UNSUPERVISED ANOMALY DETECTION USING SimSID ALGORITHM

A PROJECT REPORT

Submitted by

SWETHA R

(2023176038)

A report for the phase-I of the project
submitted to the Faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment
for the award of the degree
of

MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SPECIALIZATION IN AI & DS

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600 025

JAN 2025

ANNA UNIVERSITY

CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **UNSUPERVISED ANOMALY DETECTION USING SimSID ALGORITHM** is the bonafide work of **SWETHA R (2023176038)** who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

PLACE:CHENNAI
DATE:

Dr. S. SRIDHAR
PROFESSOR
PROJECT GUIDE
DEPARTMENT OF IST, CEG
ANNA UNIVERSITY
CHENNAI 600025

COUNTERSIGNED

Dr. S. SWAMYNATHAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600025

ABSTRACT

Radiographic images, particularly chest X-rays, are crucial in medical diagnostics as they provide detailed insights into the internal anatomy of patients. These images exhibit structural consistency across patients due to standardized imaging protocols, making them ideal for automated anomaly detection. However, the increasing volume of radiographic data necessitates advanced techniques for efficient and accurate analysis.

Radiographic images hold great diagnostic value, but manually identifying subtle anomalies, such as small lesions, remains a challenge, prone to human error. Existing automated methods depend heavily on labeled datasets, which are often unavailable for rare or subtle conditions. Additionally, these methods struggle with noise or minor variations in image data, limiting their effectiveness in clinical applications.

The proposed solution introduces SimSID (Simple Space-Aware Memory Matrix for In-painting and Detecting anomalies), an unsupervised anomaly detection method tailored for radiographic images. A key technique in SimSID is feature inpainting, which is used to reconstruct missing or altered regions by capturing recurring anatomical patterns through a space-aware memory matrix. Anomalies are detected by measuring discrepancies between reconstructed and original images. Evaluated on the Chexpert Chest X-ray dataset, SimSID demonstrates robust performance, handling noise and minor abnormalities while reducing dependence on labeled datasets. Metrics such as Mean Squared Error (MSE), Kullback-Leibler (KL) divergence, adversarial loss, and anomaly score were calculated to evaluate the model's performance. These metrics show significant improvements in anomaly detection. This method shows promise for clinical applications, enhancing anomaly detection without requiring manual annotations.

திட்டப்பணி சுருக்கம்

ரேடியோகிராபிக் படங்கள், குறிப்பாக மார்பு எக்ஸ்ரேபடங்கள், மருத்துவ நோயறிதலில் மிக முக்கியமானவை, ஏனெனில் அவை நோயாளிகளின் உட்புற உடல் அமைப்பில் விரிவான விளக்கங்களை வழங்குகின்றன. இந்தப் படங்கள் பிழைமற்ற படமெடுக்கல் முறைமைகள் காரணமாக நோயாளிகள் மத்தியில் கட்டமைப்பியல் ஒத்திசைவுகளை காண்பிக்கின்றன, இதனால் அவை தானாகவே அசாதாரண நிலைகளைக் கண்டறிய உதவுகின்றன. இருப்பினும், ரேடியோகிராபிக் தரவுகளின் அதிகரிக்கும் பருமன் துல்லியமான மற்றும் வேகமான பகுப்பாய்விற்கான மேம்பட்ட தொழில்நுட்பங்களைத் தேவைப்படுத்துகிறது.

ரேடியோகிராபிக் படங்கள் மிகவும் விலைமதிப்புள்ள மருத்துவ அறிகுறிகள் வழங்கினாலும், சிறிய கட்டிகளாகிய அசாதாரணங்களைக் கண்டறிவது மனிதரின் தவறுகளுக்கு இடமளிக்கக்கூடும். தற்போதைய தானாக வேலைசெய்யும் முறைமைகள் பெரும்பாலும் குறைந்த மொத்த அளவிலான அல்லது அசாதாரண நிலைகள் இல்லாத தரவுத்தளங்களுக்குப் பேரொற்றி செயல்படுகின்றன. மேலும், இந்த முறைமைகள் பட தரவுகளிலுள்ள குருட்டான சரிவுகளோ அல்லது சிறிய வேறுபாடுகளோ மாறாதன. இது அவர்களின் செயல்திறனை மருத்துவ பயன்பாட்டில் சிரமமாக்குகிறது.

