of the 1st airport and urban scenes, and 2nd beach scene from the ABU dataset.

4.5.1 PCA

Plots of the different scenes after PCA-based pre-processing is presented in Figure 4.9.

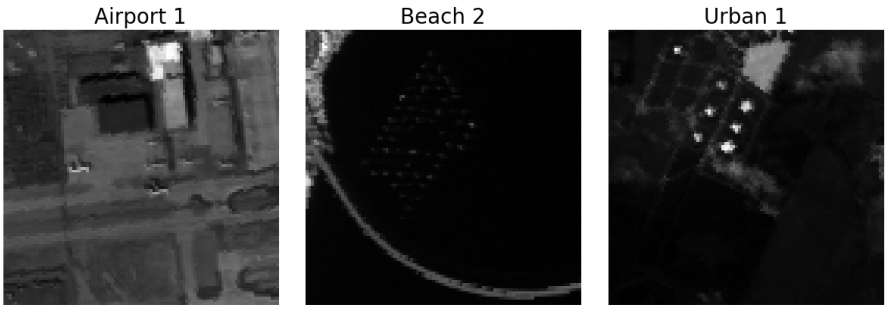


Figure 4.9: Plots of the 1st airport and urban scenes, and 2nd beach scene from the ABU dataset after PCA-based pre-processing.

Comparisons between the datasets with no pre-processing in Figure 4.7 and PCA-based pre-processing in the figure above indicate that PCA has negligible effects on the airport scene. The beach scene, however, illustrates that the PCA-based pre-processing has suppressed both the anomaly and background representation. Consequently, this results in a background with reduced variation, which can enhance the detection performance. However, this improvement is counterbalanced by the reduced representation of anomalies. The removal of background variation is also apparent in the urban scene. Additionally, the anomaly representation is not reduced, which indicates that the PCA will contribute to an increase in detection performance.

4.5.2 KPCA

Figure 4.10 visualizes the plots of the three scenes after KPCA-based pre-processing with the RBF kernel function.

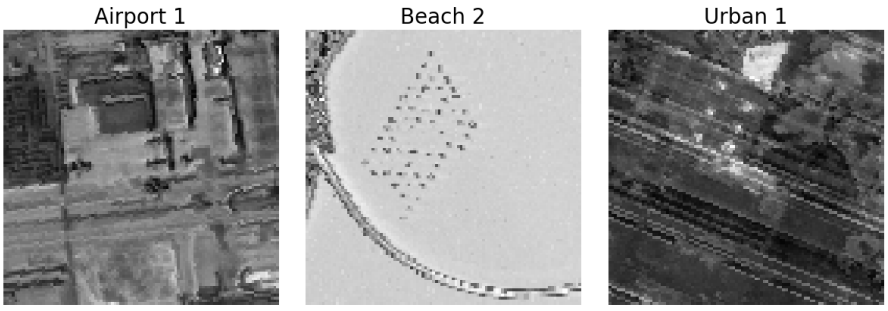


Figure 4.10: Plots of the 1st airport and urban scenes, and 2nd beach scene from the ABU dataset after KPCA-based pre-processing with the RBF kernel function has been applied.

The figure shows that the RBF kernel has had a noticeable impact on the plots compared to the previous plot where PCA was applied. For the airport and beach scene, there appears to be a more noticeable contrast between the anomalies and the background. However, in the case of the beach scene, the presence of the prominent object remains highly noticeable. This object could lead to a higher false alarm rate. Contrary to the airport and beach scenes, the contrast between the anomalies and background in the urban scene has been noticeably reduced by KPCA-based pre-processing. This reduction may indicate that the anomalies will be more difficult to separate from the background than in the previous case.

KPCA-base pre-processing with the Laplacian kernel function is visualized in Figure 4.11. For all three scenes, both the background and anomalies have become harder to differentiate due to a considerable amount of noise.

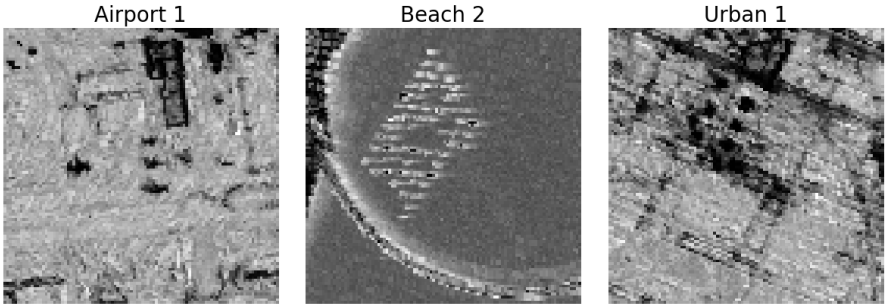


Figure 4.11: Plots of the 1st airport and urban scenes, and 2nd beach scene from the ABU dataset after KPCA-based pre-processing with the Laplacian kernel function has been applied.

Compared to the RBF-kernel function in Figure 4.10, both the beach and urban scenes appear to have a reversed colour scheme. However, the airport scene does not display this reversal. Notably, when employing the Laplacian kernel function on the airport scene, it is evident that the background contains significantly more noise.

4.5.3 RFF-KPCA

Figure 4.12 presents the of the Random Fourier Features KPCA-based pre-processing method with the Gaussian distribution function, and Figure 4.13 shows the results when the Laplacian distribution is applied.

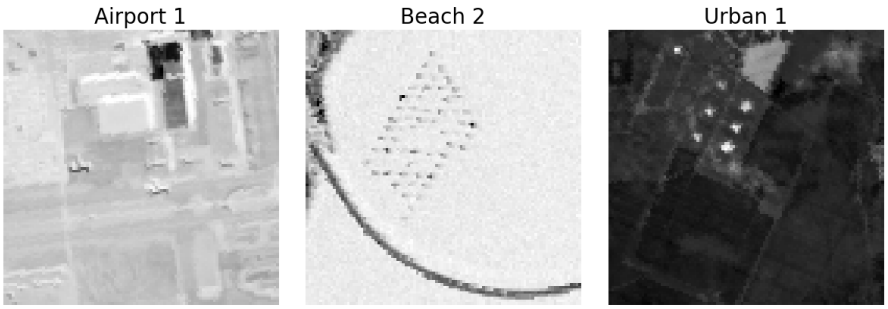


Figure 4.12: Plots of the 1st airport and urban scenes, and 2nd beach scene from the ABU dataset after RFF-KPCA-based pre-processing with the Gaussian distribution has been applied.

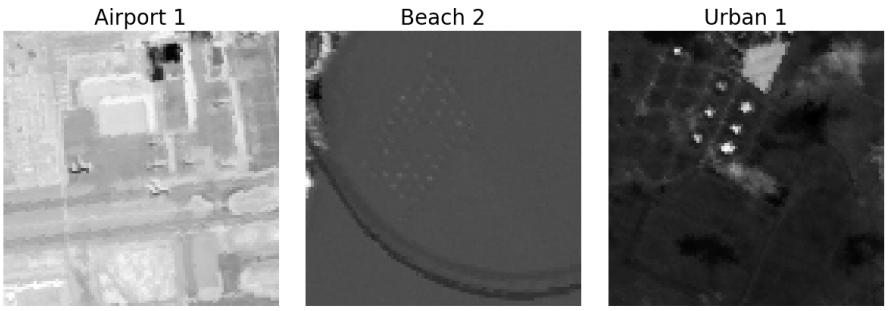


Figure 4.13: Plots of the 1st airport and urban scenes, and 2nd beach scene from the ABU dataset after RFF-KPCA-based pre-processing with the Laplacian distribution has been applied.

The results obtained from the RFF-KPCA-based pre-processing methods indicate that the Laplace and Gaussian distributions yield similar outcomes for the airport and urban scenes. However, in the case of the beach scene, the two distribution functions have had contrasting effects on the image, leading to distinct outcomes. Notably, it is observed that anomalies are almost entirely suppressed when utilizing the Laplace distribution on the beach scene.

The RFF-KPCA-based pre-processing method approximates the KPCA-based pre-processing. Therefore, it is expected that the RFF-KPCA with Gaussian distribution functions and KPCA plots with the RBF kernel function will exhibit a significant degree of similarity. Similarly, the RFF-KPCA with Laplacian distribution functions should demonstrate substantial similarity to KPCA with the Laplace kernel function. However, contrary to this expectation, the observed results indicate that the plots derived from these methods are noticeably different. This observation indicates that this approximated method is not sufficient.

4.5.4 Computational time

Table 4.5 gives the computational time for the different pre-processing methods. Since both the 1st and 4th beach scenes contain a significant number pixels, an external server was needed for computing the KPCA. Hence these are not part of the average. Everything else was run on the test setup in Table 4.1.

The results demonstrate that KPCA-based pre-processing has the highest time cost, while PCA and RFF-KPCA-based methods have negligible time costs.

Table 4.5: Time cost for the different pre-processing methods.

	PCA	KPCA RBF	KPCA Laplace	RFF Gaussian	RFF Laplace
Time cost	0.04s	94.29s	253.73s	0.11s	0.10s

4.6 Implementation of State-of-the-art Models

The RX, KIFD, SSIIFD and AUTO-AD models are implemented for comparisons to the proposed implementations. All models are implemented according to the description given in the background section. This section will present the parameter configuration for each of the models. The RX model does not require parameter setting. The RX, KIFD and SSIIFD models are implemented using code from the official implementation of the SSIIFD model in Matlab [61]. The AUTO-AD model is implemented using code from the official implementation of AUTO-AD in Python [62].

4.6.1 KIFD

The Kernel Isolation Forest Detection from Figure 2.12 was implemented according to [20]. Table 4.6 presents the parameter configurations. These parameters were, according to the paper, set using cross-validation.

Table 4.6: Parameter configuration for implementation of the KIFD model [20].

Parameter	Definition	Value
ξ	Number of PCs	300
γ	Variance in RBF	0.5
α	Threshold for size of anomaly.	$N/120$
q	Number of trees used to construct iForest	1000
M	Sub-sample size for iForest	$0.03 \cdot N$
H_{max}	Maximum length of iTree	$log_2 M$

4.6.2 SSIIFD

The parameter configurations provided by [4] for SSIIFD are presented in Table 4.7. The paper uses $U = 5$ and $V = 8$, denoting the number of scales and orientations for the Gabor filter bank, respectively. These parameters resulted in $D_{Gab} = 5 \times 8 = 40$ number of Gabor filters. All other parameters, except for ω, are determined using cross-validation. The balancing term, ω, was determined based on inspiration from [63]. The number of sub-regions, η, was set according to the type of hyper-spectral image, as can be seen in the table above. The paper concluded that using a high number of sub-regions η tends to perform poorly, leading to over-segmentation of regions, making it difficult to utilize all the samples belonging to homogeneous areas. On the other hand, a low number of sub-regions leads to overlapping homogeneous areas, making it impossible to utilize local information fully. In this thesis η was set to 3.

Table 4.7: Parameter configuration for implementation of the SSIIFD [4].

Parameter	Definition	Value
D_{Gab}	Number of Gabor filters.	40
ω	Balancing term for the decision fusion.	0.618
q	Number of trees used to construct iForest.	32
k	Number of bands used in the construction of IIFD.	B/3
M	Sub-sample size of the IIFD.	$0.025 \cdot N$
H_{max}	Maximum length of iTree.	$log_2 M$
η	Amount of sub-regions used in ERS.	3

4.6.3 AUTO-AD

The AUTO-AD method, presented in Figure 2.19, was implemented using a parameter configuration set according to [5]. However, the paper does not specify how they determined the parameters. The parameter configuration is presented in Table 4.8. The Python code for the official implementation of the AUTO-AD-model uses the PyTorch library [64] to implement the AE.

Table 4.8: Parameter configuration for implementation of the AUTO-AD model [5].

Parameter	Definition	Value
σ	Average loss value for training to stop.	$1.5 \cdot 10^{-5}$
λ	Learning rate for the ADAM.	0.01
β_1	Decay rate for ADAM .	0.9
β_2	Decay rate for ADAM .	0.99
$\mathbf{X}_0$	Uniform noise for input HSI.	[0,0.1]
H_{max}	Maximum number of epochs.	1000

In the implementation of the AUTO-AD model, min-max scaling was applied as described in (2.6). Although this step was not included in the official AUTO-AD code, its inclusion significantly enhanced the detection performance for both the airport and urban datasets.

4.7 Parameter Configuration of LW-AUTO-AD$^+$

The LW-AUTO-AD and LW-AUTO-AD$^+$ models were implemented using the PyTorch library in Python [64]. Several parameter configurations of the lightweight AE architecture in Figure 3.3 have been tested. For simplification, the tests were conducted using the LW-AUTO-AD$^+$ model with KPCA-based pre-processing using the RBF kernel function.

Tests were conducted to find the optimal dimensions for the encoder and decoder. These dimensions were with and without spectral dimensionality reduction, meaning that tests were performed with the same number of dimensions for each layer, $E_1 = E_2 = E_3$, and with decreasing number of dimensions for each subsequent layer, $E_1 > E_2 > E_3$. This analysis aimed to assess the impact of spectral dimensionality reduction on the detection performance. The results are shown in Table 4.9.

Table 4.9: Performance results for the different configuration lightweight AE architecture.

Dimensions	Airport Avg.	Beach Avg.	Urban Avg.	Epochs	Time
$[75, 50, 25]$	0.9715	0.9693	0.9875	558	87s
$[25, 25, 25]$	0.9691	0.9661	0.9856	665	87s
$[50, 50, 50]$	0.9712	0.9703	0.9868	569	107s
$[100, 100, 100]$	0.9739	0.9686	0.9869	515	165s

The results above indicate that the configurations with the lowest time cost are the first two configurations, with the first configuration ($[75, 50, 25]$) also achieving the highest average AUC scores. It is important to note that even though the configuration $[25, 25, 25]$ has fewer spectral bands than $[75, 50, 25]$, it does not demonstrate improved AUC scores, and the time cost remains unchanged. This observation can be attributed to the longer convergence time during training, as evidenced by the number of epochs required. Additionally, while the configuration $[100, 100, 100]$ achieves comparable detection performance, it comes with the highest time cost. Therefore, the results suggest that applying spectral dimensionality reduction improves detection performance while reducing the time cost.

Blocks 2,4,6,7,8, in addition to the first convolutional layer in block 9, contain kernel filters of size 3×3. This configuration is visualized in Figures 3.4 and 3.5. To evaluate the impact of the kernel filter size, tests were conducted with a kernel filter size of 5×5 and compared to the results when using a 3×3 kernel. Results, including the average AUC scores, number of epochs and computational time when applying the 5×5 kernel filter, are presented in Table 4.10. The results when applying the 3×3 kernel filter are also presented for comparison.

Table 4.10: Table presents the results for different configuration of the proposed AE.

Dimensions	Kernel Size	Airport Avg.	Beach Avg.	Urban Avg.	Epochs	Time
$[75, 50, 25]$	3×3	0.9715	0.9693	0.9875	558	87s
$[75, 50, 25]$	5×5	0.9697	0.9695	0.9855	550	123

The result shows that the AUC scores are similar for both kernel configurations, but computational time is significantly increased when using larger kernel sizes. Therefore, the overall performance is better when using the proposed kernel filter size of 3×3.

Based on these tests, the resulting channel configuration of the encoder and decoder in the LW-AUTO-AD$^+$ model is presented in Table 4.11. This configuration will also be used for the LW-AUTO-AD model, where no pre-processing is present.

Table 4.11: The parameter configuration with the highest detection performance for the LW-AUTO-AD$^+$ model.

Parameter	Definition	Value
$[E_1, E_2, E_3]$	Number of input channels in encoder.	$[75, 50, 25]$
$[D_1, D_2, D_3]$	Number of output channels in decoder.	$[75, 50, 25]$

4.8 Comparison of AUTO-AD and LW-AUTO-AD

The primary aim of the Lightweight AUTO-AD model was to decrease the computational cost associated with the AUTO-AD model while preserving or enhancing the detection performance. Therefore, this section will compare the AUTO-AD model and the LW-AUTO-AD model regarding computational cost and detection performance.

Table 4.12 presents the results using the average AUC score for the airport, urban and beach scenes, average time cost, number of learnable parameters in the AEs and number of epochs required to reach convergence.

Table 4.12: Results for the LW-AUTO-AD and AUTO-AD models.

Model	Airport Avg.	Beach Avg.	Urban Avg.	Epochs	Time	Nr. of Param.
AUTO-AD	0.8728	0.9747	0.8988	785	362s	3250125
LW-AUTO-AD	0.8573	0.9804	0.8759	935	181s	429755

For further analysis, Figure 4.14 illustrates the relation between the average AUC scores and average time cost over the airport, urban and beach scenes for the two models. The blue dots represent the LW-AUTO-AD, while the orange represents the AUTO-AD . To give a better visualization of the comparison, the x-axis range is limited to the upper 20% of its total range since none of the AUC scores falls below 80%.

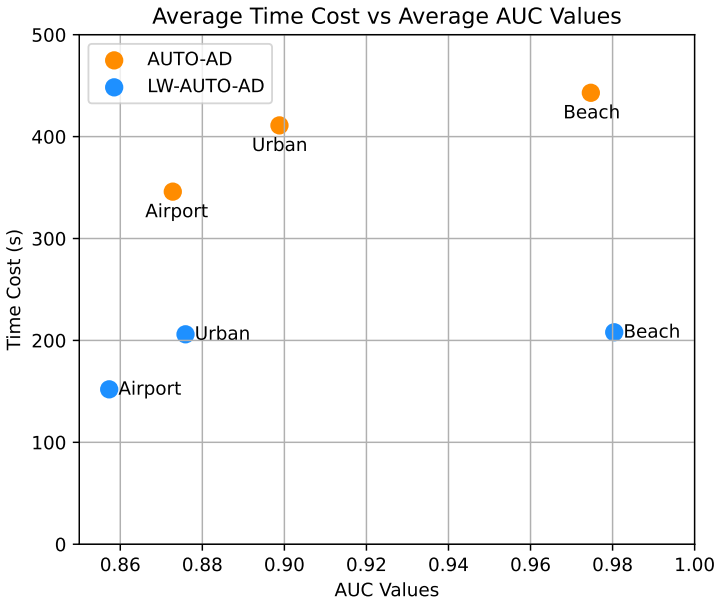


Figure 4.14: Plot of the time cost vs. AUC values for the LW-AUTO-AD and AUTO-AD models.

Regarding the AUC score, the AUTO-AD outperforms the LW-AUTO-AD model on both the airport and urban scenes, with a difference of about 2%. However, both models perform poorly in these scenes, with an AUC score of less than 0.90. On the other hand, both models have a high AUCvalue for the beach scenes, where the LW-AUTO-AD model performs slightly better than AUTO-AD, but not by a significant margin. Both models require a significant number of epochs to converge, which might indicate lower generalization performance and higher variance. However, it is worth noting that AUTO-AD requires almost 200 epochs less on average.

The average time cost analysis reveals that the LW-AUTO-AD model exhibits significantly lower computational cost across all scenes, requiring only half the time compared to the AUTO-AD model. This finding is further supported by Figure 4.14, which illustrates the average time cost for each scene. The plot demonstrates that the LW-AUTO-AD consistently outperforms AUTO-AD regarding computational efficiency, with the airport scenes exhibiting the lowest time cost across all scenes. This result can be explained by the significant reduction in the number of learnable parameters, which has been reduced by 87% from the AUTO-AD model to the LW-AUTO-AD model.

4.9 Performance Enhancement of LW-AUTO-AD$^+$

The LW-AUTO-AD$^+$ model was tested with several different pre-processing methods including, PCA-, KPCA-, RFF-KPCA- and MKL-based. This section will present the results of these configurations regarding the detection performance and computational cost.

Table 4.13 presents the average AUC scores and time cost for different pre-processing configurations, including PCA-based, KPCA-based with the RBF and Laplacian kernel function and RFF-KPCA-based pre-processing with the Gaussian and Laplace distribution. These pre-processing methods will be referred to as PCA, KPCA-RBF, KPCA-Laplace, RFF-Gaussian and RFF-Laplace, respectively. The average AUC scores are given for the airport, urban and beach scenes, whereas the time cost is given as an average across every scene. For comparison, the first column contains the AUC score for the LW-AUTO-AD model when no pre-processing has been applied.