விளக்கமாக கூறப்பட்ட தீர்வு, SimSID (சிம்பிள் ஸ்பேஸ்-அவரேர் மெமரி மேட்ரிக்ஸ் ஃபார் இன்-பெயிண்டிங் அண்ட் டெடெக்டிங் அனாமலீஸ்) என்ற பெயரில் ஒரு உதாரணம் இல்லாத அசாதாரணத் தன்மையை கண்டறிதல் முறைமையை அறிமுகப்படுத்துகிறது. SimSID இன் முக்கியமான தொழில்நுட்பமாக உள்ளன ஃபீச்சர் இன்-பெயிண்டிங், இது காணாமற்பட்ட அல்லது மாறிய பகுதிகளை மறுபடியும் உருவாக்க உதவுகின்றது, இது ஸ்பேஸ்-அவரேர் மெமரி மேட்ரிக்ஸ் மூலம் அடிக்கடி மீண்டும் வரும் உடல் அமைப்புகளைப் பிடிக்கும். அசாதாரணங்களை மீண்டும் உருவாக்கப்பட்ட படங்களுடன் ஒப்பிடுவதன் மூலம் கண்டறியப்படுகிறது. செக்ஸ்பர்ட் மார்பு எக்ஸ்-ரே தரவுத்தளத்தில் மதிப்பீடு செய்தபோது, SimSID பல நுழைவுகள் மற்றும் சிறிய மாறுபாடுகளை கையாளும் திறன் காண்பித்து, குறைந்த அளவிலான தரவுத்தளங்களின் மேலாண்மையை குறைக்கும் செயல்திறனை விளக்குகிறது. மொத்தமாக, மையஸ்கொயர்டு பிழை, குல்பேக்-லைப்லர் விசிறி, எதிர்மறை இழப்பு மற்றும் அசாதாரணத்திற்கான மதிப்பெண் போன்ற மதிப்பீடுகள் கணக்கிடப்பட்டுள்ளன, இதன் மூலம் முறைமையின் செயல்திறனுக்கு முக்கியமான மேம்பாடுகளை உறுதிப்படுத்துகின்றன. இந்த முறைமை, தானாகவே அறிகுறிகளை கண்டறிவதற்காக மருத்துவ பயன்பாடுகளில் பெரிய பலனை அளிக்கும் என்று காட்டுகிறது.

ACKNOWLEDGEMENT

It is my privilege to express my deepest sense of gratitude and sincere thanks to **Dr. S. SRIDHAR ,** Professor , Department of Information Science and Technology, College of Engineering, Guindy, Anna University, for his constant supervision, encouragement, and support in my project work. I greatly appreciate the constructive advice and motivation that was given to help me advance my project in the right direction.

I am grateful to **Dr. S. SWAMYNATHAN,** Professor and Head, Department of Information Science and Technology, College of Engineering Guindy, Anna University for providing us with the opportunity and necessary resources to do this project.

I would also wish to express my deepest sense of gratitude to the Members of the Project Review Committee: **Dr. S. SRIDHAR,** Professor, **Dr. G. GEETHA,** Associate Professor, **Dr. D. NARASHIMAN,** Teaching Fellow Department of Information Science and Technology,College of Engineering Guindy, Anna University, for their guidance and useful suggestions that were beneficial in helping me improve my project.

I also thank the faculty member and non teaching staff members of the Department of Information Science and Technology, Anna University, Chennai for their valuable support throughout the course of our project work.

SWETHA R

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

A	Anomaly Score
AUC	Area Under the Receiver Operating Characteristic
CNN	Convolutional Neural Network
D	Discriminator
DCGAN	Deep Convolutional Generative Adversarial Network
DCNN	Deep Convolutional Neural Network
DDPM	Denoising Diffusion Probabilistic Models
E	Encoder
EDC	Encoder-Decoder Contrast
$\mathbf{G}_s$	Student Generator
$\mathbf{G}_t$	Teacher Generator
I	Image
KL	Kullback-Leibler Divergence
MM	Memory Matrix
MSE	Mean Squared Error
OCT	Optical Coherence Tomography
PSNR	Peak Signal-to-Noise Ratio
RIAD	Reconstruction-by-Inpainting Anomaly Detection
SimSID	Simple Space-Aware Memory Matrix for In-painting and Detecting anomalies
SSIM	Structural Similarity Index
SQUID	Space-aware Memory Queues for In-painting and Detecting anomalies

CHAPTER 1

INTRODUCTION

This chapter presents a novel framework for anomaly detection in radiography images using feature extraction and inpainting techniques. It outlines the problem, defines the objectives, and discusses challenges associated with anomaly detection in medical imaging.

1.1 OVERVIEW

Radiography is a medical imaging technique that uses X-rays to produce detailed images of the internal structures of the body. These images, known as radiographic images, provide valuable insights into the condition of various tissues and organs, helping healthcare professionals diagnose a wide range of medical conditions. Radiography is commonly used to detect fractures, tumors, infections, and other abnormalities in bones and soft tissues. The clarity and detail of these images are crucial for making informed clinical decisions.

Anomalies in radiographic images refer to irregularities or deviations from the normal structure and function of the body as seen in the images. Examples of anomalies include fractures, infections, abnormal tissue growths, or changes in organ structure that may indicate underlying health issues. Timely and accurate detection of these anomalies is critical for effective treatment and patient management.

Traditional anomaly detection methods primarily rely on visual inspection by radiologists, which can be labor-intensive and prone to human error. The challenge is further compounded by the subtle nature of some

anomalies, making them difficult to distinguish, particularly in complex cases where the differences may be minor.

The limited availability of experienced radiologists in many healthcare facilities can lead to delayed diagnoses, resulting in suboptimal patient outcomes. In addition, the increasing volume of radiographic images generated in clinical practice exacerbates the workload for radiologists, contributing to stress and potential burnout. Thus, there is a growing need for automated systems that can efficiently detect anomalies and provide reliable diagnostic support. Automated approaches capable of accurately identifying anomalies are essential to reduce the burden on healthcare professionals and ensure timely interventions.

To address these limitations, this research proposes an advanced anomaly detection framework that leverages feature-level inpainting and memory-based augmentation. Unlike traditional pixel-based models, this approach focuses on filling in missing or altered feature information through inpainting, allowing for a more nuanced understanding of both healthy and anomalous regions. By utilizing a hierarchical memory matrix, the framework maintains the spatial consistency of image features, enhancing retrieval accuracy and detection reliability.

The proposed framework also integrates convolutional neural networks (CNNs) for robust feature extraction, capturing essential characteristics of the images. Additionally, the incorporation of transformer-based augmentation enables the model to generalize effectively, avoiding memorization of specific patterns while still detecting subtle anomalies. This holistic approach ensures that the system can detect anomalies with high accuracy and efficiency, thereby contributing to improved diagnostic reliability.

The system aims to empower healthcare professionals by providing detailed diagnostic reports that highlight detected anomalies, associated confidence levels, and suggestions for further investigation. By streamlining the diagnostic process and minimizing the potential for human error, the proposed framework not only enhances the accuracy of radiographic interpretations but also supports better clinical decision-making and patient outcomes.

1.2 PROBLEM STATEMENT

Current methods for anomaly detection in radiography often rely on pixel-based reconstructions or supervised learning, which require large amounts of labeled data and struggle to capture subtle abnormalities effectively. However, the scarcity of labeled data, especially for rare anomalies, limits the applicability and scalability of these approaches.

These challenges are addressed by developing a feature-based anomaly detection framework that eliminates the need for labeled datasets. By using feature extraction, inpainting, and a memory matrix, the system aims to enhance anomaly detection without relying on pixel-level reconstruction or supervised learning techniques.