Table 4.13: AUC score and average time cost for LW-AUTO-AD$^+$ with different pre-processing methods.

Model	Airport Avg.	Beach Avg.	Urban Avg.	Time
LW-AUTO-AD	0.8573	**0.9804**	0.8759	181s
PCA	0.8910	0.9482	0.9781	**44s**
KPCA - RBF	**0.9715**	0.9693	**0.9875**	181s
KPCA - Laplace	0.1010	0.5546	0.1832	411s
RFF - Gaussian	0.9027	0.9690	0.9128	129s
RFF - Laplace	0.9045	0.9730	0.8893	137s

By first analysing the average AUC scores, it is evident that the KPCA-based pre-processing with the RBF kernel has the overall best average score for the airport and urban scenes. However, for the beach scenes, the LW-AUTO-AD model with no pre-processing performed the best.

KPCA-Laplace pre-processing performs the worst in terms of AUC scores and time cost, but it appears that this poor performance is due to the inverted nature shown in Figure 4.11. A solution for this pre-processing configuration for both the airport and urban scenes would be to invert the AUC score, which would give an AUC score of almost 90%. However, this would not work for the beach scene.

PCA-based pre-processing has a lower detection performance than KPCA-RBF and a lower detection performance than both RFF-KPCA-based methods on the airport and beach scenes. However, PCA has the second highest AUC score on the urban scenes, outperforming both RFF-KPCA- based methods and the LW-AUTO-AD model. Moreover, PCA has the lowest time cost overall, taking only a third of the time required by RFF-KPCA-based methods and half as much as the KPCA-RBF method. Considering the pre-processing methods' time cost, the time difference between PCA and KPCA-based methods increases by more than 100 seconds. This observation further highlights the trade-off between time cost and detection performance, indicating that choosing the optimal pre-processing method requires balancing these factors.

This trade-off is illustrated in Figure 4.15 for all proposed pre-processing methods, except for KPCA-Laplace, because of the low performance. The time cost and AUC scores are given as an average over all the ABU datasets. To enhance the readability of the plot, the AUC values have been adjusted to start at 90%, as none of the models exhibits detection performance below this threshold.

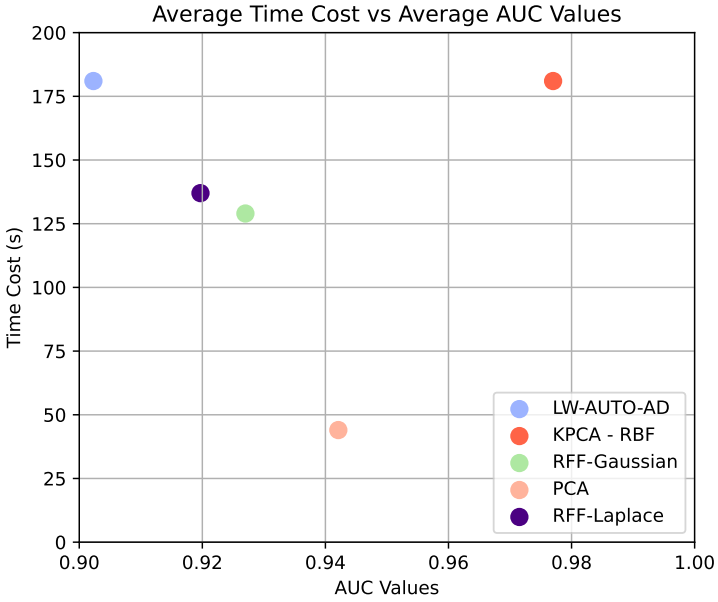


Figure 4.15: Plot of the time cost vs AUC values for the LW-AUTO-AD$^+$ model with different pre-processing methods and the LW-AUTO-AD model.

The plot further emphasises how LW-AUTO-AD demonstrates the worst performance, with the lowest detection performance, and highest time cost. Further, the plot highlights the trade-off between PCA-based and KPCA-RBF-based pre-processing. PCA achieves lower time cost but sacrifices detection performance by over 3%.

Despite the RFF-KPCA-based method being implemented to approximate the KPCA-based method, it is apparent that this is not the case for the airport or urban scenes. The average AUC scores in both scenes are much lower than those of the KPCA-based method, suggesting that the approximation could be more effective. This result is consistent with the pre-processing analysis conducted in Section 4.5, which demonstrated that the RFF-KPCA and KPCA-based methods had different effects on the datasets. However, the AUC scores for the beach scenes are almost identical for the Gaussian distribution and even slightly higher for the Laplacian distribution than KPCA-RBF.

Further, the AUC score for every ABU dataset is presented in Table 4.14 for the LW-AUTO-AD$^+$ model with all the implemented pre-processing methods: PCA, KPCA-RBF, KPCA-Laplace, RFF-Gaussian and RFF-Laplace. Additionally, the average time cost over all scenes is given. For comparison, the first row contains the results for the LW-AUTO-AD, where no pre-processing has been applied.

Table 4.14: Results for the LW-AUTO-AD model and LW-AUTO-AD$^+$ model with different pre-processing methods.

ABU Scene	**LW-AUTO-AD**	**PCA**	**KPCA RBF**	**KPCA Laplace**	**RFF Gaussian**	**RFF Laplace**
Airport 1	0.8048	0.8453	0.9474	0.1130	0.8959	0.9009
Airport 2	0.8324	0.8702	0.9811	0.0277	0.8844	0.9001
Airport 3	0.8740	0.9146	0.9622	0.2312	0.8707	0.8642
Airport 4	0.9180	0.9339	0.9954	0.0321	0.9597	0.9526
Airport Avg.	0.8573	0.8910	**0.9715**	0.1010	0.9027	0.9045
Beach 1	0.9868	0.9537	0.9894	0.5010	0.9843	0.9893
Beach 2	0.9617	0.9164	0.9124	0.9312	0.9604	0.9687
Beach 3	0.9916	0.9999	0.9995	0.1014	0.9999	0.9995
Beach 4	0.9814	0.9229	0.9761	0.6847	0.9313	0.9345
Beach Avg.	**0.9804**	0.9482	0.9693	0.5546	0.9690	0.9730
Urban 1	0.8436	0.9916	0.9907	0.0284	0.8971	0.8757
Urban 2	0.8835	0.9788	0.9894	0.1253	0.9007	0.8999
Urban 3	0.8977	0.9658	0.9845	0.1265	0.9323	0.9075
Urban 4	0.9485	0.9839	0.9931	0.6060	0.9533	0.9272
Urban 5	0.8061	0.9705	0.9797	0.0300	0.8807	0.8361
Urban Avg.	0.8759	0.9781	**0.9875**	0.1832	0.9128	0.8893
Time	181s	44s	181s	157s	129s	137s

The analysis of AUC scores for the different pre-processing methods shows that the airport scenes have the highest range between the highest and lowest achieved AUC scores. This observation is particularly evident in the 1st and 2nd scenes, with differences of around 14%. By comparing the AUC score with the dataset analysis, it is evident that the observations made for the 1st and 4th scenes correspond. The spectral signatures of the 1st scene showed a significantly higher degree of overlap when compared to the 4th scene. This suggests that the anomalies are easier to separate in the 4th scene, which is also evident in the AUC score.

The ROC-plot of the 1st airport scene for the LW-AUTO-AD$^+$ model with the different pre-processing methods, except KPCA-Laplace, are presented in Figure 4.16. Additionally, the LW-AUTO-AD is also presented in the plot. The plot highlights that KPCA-RBF outperforms the other pre-processing methods by a significant margin, particularly when compared to the LW-AUTO-AD and the PCA-based pre-processing.

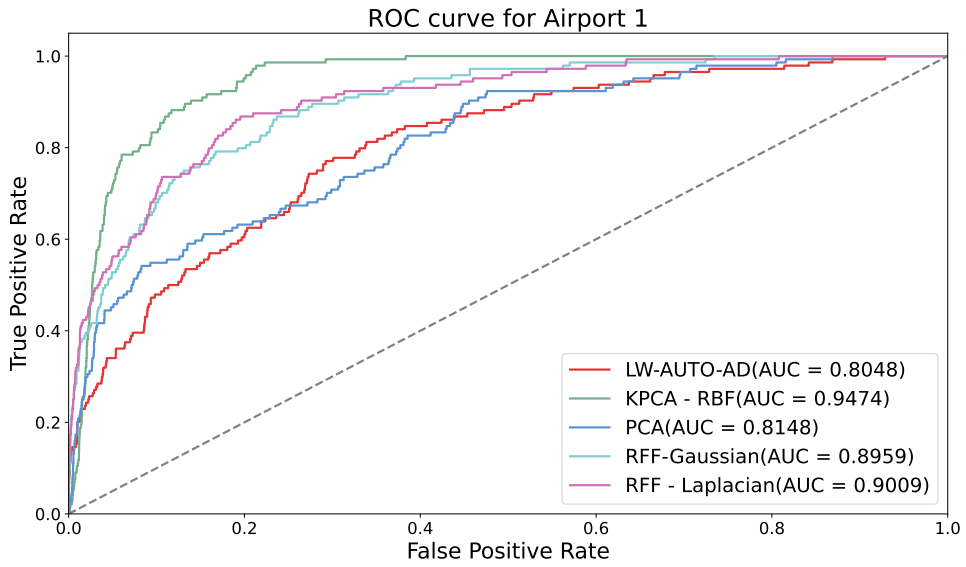


Figure 4.16: ROC plots for the 1st airport scene for the LW-AUTO-AD model and LW-AUTO-AD$^+$ model with the different pre-processing models.

Furthermore, Figure 4.17 presents the detection map plots for the 2nd airport scene, illustrating the difference between the LW-AUTO-AD model with no pre-processing and the LW-AUTO-AD$^+$ model with KPCA-based pre-processing with the RBF kernel function and Laplace kernel function. The plots demonstrate that without pre-processing, most anomalous pixels are misclassified as background pixels. This observation highlights the importance of KPCA in enhancing the separability between the background and anomalies. However, this is only the case for the RBF kernel function. The detection map shows that the Laplacian kernel function has a somewhat inverted plot from the RBF kernel function, which explains the low AUC score.