1.3 OBJECTIVE

The primary objective of this project is to design an automated anomaly detection system for radiography images that addresses challenges related to subtle anomalies, varying imaging conditions, and the need for consistent detection across diverse datasets. By employing feature extraction, inpainting, and memory-based augmentation, the framework enhances the system's ability to detect abnormalities with high precision while ensuring spatial consistency. The incorporation of transformer-based augmentation

guarantees effective generalization, minimizing the risk of overfitting to specific data patterns.

Ultimately, this project aims to provide healthcare professionals with a reliable tool that delivers detailed insights into anomalies, reduces diagnostic workloads, and facilitates timely medical intervention, thereby contributing to improved patient care and outcomes.

1.4 BLOCK DIAGRAM

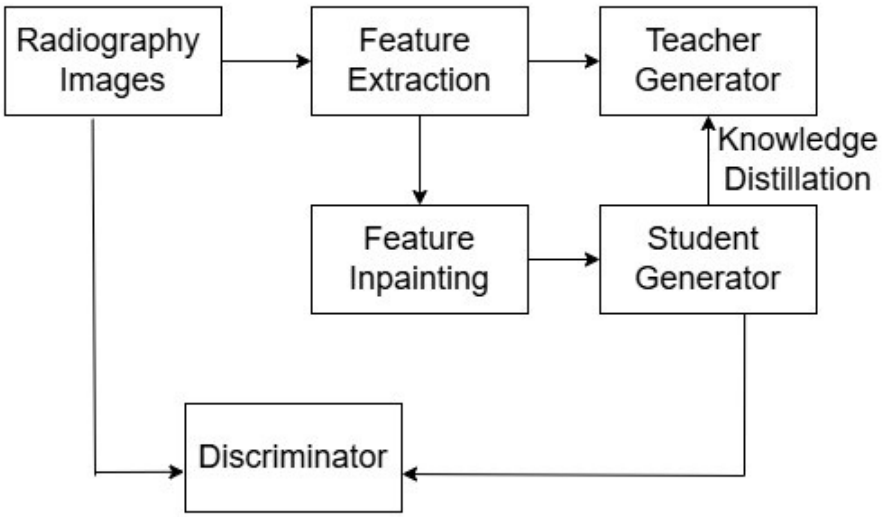

Figure 1.1: Block Diagram

1.4.1 MODULE EXPLANATION

1. INPUT IMAGES

This block represents the input data for the system, consisting of chest radiograph images. These images are analyzed to detect anomalies, such

as abnormalities. Healthy images are typically used for training, while the system identifies deviations in test images.

2. FEATURE EXTRACTION

This block extracts meaningful features from the input images using a Convolutional Neural Network (CNN). The input images are divided into non-overlapping patches, and features are extracted from these patches. These features serve as the foundation for anomaly detection by capturing patterns, textures, and structures in the radiographs.

3. FEATURE INPAINTING

The extracted features are inpainted using a memory matrix. This block ensures that missing or corrupted features are filled in, making the features more robust. The inpainting process prevents the model from memorizing specific images and enhances the system's ability to generalize during anomaly detection.

4. GENERATOR

a) TEACHER GENERATOR: The teacher generator produces reconstructed images from the extracted features. It serves as a baseline for training the student generator through knowledge distillation, ensuring the student generator mimics the teacher's outputs during training.

b) STUDENT GENERATOR: This block reconstructs the images from the inpainted features. The purpose of this generator is to mimic the teacher generator and its functionality. The reconstructed images are compared with the input images to identify anomalies.

5. DISCRIMINATOR

This block differentiates between original images and reconstructed images. It is part of an adversarial framework and assesses the quality of the reconstructed images. The discriminator calculates an anomaly score by measuring the discrepancies between the input images and reconstructed images. Higher discrepancies indicate anomalies.