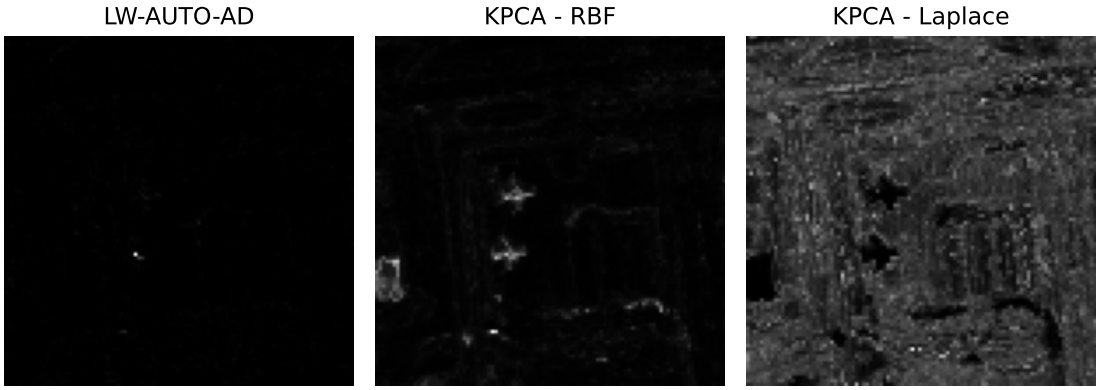


Figure 4.17: Detection map plot for the 2nd airport scene.

For the beach scenes, the 1st and 3rd scenes have similar AUC scores across all models, except for KPCA-Laplace. Conversely, the 2nd and 4th scenes exhibit a higher range of AUC scores. For the 2nd beach scene, the LW-AUTO-AD and both KPCA-RFF-based pre-processing methods exhibit similar AUC scores, while PCA and both KPCA-based methods exhibit lower AUC scores. It is worth noting that the KPCA-Laplace outperforms KPCA-RBF in this scene.

The lower AUC score observed for the 2nd beach scene is consistent with the dataset analysis, which identified a considerable number of anomaly objects with spectral signatures that closely resembled those of the background pixels. On the contrary, the dataset analysis revealed that the 1st beach scene consisted of only one anomaly, with considerably less overlap in the spectral

signatures. This observation suggested that the dataset would likely provide higher detection performance, which is verified by the corresponding AUC scores. This observation was also evident in the 3rd beach scene, which also contained one anomaly.

Figure 4.18 displays the ROC curves for the 2nd beach scenes with the same pre-processing methods as in the previous ROC plot. The curves exhibit significantly higher detection performance than the ROC plots for the airport scenes and highlight the poor performance of KPCA-RBF and PCA.

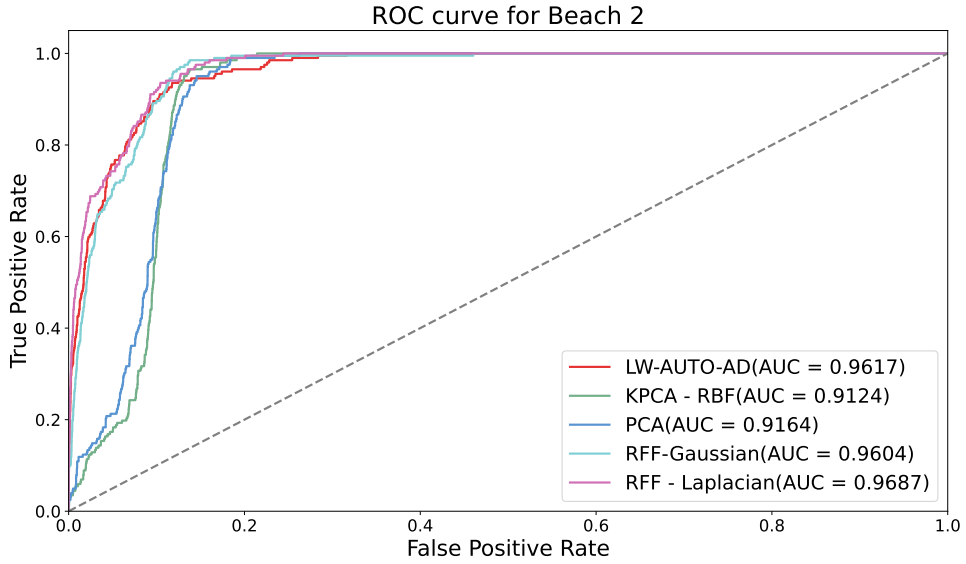


Figure 4.18: ROC plots for the 2nd beach scene for the LW-AUTO-AD$^+$ model with the different pre-processing models

The detection map plot of the 2nd beach scene, for the LW-AUTO-AD model with no pre-processing, and the LW-AUTO-AD$^+$ model with KPCA-based pre-processing with the RBF and Laplacian kernel is presented in Figure 4.19. Similar to the previously observed detection map, the model without pre-processing misclassifies anomalous pixels as background pixels. On the other hand, the KPCA-RBF case is able to classify more of the anomalies correctly. However, this has also caused the model to obtain a higher false alarm rate. Therefore, it is evident that there is a trade-off between detection performance and false alarm rate, where the highest AUC score is obtained without pre-processing since it does not misclassify background pixels for anomalies. This observation is also apparent when applying the Laplacian kernel. However, the AUC score reveals that it has been more successful in distinguishing anomalies from the background pixels than the RBF kernel.

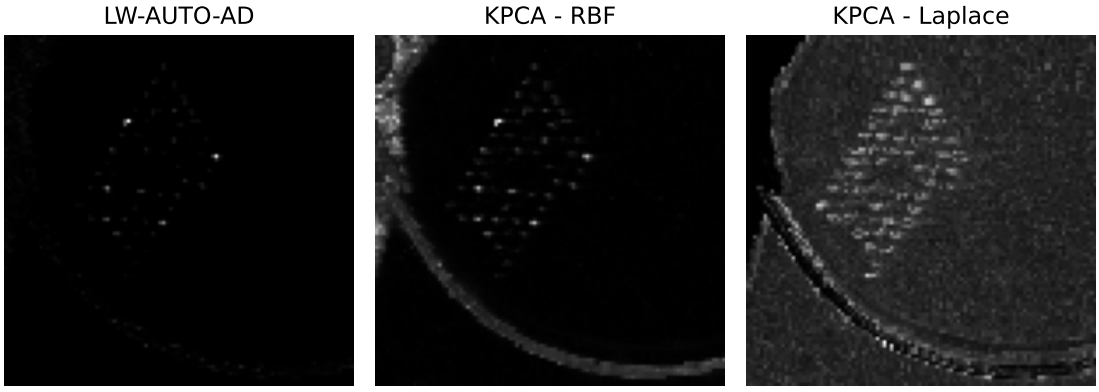


Figure 4.19: Detection map plot for 2nd beach scene.

For the urban scenes, the range between the highest and lowest achieved AUC score is most apparent for the 1st and 5th scenes, with the LW-AUTO-AD model and the LW-AUTO-AD$^+$ model with RFF-Laplace, and RFF-Gaussian performing the worst. The 4th urban scene shows the least range of differences, with no AUC score falling below 90%. This result is also reflected by the dataset analysis, which indicated that the 4th urban scene exhibited a notably low overlap between the spectral signatures of the anomalies and background pixels. Figure 4.20 presents the ROC-curves for the 4th urban scene with the same pre-processing methods as in the previous ROC plots.

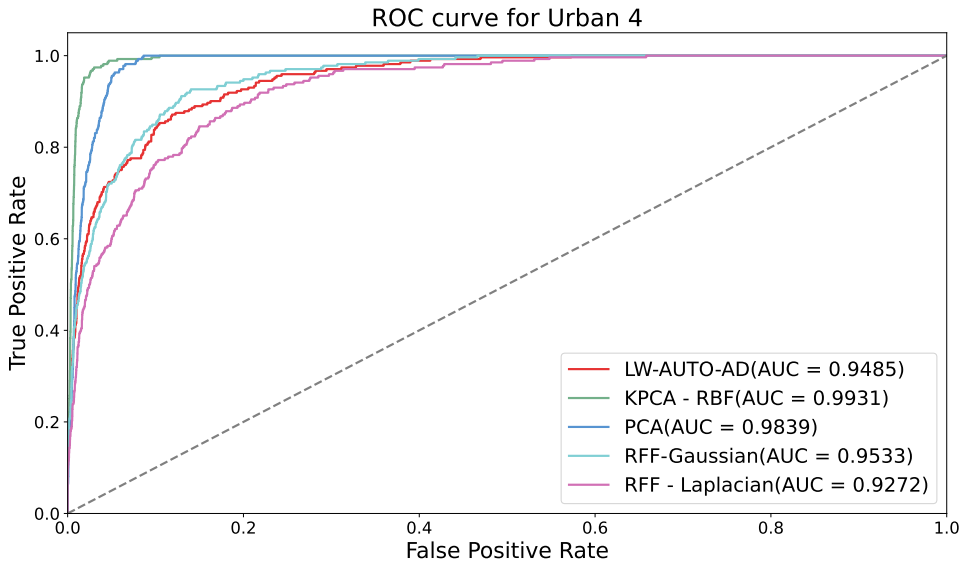


Figure 4.20: ROC plot for urban scene 4.

The results for the MKL-based pre-processing, following the structure in Figure 3.7, are presented in Table 4.15. Results are presented when combining the KPCA-based pre-processing with the RBF and Laplacian kernel function and the combination of RFF-KPCA-based pre-processing with the Gaussian and Laplacian distribution. For simplification, the two methods will be denoted by MKPCA and RFF-MKPCA, respectively. The AUC score is given for every ABU dataset, in addition to the average AUC score for each scene and the average time cost over all datasets. The fusion weight v_1, found using a grid search, is given for each method together with the variance.

Table 4.15: Results for the LW-AUTO-AD$^+$ model with MKPCA and RFF-MKPCA based pre-processing.