1.5 ORGANIZATION OF THE REPORT

This section provides an overview of the project structure, outlining the sequence and content of each chapter. It serves as a roadmap for navigating through the project and understanding its progression.

Chapter 2: This chapter provides a succinct literature review that highlights earlier research in the field of anomaly detection in radiography images. It offers a summary of current research, methods, and strategies employed in anomaly detection, with a focus on feature inpainting, memory-based approaches, and deep learning methodologies.

Chapter 3: This chapter covers the project's system design, datasets, and methods. It details the datasets utilized, the architecture of the system for anomaly detection in radiography images, and the approach to feature inpainting and spatial-aware memory.

Chapter 4: This chapter provides an overview of the actual procedures used in developing the feature inpainting mechanism, memory matrix implementation, and the integration of the teacher-student generator and discriminator. It addresses dataset preparation, feature extraction, and the training procedures for effective anomaly detection.

Chapter 5: This chapter displays the final outcome of the project and future enhancements that can be made.

Chapter 6: This chapter summarizes the project's findings and highlights the successful implementation of the anomaly detection system using a memory-augmented framework and adversarial learning techniques.

CHAPTER 2

LITERATURE SURVEY

This chapter delves into how unsupervised anomaly detection techniques are revolutionizing the analysis of medical images, particularly in identifying anomalies without the need for labeled datasets. By exploring existing research, the transformative impact of these methodologies on the accuracy and efficiency of medical image classification. The survey navigates through the literature, dissecting various applications and innovative approaches in unsupervised anomaly detection, such as diffusion models, feature in-painting, and adversarial learning. An analysis is conducted on how these techniques enhance the detection of anomalies across different imaging modalities, ultimately leading to improved diagnostic capabilities. Through this literature review, we aim to provide a comprehensive understanding of the current state of unsupervised anomaly detection in medical imaging and its potential to advance clinical practice.

2.1 ANOMALY DETECTION BASED ON DIFFUSION MODELS

Julia Wolleb et al.[1] proposed a novel methodology that leverages Denoising Diffusion Probabilistic Models (DDPMs) in a two-part process for effective anomaly detection in medical images. In the first part, the DDPM is trained on datasets containing images of healthy and diseased subjects, allowing the model to learn the underlying distribution of healthy anatomy. During evaluation, the model generates a synthetic healthy image from a noisy input, and the anomaly map is created by calculating the difference between the original and the generated images, effectively highlighting the pathological regions. The authors evaluate their weakly supervised anomaly

detection method using two primary metrics: the Dice score, which assesses the overlap between predicted anomaly maps and ground truth labels, and the Area Under the Receiver Operating Characteristic (AUROC), which measures the model's ability to distinguish between diseased and healthy images. These metrics provide a comprehensive understanding of the model's performance on the BRATS2020 and CheXpert datasets. The results showcase the generation of detailed anomaly maps and emphasize the importance of hyperparameter tuning to avoid artifacts. The significance of this work lies in its ability to simplify the training process for anomaly detection in medical imaging, potentially enhancing diagnostic accuracy and efficiency in clinical settings. Furthermore, applications of this method could extend to various medical imaging tasks, providing a robust tool for radiologists and healthcare professionals in identifying and analyzing anomalies in patient images.

Muzaffer Özbey et al.[2] developed a system that introduces SynDiff, an adversarial diffusion model designed for unsupervised medical image translation. This model employs a conditional diffusion process that progressively maps noise and source images to target images, utilizing a cycle-consistent architecture that integrates both diffusive and non-diffusive modules for bilateral translation between modalities. This approach allows for effective training on unpaired datasets, enhancing the synthesis of medical images while maintaining anatomical accuracy. The results demonstrate that SynDiff outperforms state-of-the-art models, including various Generative Adversarial Networks (GANs) and diffusion models, in multi-contrast MRI and MRI-CT translation tasks. Performance metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) were used for evaluation, with SynDiff achieving an average improvement of 1.5 dB PSNR and 3.5 percentage SSIM over non-attentional GANs, as well as even greater improvements over attentional GANs and diffusion models. The significance of these results was validated through non-parametric Wilcoxon signed-rank tests,

indicating statistical significance (p ¡ 0.05) in performance differences. SynDiff has significant applications in medical imaging, particularly in synthesizing images from different modalities, which can enhance diagnostic capabilities and improve imaging protocols. Its ability to generate high-fidelity images from unpaired datasets is crucial in clinical settings where acquiring paired images is challenging, thus broadening the scope of medical imaging applications.

2.2 ANOMALY DETECTION BASED ON GAN

Thomas Schlegl et al.[3] developed a system using a deep convolutional generative adversarial network (DCGAN) to learn a manifold of normal anatomical variability, which is crucial for detecting anomalies in imaging data. The methodology includes training on 2D image patches extracted from clinical optical coherence tomography (OCT) volumes of healthy subjects, ensuring that the data excludes fluid regions. The OCT volumes are normalized and resized to standardize the input for GAN training. The GAN architecture features four fractionally-strided convolution layers in the generator and four convolution layers in the discriminator, based on the stable DCGAN architecture proposed by Radford et al. Once trained, the model employs a novel mapping approach to evaluate new data, scoring anomalies based on their fit within the learned distribution of normal images. This enables effective detection of both known and unknown anomalies, such as retinal fluid and hyperreflective foci. Evaluation metrics includes ROC curves, AUC, sensitivity, specificity, precision, and recall provide comprehensive insights into the model's accuracy, while residual and discrimination scores assess its effectiveness at both image and pixel levels. The significance of this research lies in its ability to automate the detection of imaging markers related to disease progression and treatment monitoring without requiring extensive annotated datasets, thereby enhancing the efficiency of marker discovery. The findings have crucial applications in medical imaging, particularly in ophthalmology, as

they aid in the early diagnosis and monitoring of retinal diseases. Furthermore, the methodology is adaptable for various imaging modalities and medical conditions, making it a versatile tool for improving diagnostic accuracy and treatment strategies.

Bao Nguyen et al.[4]proposed method for unsupervised region-based anomaly detection in brain MRI utilizes a deep convolutional neural network (DCNN) to reconstruct missing healthy brain regions through adversarial training, where a generator produces realistic reconstructions, and a discriminator evaluates their authenticity. A masked sliding window operation is employed to analyze each query slice by applying a square mask and passing the masked image to the generator. The reconstruction loss is calculated to identify high-loss areas, which are visualized as a heatmap to localize potential anomalies. Superpixel segmentation is then applied based on the heatmap to enhance tumor boundary identification. Evaluation metrics focus on segmentation performance, achieving a mean Dice score indicating strong overlap between predicted and actual tumor regions, along with a standard deviation that highlights consistent performance across varying tumor sizes. Additionally, reconstruction loss helps identify areas with tumors, demonstrating the effectiveness of the inpainting network in reconstructing healthy regions. The significance of this method lies in its potential for early brain tumor detection, automation of the segmentation process, improved surgical planning, and adaptability to various tumor sizes, paving the way for enhanced patient care and outcomes in clinical practice.

2.3 UNSUPERVISED ANOMALY DETECTION

Jia Guo et al.[5] proposed an unsupervised anomaly detection method called Encoder-Decoder Contrast (EDC), which optimizes the entire network to enhance feature extraction from normal medical images, thereby

reducing biases from pre-trained models and improving performance through a new global cosine distance objective function. The methodology involves a feature reconstruction approach combined with contrastive learning to optimize both the encoder and decoder simultaneously, effectively addressing pattern collapse by introducing a global cosine distance metric that enhances stability and performance in distinguishing normal from anomalous images. Extensive experiments across various medical image modalities demonstrate that this approach outperforms existing state-of-the-art methods in unsupervised anomaly detection. To assess the performance of the proposed method, the paper employs several evaluation metrics, including Area Under the Receiver Operating Characteristic (AUC), F1-score (F1), accuracy (ACC), sensitivity (SEN), and specificity (SPE). AUC evaluates overall performance, while F1, ACC, SEN, and SPE focus on the detection of positive and negative samples, with the optimal operating threshold determined by the best F1 score. Overall, the Encoder-Decoder Contrast (EDC) method significantly enhances unsupervised anomaly detection in medical images by optimizing the entire network to improve feature extraction from normal images. This approach addresses data imbalance, improves detection performance across various medical imaging modalities, and aids in clinical diagnosis by generating informative anomaly maps.