ABU Scene	MKPCA	RFF MKPCA
Airport 1	0.9474	0.9196
Airport 2	0.9811	0.9135
Airport 3	0.9622	0.8928
Airport 4	0.9954	0.9743
Airport Avg.	0.9715	0.9250
Beach 1	0.9894	0.9893
Beach 2	0.9823	0.9687
Beach 3	0.9996	0.9999
Beach 4	0.9763	0.9347
Beach Avg.	0.9869	0.9732
Urban 1	0.9907	0.9154
Urban 2	0.9894	0.9155
Urban 3	0.9845	0.9440
Urban 4	0.9932	0.9570
Urban 5	0.9797	0.9008
Urban Avg.	0.9875	0.9265
Fusion Weight	0.94 ± 0.02	0.48 ± 0.07
Time Cost	642s	266s

An illustration of the trade-off between the AUC score and time cost between the MKPCA and RFF-MKPCA together with the results from Table 4.14, is illustrated in Figure 4.21. The time cost and AUC scores are given as an average over all the ABU datasets.

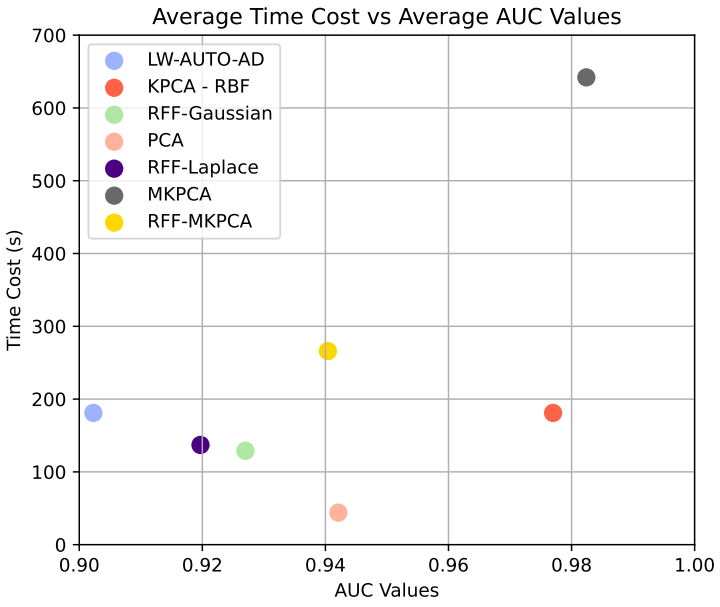


Figure 4.21: Plot of the time cost vs AUC values for the LW-AUTO-AD model and the LW-AUTO-AD$^+$ model with different pre-processing methods.

By comparing the results in Table 4.15 and Table 4.14, the AUC scores show that the MKPCA-based method performs better overall than the other pre-processing methods in terms of AUC score, with the most significant enhancement in the beach scenes. This result is also evident in the trade-off plot in Figure 4.21. It is worth noting that the average AUC scores for the airport and urban

scenes are the same as with only the KPCA with RBF kernel function. This observation indicates that the Laplacian kernel function provides no additional benefit for these scenes.

The fusion weight has a mean value of 0.94 with a variance of 0.02, indicating that the method has a good generalization property when applied to different datasets. However, although MKPCA-based pre-processing has proven to give the highest AUC score, it comes with a significantly increased time cost, as shown in Figure 4.21. The total average time consists of running both KPCA-based methods, 94 and 254 seconds, and the time cost of the AE at $87s$ and $157s$, which results in a time cost of more than $600s$. Hence the increased AUC score has caused a significant increase in the time cost.

The most notable improvement in the AUC score is observed for the 2nd beach scene, where the MKPCA-based method increases the score by 7%

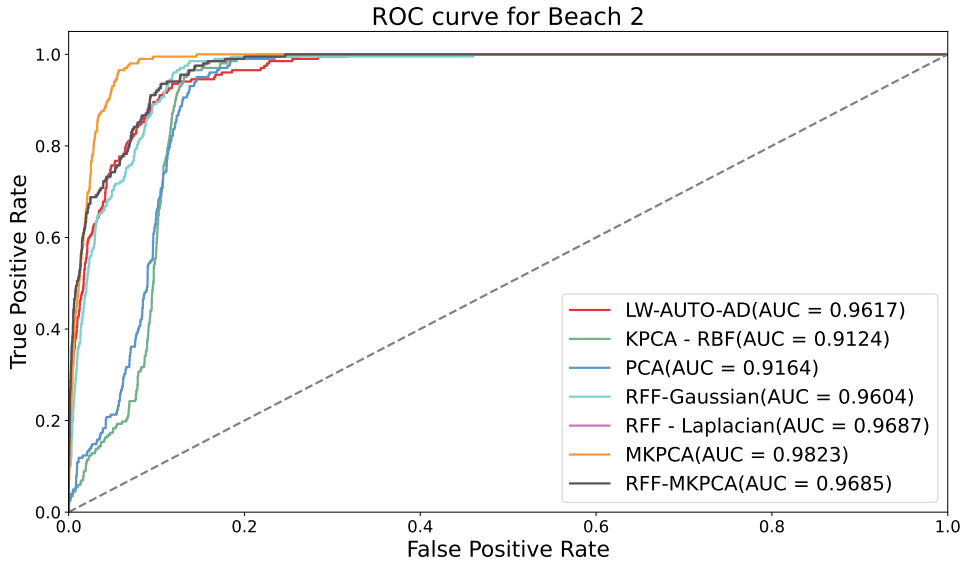


Figure 4.22: ROC plot of the 2nd beach scene.

To offer additional insights into the findings, Figure 4.23 presents the detection map plots for the KPCA-based method utilizing the RBF and Laplacian kernel functions, as well as the MKPCA-based method. The plot illustrates that the MKPCA-based method combines the effectively represented anomalies from the KPCA-Laplace method with the homogeneous background from KPCA-RBF. As a result, the detection map contains anomalies that are sufficiently represented, which is also evident in the improved AUC score.

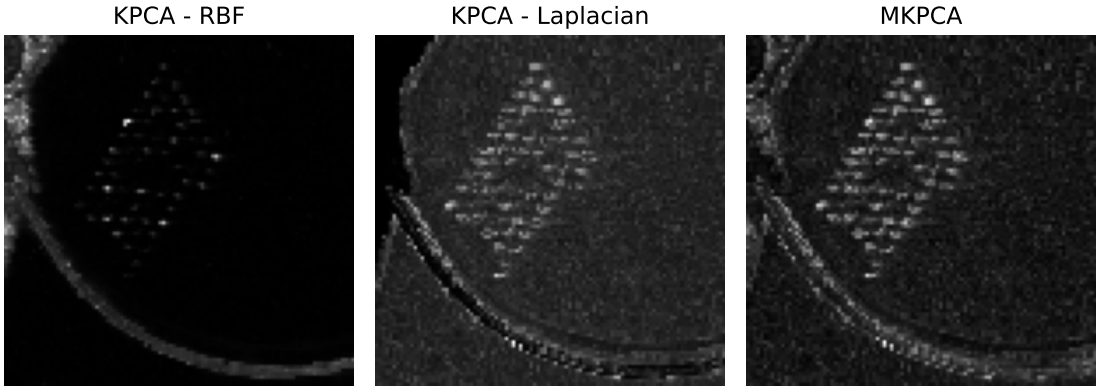


Figure 4.23: Detection map plot for 2nd beach scene.

The RFF-MKPCA-based pre-processing method shows an improvement in AUC score compared to the RFF-Laplace and RFF-Gaussian methods, but still not close to the performance achieved by the MKPCA-based method. The RFF-MKPCA also outperforms the PCA method on the airport and beach scenes but has a lower average AUC score on urban scenes, with a difference of over 5% compared to PCA. It is also worth noting that using two RFF-KPCA models for pre-processing increases the time cost.

The results of the different pre-processing methods can be summarized as follows: MKPCA-based pre-processing demonstrates the highest detection performance, indicating its effectiveness in separating anomalies from the background. However, MKPCA also results in the highest computational cost. On the other hand, PCA exhibits the lowest time cost, making it computationally efficient but with a lower detection performance than both MKPCA and KPCA-RBF. KPCA-RBF shows a detection performance comparable to MKPCA with a significantly lower time cost, making it a viable alternative. It achieves a higher AUC score than PCA, indicating better overall anomaly detection, but at the cost of higher computational time than PCA. Given these findings, the subsequent comparisons will focus on the LW-AUTO-AD$^+$ with the MKPCA-based, PCA-based and KPCA-RBF-based pre-processing methods.

4.9.1 Comparative Analysis of LW-AUTO-AD$^+$ and AUTO-AD$^+$

Table 4.16 provides the results of the LW-AUTO-AD model, as well as the LW-AUTO-AD$^+$ and AUTO-AD$^+$ models with KPCA-based pre-processing using the RBF kernel function. The results include the average AUC score for the airport, urban and beach scenes, average time cost across all scenes, and the number of epochs required to reach convergence.

Table 4.16: Results for the LW-AUTO-AD, LW-AUTO-AD$^+$ and AUTO-AD$^+$ models.

Model	**Airport Avg.**	**Beach Avg.**	**Urban Avg.**	**Epochs**	**Time**
LW-AUTO-AD	0.8573	0.9804	0.8759	935	181s
LW-AUTO-AD$^+$	0.9715	0.9693	0.9875	558	181s
AUTO-AD$^+$	0.9659	0.9680	0.9854	350	231s

As previously observed, KPCA-based pre-processing significantly impacts the AUC score and time cost for the LW-AUTO-AD model. Further, the LW-AUTO-AD$^+$ model outperforms AUTO-AD$^+$ with regards to both average time cost and detection performance. The reduced time cost is expected due to the lightweight AE architecture, significantly reducing the number of learnable parameters. Although the AUTO-AD model outperformed the LW-AUTO-AD model in terms of detection performance, the comparison between the LW-AUTO-AD$^+$ and AUTO-AD$^+$ models revealed a different outcome. This result highlights that the LW-AUTO-AD$^+$ model can achieve better detection performance while simultaneously reducing computational cost, offering a more efficient solution.

4.10 Performance Comparison to State-of-the-art

In this section, a comparison is presented between the proposed LW-AUTO-AD and LW-AUTO-AD$^+$ models and state-of-the-art models, namely the RX, KIFD and SSIIFD models. This comparison aims to evaluate the performance of the proposed models in relation to these existing approaches.

The LW-AUTO-AD$^+$ model is presented for two configurations, the first utilizes KPCA-based pre-processing with the RBF kernel function, and the second applies MKPCA -based pre-processing with the RBF and Laplacian kernel functions. Table 4.17 presents the results using the AUC scores and average time cost. The LW-AUTO-AD$^+$ models are denoted as KPCA and MKPCA .

Table 4.17: Results for the RX, KIFD, SSIIFD and AUTO-AD, in addition to the AUTO-AD model with the proposed AE with KPCA-RBF and MKPCA .

ABU Scene	**RX** [7]	**KIFD** [20]	**SSIIFD** [4]	**LW-AUTO-AD**	**KPCA**	**MKPCA**
Airport 1	0.8221	0.9192	0.8991	0.8048	0.9474	0.9474
Airport 2	0.8404	0.9755	0.9616	0.8324	0.9811	0.9811
Airport 3	0.9288	0.9553	0.9594	0.8740	0.9622	0.9622
Airport 4	0.9526	0.9914	0.9976	0.9180	0.9954	0.9954
Airport	0.8860	0.9604	0.9544	0.8573	0.9715	0.9715
Beach 1	0.9807	0.9849	0.9560	0.9868	0.9894	0.9894
Beach 2	0.9106	0.9038	0.9383	0.9617	0.9124	0.9823
Beach 3	0.9998	0.9891	0.99997	0.9916	0.9995	0.9996
Beach 4	0.9538	0.9738	0.9488	0.9814	0.9761	0.9763
Beach	0.9612	0.9629	0.9608	0.9804	0.9693	0.9869
Urban 1	0.9907	0.9865	0.9697	0.8436	0.9907	0.9907
Urban 2	0.9946	0.9863	0.9971	0.8835	0.9894	0.9894
Urban 3	0.9513	0.9550	0.9649	0.8977	0.9845	0.9845
Urban 4	0.9887	0.9822	0.9899	0.9485	0.9931	0.9932
Urban 5	0.9692	0.9772	0.9541	0.8061	0.9797	0.9797
Urban	0.9789	0.9775	0.9751	0.8759	0.9875	0.9875
Time	$< 1s$	98s	28s	181s	181s	642s

By first observing the average AUC scores, it is evident that the LW-AUTO-AD$^+$ model with MKPCA-based pre-processing outperforms all the presented state-of-the-art models. In addition, the LW-AUTO-AD$^+$ model with KPCA-based pre-processing outperforms the other state-of-the-art models on both the airport and urban scenes. The results and analysis presented earlier demonstrated that the LW-AUTO-AD$^+$ model outperforms the LW-AUTO-AD model. Based on the results in Table 4.17, the state-of-the-art models outperform the LW-AUTO-AD model in the airport and urban scenes. However, the LW-AUTO-AD model achieves the second-highest detection performance for the beach dataset among the evaluated models.

The detection map plots for the configuration of the highest detection performance, namely the MKPCA-based method, are illustrated in Figure 4.24 for all ABU datasets. The ground truth plots for these scens are given in Figures 4.1, 4.2 and 4.3.

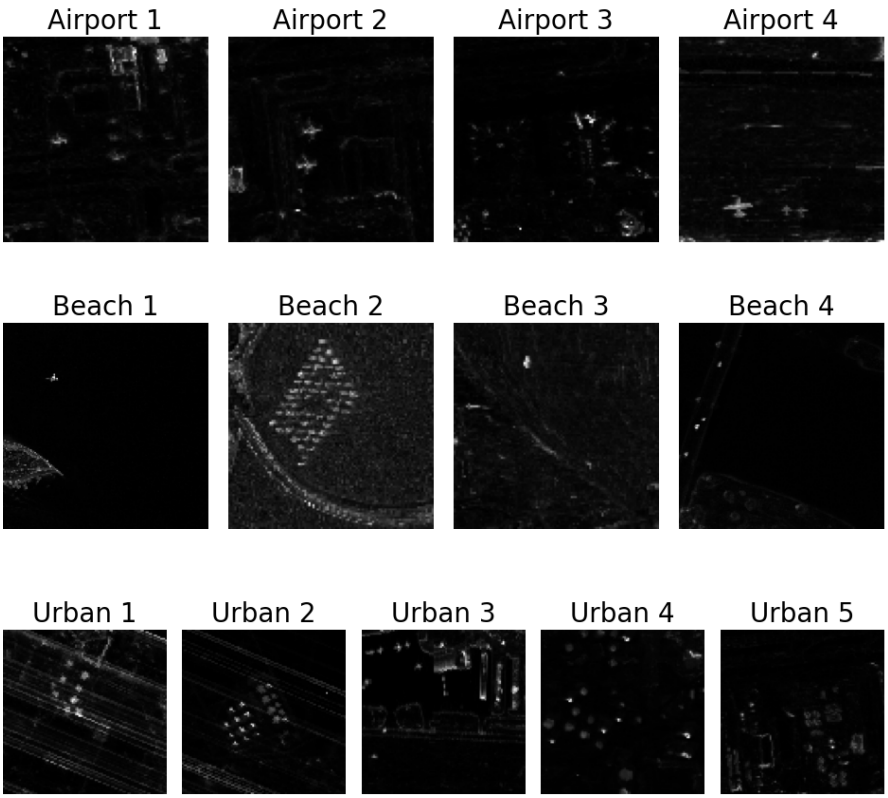


Figure 4.24: Detection map plots when using the MKPCA-based pre-processing method.

Further, the results show that the three deep learning-based HAD models have an average time cost significantly higher than the traditional-based model, particularly the RX model, which has a time cost below 1 second. The KIFD model has the highest time cost of the traditional-based models, which is due to the KPCA-based pre-processing algorithm. For the deep-learning-based models, the LW-AUTO-AD$^+$ model with KPCA has the lowest time cost, whereas the same model with MKPCA-based pre-processing has the highest with more than 600 seconds.

A notable observation from the results is that the average AUC score for the beach scenes is similar for all models. This is not the case for the airport or urban scenes, where the AUC score ranges around 10% from the lowest to the highest AUC score.

It is evident from the table that the KIFD and SSIIFD models have similar performance. However, the KIFD model obtains the highest overall AUC scores. This result indicates that the improvements that were supposed to be introduced by the SSIIFD model have had a small impact on the overall result. This outcome can, however, be a result of the parameter configuration of the SSIIFD model, which was standard for each dataset. In contrast, the SSIIFD paper sets the parameter based on the given dataset. Hence, the SSIIFD model may have outperformed the KIFD model if tuned to each dataset.

Chapter 5

Conclusion and Future Directions

The primary objective of this thesis was to develop a Hyperspectral Anomaly Detection model suitable for integration onto the Hyper-Spectral Small Satellite for Ocean Observations onboarding process. Three models were developed, each focused on enhancing the performance and computational efficiency of the Autonomous Hyperspectral Anomaly Detection Network (AUTO-AD) model. This section provides a summary of the most significant findings in the results. Moreover, it presents possible further investigations or future directions to pursue in subsequent research.

5.1 Summary of Results

The first contribution was the AUTO-AD$^+$ model, which focuses on improving the detection performance by implementing KPCA-based pre-processing. By comparing the AUC score of the AUTO-AD model with the AUTO-AD$^+$ model, it became apparent that the detection performance exhibited a notable enhancement of approximately 10% for both urban and airport scenes. These findings further support the assumption that incorporating KPCA-based pre-processing enhances the detection performance. Additionally, it is essential to highlight that the inclusion of KPCA-based pre-processing has also caused a reduction in computational costs by significantly decreasing the training time of the autoencoder.

Further, the Lightweight AUTO-AD model focused on reducing the computational cost of the autoencoder architecture of the AUTO-AD model. Results revealed that the LW-AUTO-AD model reduced the number of learnable parameters by 87% compared to the AUTO-AD model. This significant reduction in parameters led to a considerable time cost reduction of 50%. However, the AUC score of the LW-AUTO-AD model had a slightly lower AUC score of about 1% on average in comparison to the AUTO-AD model. Consequently, the LW-AUTO-AD model demonstrated lower detection performance, however, offering a significant reduction in time cost.

The most notable finding of this thesis was the LW-AUTO-AD$^+$ model, which integrated the LW-AUTO-AD model with a pre-processing step. LW-AUTO-AD$^+$ proved to be highly effective in optimizing the detection performance and time cost compared to the AUTO-AD model. The LW-AUTO-AD$^+$ model was configured with various pre-processing methods, among which the most notable ones included PCA-based, KPCA-based, and MKPCA-based pre-processing. Each pre-processing method offered distinct advantages regarding the trade-off between detection performance and time cost.

The Multi KPCA-based pre-processing method, which combined the RBF and Laplace kernel function, demonstrated the overall highest detection performance among the models. The average AUC scores obtained were 0.9715 for airport scenes, 0.9869 for beach scenes, and 0.9857 for urban scenes. These scores reflected a considerable improvement of more than 10% for airport and beach scenes and approximately 1% for urban scenes when compared to the AUTO-AD model. However, the time cost increased from 362 to 642 seconds.

In contrast, the PCA-based pre-processing method demonstrated the lowest time cost, requiring only 44 seconds. Although the detection performance increased for both the airport and urban scenes compared to the AUTO-AD model, it remained lower than that of the MKPCA-based pre-processing.

The KPCA-based pre-processing method with the RBF kernel function provided a balance

between the low time cost of PCA and the detection performance of MKPCA. This pre-processing configuration had a time cost of 181 seconds and achieved the same AUC score as MKPCA-based pre-processing for the airport and urban scenes. However, the AUC score for the beach scene was slightly lower, approximately 2% less, compared to MKPCA-based pre-processing.

It is also worth noting that the LW-AUTO-AD$^+$ model was tested with the RFF-KPCA- and RFF-MKPCA-based pre-processing methods, which aimed to approximate the data transformation performed by KPCA while reducing the time cost. However, the results demonstrated that the RFF-KPCA approximation was insufficient, giving lower detection performances than all previously mentioned pre-processing methods.

Further, the comparison between the state-of-the-art models revealed that the LW-AUTO-AD$^+$ achieved the highest detection performance with both the KPCA- and MKPCA-based pre-processing methods. However, the time cost results showed that the LW-AUTO-AD$^+$ came with a higher time cost than the traditional-based state-of-the-art models.

These results highlight the trade-off between time cost and detection performance. In applications like satellite-based systems such as HYPSO, evaluating the time cost becomes crucial to minimize unnecessary power usage. However, maintaining a reliable model is equally important. Therefore, finding an optimal balance between time cost and detection performance is crucial in such applications.

5.2 Future Work

While this thesis has provided insight into optimizing hyperspectral anomaly detectors based on detection performance and computational complexity, several aspects can be further investigated and explored. To summarize, future research should primarily concentrate on two main areas: reducing the computational complexity of the model and conducting tests on additional datasets. By addressing these issues, integrating this model into the HYPSO satellite is more likely to yield efficient power usage and improve overall performance. In the following section, an outline of potential directions for future work will be presented.

5.2.1 Pre-processing

Based on the results, it was concluded that pre-processing enhanced the detection performance significantly, in addition to reducing the number of epochs required for the model to converge. However, as has been mentioned several times, both KPCA and MKPCA require the computation of a kernel matrice with a high computational cost. Despite introducing the RFF-KPCA-based pre-processing method as an approximation to reduce the computational costs of the kernel matrix, the model proved insufficient. Consequently, additional research can be conducted to enhance the detection performance of this approximation without increasing the time cost. One potential method for exploration is investigating different distribution functions that can improve the accuracy of the approximation.

Further, this thesis employed MKL by running the pre-processing and anomaly detection blocks in parallel. This approach doubled the computational cost, as the kernel matrices and anomaly detection algorithms had to be executed twice. A potential direction for future work is to apply MKL using the configuration described in Equation (2.16), where the kernel matrix is computed using a linear combination of M kernel functions. This approach eliminates the need to compute multiple kernel matrices or run the anomaly detection algorithm twice, significantly reducing the computational cost. Furthermore, this thesis only tested the model with two kernel functions. For future research, exploring the model's performance using more than two kernel functions would be interesting.

5.2.2 Parameter Tuning

As stated in the introduction and demonstrated in the experimental testing, simplifications were applied to avoid extensive testing regarding the parameter configurations of the AE and pre-processing methods. Therefore, parameter configurations are areas for further investigation. A possible solution is to apply a hyperparameter tuning tool, which can simplify the process of finding optimal hyperparameters for increased performance. One such tool is the Ray Tune parameter

tuner by PyTorch, which applies the latest hyperparameter search algorithms in addition to other analysis libraries [65].

5.2.3 Non-linear Filters for CNNs

Another aspect worth investigating is the introduction of non-linear filters in CNN. Currently, the standard convolutional filter is applied at each convolutional layer in the AE. However, recent research has proposed to use non-linear filters instead of the standard convolutional filter [66, 67]. Gabor filters were introduced to CNNs in [66] to enhance the robustness of orientation and scale change. This model, denoted as the Gabor Convolutional Network (GCN), has shown promising results when applied to 2D images. Additionally, the Gabor filter has been shown to give good results when applied to HAD models such as the SSIIFD model [4]. Given the considerations above, exploring GCNs can be a promising avenue for future research.

5.2.4 Dataset

This thesis has solely focused on testing with the ABU dataset. However, future tests should be performed using other datasets, such as the HYPSO dataset. For this to be possible, the computational cost of the pre-processing methods, particularly the KPCA-based pre-processing, must be reduced since these datasets consist of a significantly higher number of pixels. Another factor is the lack of labeled datasets, which, although the AE-based network is an unsupervised learning method, would make it difficult to evaluate the model's detection performance.

References

[1] National Geographics. *Oceanography*. 2022. URL: https://education.nationalgeographic.org/resource/oceanography.

[2] Gamal ElMasry and Da-Wen Sun. 'Principles of Hyperspectral Imaging Technology'. In: *Hyperspectral Imaging for Food Quality Analysis and Control*. Ed. by Da-Wen Sun. San Diego: Academic Press, 2010, pp. 3–43.

[3] *HYPSO hyperspectral satellite data fusion with in-situ sensors - NTNU*. 2022. URL: https://www.ntnu.edu/smallsat/hypso-hyperspectral-satellite-data-fusion-with-in-situ-sensors.

[4] Xiangyu Song et al. 'Spectral–Spatial Anomaly Detection of Hyperspectral Data Based on Improved Isolation Forest'. In: *IEEE Trans. Geosci. Remote Sens.* 60 (2022). DOI: 10.1109/TGRS.2021.3104998.

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

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

[7] I.S. Reed and X. Yu. 'Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution'. In: *IEEE Trans. Acoust., Speech, Signal Process.* 38.10 (1990), pp. 1760–1770. DOI: 10.1109/29.60107.

[8] Wei Li and Qian Du. 'Collaborative Representation for Hyperspectral Anomaly Detection'. In: *IEEE Trans. Geosci. Remote Sens.* 53 (2015), pp. 1463–1474. DOI: 10.1109/TGRS.2014.2343955.

[9] Lu Li et al. 'Prior-Based Tensor Approximation for Anomaly Detection in Hyperspectral Imagery'. In: *IEEE Trans. Neural Net. Learn. Syst.* 33.3 (2022), pp. 1037–1050. DOI: 10.1109/TNNLS.2020.3038659.

[10] Xiyou Fu et al. 'Hyperspectral Anomaly Detection via Deep Plug-and-Play Denoising CNN Regularization'. In: *IEEE Trans. Geosci. Remote Sens.* 59 (2021), pp. 9553–9568. DOI: 10.1109/TGRS.2021.3049224.

[11] Wei Li, Guodong Wu and Qian Du. 'Transferred Deep Learning for Anomaly Detection in Hyperspectral Imagery'. In: *IEEE Geosci. Remote Sens. Lett.* 14.5 (2017), pp. 597–601. DOI: 10.1109/LGRS.2017.2657818.

[12] Jiaping Zhong et al. 'Characterization of Background-Anomaly Separability With Generative Adversarial Network for Hyperspectral Anomaly Detection'. In: *IEEE Trans. on Geosci. and Remote Sens.* 59.7 (2021), pp. 6017–6028. DOI: 10.1109/TGRS.2020.3013022.

[13] Pei Xiang et al. 'Hyperspectral Anomaly Detection With Guided Autoencoder'. In: *IEEE Trans. on Geosci. and Remote Sens.* 60 (2022), pp. 1–18. DOI: 10.1109/TGRS.2022.3207165.

[14] J. Zhang et al. 'Anomaly detection in hyperspectral image using 3D-convolutional variational autoencoder'. In: *Proc. IEEE Int. Geosci. Remote Sens. Symp.* 2021, pp. 2512–2515.

[15] Shizhen Chang, Bo Du and Liangpei Zhang. 'A Sparse Autoencoder Based Hyperspectral Anomaly Detection Algorihtm Using Residual of Reconstruction Error'. In: *Proc. EEE Int. Geosci. Remote Sens. Symp.* 2019, pp. 5488–5491.

[16] Katinka Müller. *Hyperspectral Anomaly Detection based on state-of-the-art Machine Learning Models*. Tech. rep. Norwegian University of Science and Technology, Dec. 2022.

[17] K.-R. Muller et al. 'An introduction to kernel-based learning algorithms'. In: *IEEE J. Trans. Neural Net.* 12.2 (2001), pp. 181–201. DOI: 10.1109/72.914517.

[18] Yen-Yu Lin, Tyng-Luh Liu and Chiou-Shann Fuh. 'Multiple Kernel Learning for Dimensionality Reduction'. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33 (2011), pp. 1147–1160. DOI: 10.1109/TPAMI.2010.183.

[19] Ali Rahimi and Benjamin Recht. 'Random Features for Large-Scale Kernel Machines'. In: *Proc. Adv. Neural Info. Process. Syst.* Vol. 20. 2007.

[20] Shutao Li et al. 'Hyperspectral Anomaly Detection With Kernel Isolation Forest'. In: *IEEE Trans. on Geosci. and Remote Sens.* 58.1 (2020), pp. 319–329. DOI: 10.1109/TGRS.2019.2936308.

[21] Yichu Xu et al. 'Hyperspectral Anomaly Detection Based on Machine Learning: An Overview'. In: *IEEE J. Sel. Topics Appl. Earth Observ. Remote Ses.* 15 (2022), pp. 3351–3364. DOI: 10.1109/JSTARS.2022.3167830.

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

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

[24] eoPortal. *HYPSO (Hyper-Spectral Small Satellite for Ocean Observation)*. 2022. URL: https://www.eoportal.org/satellite-missions/hypso#hsi-hyperspectral-imager.

[25] *EnMAP*. URL: https://www.enmap.org/ (visited on 5th Dec. 2022).

[26] *PRISMA (Hyperspectral)*. URL: https://www.eoportal.org/satellite-missions/prisma-hyperspectral#prisma-hyperspectral-precursor-and-application-mission (visited on 2nd Dec. 2022).

[27] *AVIRIS - Airborne Visible / Infrared Imaging Spectrometer*. URL: https://aviris.jpl.nasa.gov/ (visited on 3rd June 2023).

[28] Varun Chandola, Arindam Banerjee and Vipin Kumar. 'Anomaly detection: A survey'. In: *ACM Comput. Surv* 41.3 (2009), pp. 1–58. DOI: 10.1145/1541880.1541882.

[29] *Data sets*. Xudong Kang's Homepage. URL: http://xudongkang.weebly.com/data-sets.html (visited on 3rd Dec. 2022).

[30] Jiawei Han, Jian Pei and Micheline Kamber. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[31] 'On the Generalised Distance in Statistics'. In: 80 (1936), pp. 1–7. DOI: 10.1007/s13171-019-00164-5.

[32] Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou. 'Isolation-Based Anomaly Detection'. In: *ACM Trans. Knowl. Discov. Data* 6.1 (2012), 3:1–3:39. DOI: 10.1145/2133360.2133363.

[33] V. Klema and A. Laub. 'The singular value decomposition: Its computation and some applications'. In: *IEEE Trans. on Auto. Control* 25.2 (1980), pp. 164–176. DOI: 10.1109/TAC.1980.1102314.

[34] Saad Albawi, Tareq Abed Mohammed and Saad Al-Zawi. 'Understanding of a convolutional neural network'. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[35] Antonia Creswell et al. 'Generative Adversarial Networks: An Overview'. In: *IEEE Signal Process. Mag.* 35.1 (2018), pp. 53–65. DOI: 10.1109/MSP.2017.2765202.

[36] Heesung Kwon and N.M. Nasrabadi. 'Kernel RX-algorithm: a nonlinear anomaly detector for hyperspectral imagery'. In: *IEEE Trans. Geosci. Remote Sensing* 43.2 (2005), pp. 388–397. DOI: 10.1109/TGRS.2004.841487.

[37] Hamidullah Binol. 'Ensemble Learning Based Multiple Kernel Principal Component Analysis for Dimensionality Reduction and Classification of Hyperspectral Imagery'. In: *Mathematical Problems in Engineering* 2018 (2018), pp. 1–14. DOI: 10.1155/2018/9632569.

[38] Svante Wold, Kim Esbensen and Paul Geladi. 'Principal component analysis'. In: *Chemom. Intell. Lab. Syst.* 2.1 (1987), pp. 37–52. DOI: 10.1016/0169-7439(87)80084-9.

[39] Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller. 'Nonlinear Component Analysis as a Kernel Eigenvalue Problem'. In: *J. Neural Comp.* 10 (1998), pp. 1299–1319. DOI: 10.1162/089976698300017467.

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

[41] Vitor R. M. Elias et al. 'Adaptive Graph Filters in Reproducing Kernel Hilbert Spaces: Design and Performance Analysis'. In: *IEEE Trans. on Signal and Info. Process. over Net.* 7 (2021), pp. 62–74. DOI: 10.1109/TSIPN.2020.3046217.

[42] Ming-Yu Liu et al. 'Entropy rate superpixel segmentation'. In: *CVPR 2011*. 2011, pp. 2097–2104. DOI: 10.1109/CVPR.2011.5995323.

[43] Francesco Bianconi and Antonio Fernández. 'Evaluation of the effects of Gabor filter parameters on texture classification'. In: *Pattern Recognition* 40.12 (2007), pp. 3325–3335. DOI: 10.1016/j.patcog.2007.04.023.

[44] J.-K. Kamarainen, V. Kyrki and H. Kalviainen. 'Invariance properties of Gabor filter-based features-overview and applications'. In: *IEEE Trans. on Img.Process.* 15.5 (2006), pp. 1088–1099. DOI: 10.1109/TIP.2005.864174.

[45] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. 'Deep learning'. In: *Nature* 521 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[46] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: https://www.deeplearningbook.org/ (visited on 17th Mar. 2023).

[47] Ivan Nunes da Silva et al. 'Introduction'. In: *Artificial Neural Networks : A Practical Course*. Springer International Publishing, 2017, pp. 3–19. DOI: 10.1007/978-3-319-43162-8_1. (Visited on 5th Dec. 2022).

[48] Bin Ding, Huimin Qian and Jun Zhou. *Activation functions and their characteristics in deep neural networks*. 2018, pp. 1836–1841. DOI: 10.1109/CCDC.2018.8407425.

[49] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 29th Jan. 2017. arXiv: 1412.6980[cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 1st Oct. 2022).

[50] Shibani Santurkar et al. 'How Does Batch Normalization Help Optimization?' In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[51] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. DOI: 10.48550/arXiv.1502.03167.

[52] Federico Girosi, Michael Jones and Tomaso Poggio. 'Regularization Theory and Neural Networks Architectures'. In: *Neural Comp.* 7.2 (1995), pp. 219–269. DOI: 10.1162/neco.1995.7.2.219.

[53] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015.

[54] Mohammad A. Salahuddin et al. 'Time-based Anomaly Detection using Autoencoder'. In: *2020 16th International Conference on Network and Service Management (CNSM)*. 2020 16th International Conference on Network and Service Management (CNSM). 2020, pp. 1–9. DOI: 10.23919/CNSM50824.2020.9269112.

[55] M. A. Kramer. 'Autoassociative neural networks'. In: *Computers & Chemical Engineering*. Neutral network applications in chemical engineering 16.4 (1st Apr. 1992), pp. 313–328. ISSN: 0098-1354. DOI: 10.1016/0098-1354(92)80051-A. URL: https://www.sciencedirect.com/science/article/pii/009813549280051A (visited on 27th Feb. 2023).

[56] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: 10.48550/arXiv.1505.04597.

[57] Olivier Rukundo and Hanqiang Cao. 'Nearest neighbor value interpolation'. In: *Int. J. Adv. Comput. Sci. Appl.* 3.4 (2012), pp. 25–30. DOI: 10.14569/IJACSA.2012.030405.

[58] Guy S. Handelman et al. 'Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods'. In: *American Journal of Roentgenology* 212.1 (2019), pp. 38–43. DOI: 10.2214/AJR.18.20224.

[59] Konstantinos Makantasis et al. 'Deep supervised learning for hyperspectral data classification through convolutional neural networks'. In: *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2015, pp. 4959–4962. DOI: 10.1109/IGARSS.2015.7326945.

[60] 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), p. 3387. DOI: 10.3390/rs12203387.

[61] Song Xiangyu. *SSIIFD*. original-date: 2022-09-29T06:56:03Z. 8th Dec. 2022. URL: https://github.com/xiangyusong19/SSIIFD_Hyperspectral-Anomaly-Detection (visited on 8th Dec. 2022).

[62] RSIDEA. *Project Auto-AD*. original-date: 2021-11-11T01:25:49Z. 5th Dec. 2022. URL: https://github.com/RSIDEA-WHU2020/Auto-AD (visited on 8th Dec. 2022).

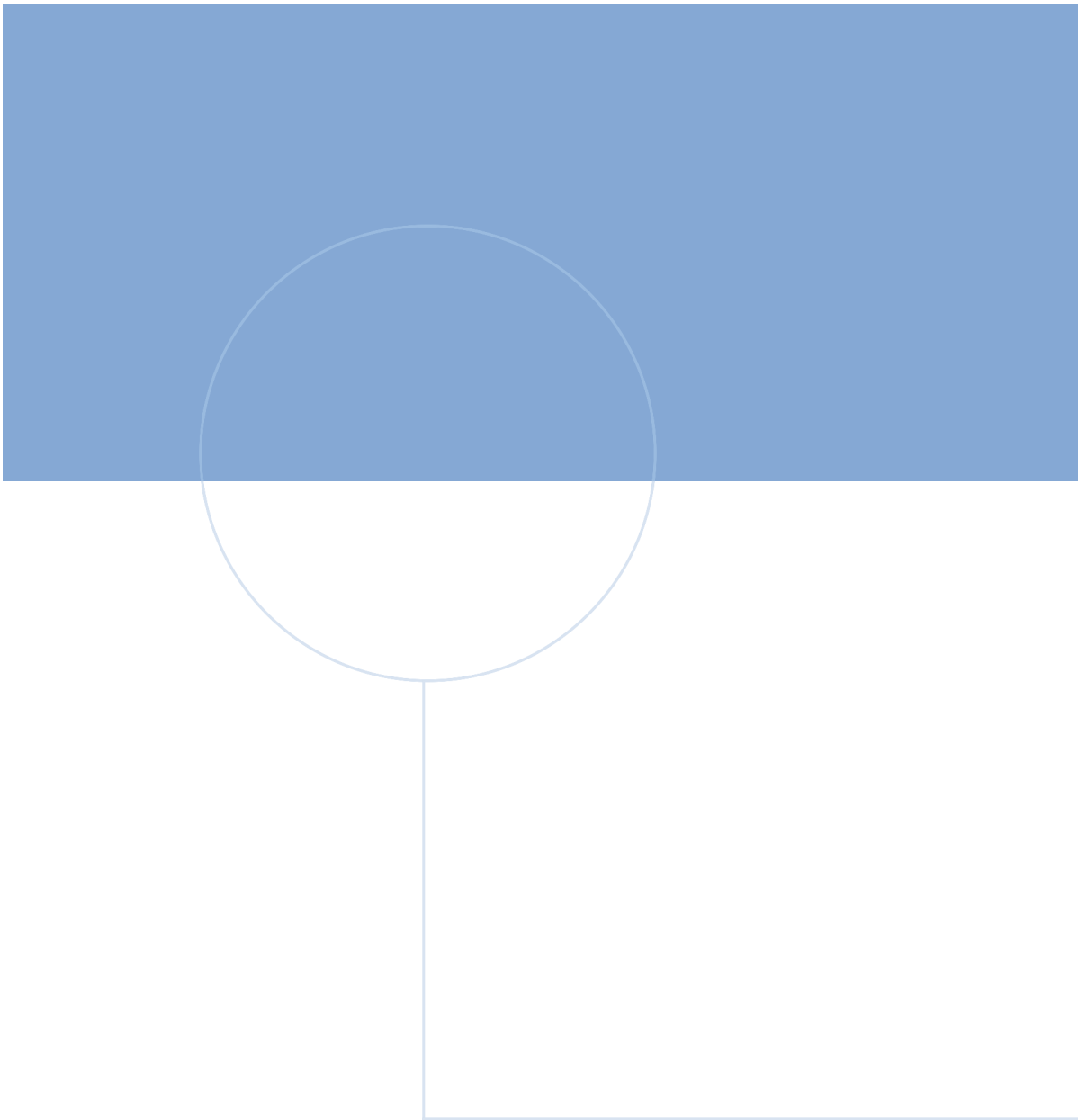
[63] A. P. Stakhov. 'The Generalized Principle of the Golden Section and its applications in mathematics, science, and engineering'. In: *Chaos, Solitons & Fractals* 26.2 (2005), pp. 263–289. ISSN: 0960-0779. DOI: 10.1016/j.chaos.2005.01.038. URL: https://www.sciencedirect.com/science/article/pii/S096007790500144X.

[64] *PyTorch*. URL: https://www.pytorch.org (visited on 13th May 2023).

[65] *Hyperparameter tuning with Ray Tune — PyTorch Tutorials 2.0.1+cu117 documentation*. URL: https://pytorch.org/tutorials/beginner/hyperparameter_tuning_tutorial.html (visited on 22nd May 2023).

[66] Shangzhen Luan et al. 'Gabor Convolutional Networks'. In: *IEEE Trans. on Image Process.* 27.9 (2018), pp. 4357–4366. DOI: 10.1109/TIP.2018.2835143.

[67] Georgios Zoumpourlis et al. 'Non-linear Convolution Filters for CNN-Based Learning'. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 4771–4779. DOI: 10.1109/ICCV.2017.510.



NTNU

Norwegian University of
Science and Technology