Tiange Xiang et al.[6] , proposed the SQUID system, which utilizes Space-aware Memory Queues to enhance anomaly detection in radiography images. It employs an in-painting technique that reconstructs masked image areas, allowing the model to learn normal patterns effectively. By leveraging a memory queue, SQUID captures and aggregates features from previous training data, which aids in identifying unseen anomalies during inference. This approach significantly improves performance, surpassing existing methods in unsupervised anomaly detection metrics. The SQUID methodology involves dividing input images into non-overlapping patches and applying a generator

and discriminator model to learn robust feature representations. The model is trained to identify normal anatomical structures and distinguish them from anomalies by leveraging the consistent patterns found in radiography images. Additionally, hyper-parameters such as loss weights and learning rates are carefully set to optimize the training process, ensuring effective anomaly detection. The evaluation of SQUID's performance is conducted using several standard metrics, including the Receiver Operating Characteristic (ROC) curve, Area Under the ROC Curve (AUC), Accuracy (Acc), and F1-score (F1). These metrics provide a comprehensive assessment of the model's ability to detect anomalies effectively. The models are trained independently multiple times to ensure the reliability of the results. Furthermore, SQUID is compared against 13 major baselines to validate its performance in unsupervised anomaly detection. Its significance lies in effectively identifying anomalies in radiography images by leveraging structured anatomical information, which enhances accuracy in medical imaging. SQUID's application extends to improving diagnostic processes in healthcare, particularly in identifying unseen or modified patterns in chest X-rays, thereby aiding in early disease detection and treatment planning.

Vitjan Zavrtanik et al.[**?**], proposed a system called RIAD (Reconstruction-by-Inpainting Anomaly Detection), which focuses on visual anomaly detection by reconstructing images from randomly removed regions. This self-supervised approach enhances anomaly detection capabilities by addressing the limitations of traditional auto-encoders, which often reconstruct anomalous areas too well, thus masking the anomalies. The system employs a U-Net-based architecture and is evaluated on the MVTec dataset, demonstrating state-of-the-art performance in anomaly detection tasks. The methodology of RIAD involves several key steps: first, the input image is divided into a grid of square regions, allowing for systematic sampling of areas to be removed for inpainting. Next, randomly selected regions are set to zero, effectively masking them and simulating the presence of anomalies. A trained neural

network is then used to reconstruct the original image from the altered version, focusing on the masked regions. Anomaly detection is performed by analyzing the difference between the original and reconstructed images, leveraging the assumption that the network will struggle to accurately reconstruct these masked areas. The evaluation of the proposed method involves two primary tasks: image-level anomaly detection and pixel-level anomaly localization, with the standard metric used for both being the ROC AUC score, which is widely recognized for its effectiveness in measuring performance. Additionally, a pixel-based ROC AUC score is utilized specifically for assessing the anomaly localization capability of the method. RIAD is significant as it effectively reconstructs images with missing regions, thereby enhancing anomaly detection capabilities. This method is applicable in various fields such as manufacturing quality control, surveillance for detecting unusual activities, and medical imaging for identifying abnormalities, showcasing its versatility and robustness in real-world scenarios.

2.4 LITERATURE SURVEY SUMMARY

This chapter explores the transformative role of unsupervised anomaly detection techniques in medical image analysis, emphasizing their ability to identify anomalies without relying on labeled datasets. It highlights innovative methodologies such as diffusion models, GANs, and feature inpainting. Techniques like DDPM are employed to generate synthetic healthy images for anomaly detection, while adversarial diffusion models facilitate unsupervised medical image translation. Deep convolutional GANs learn normal anatomical variability to enhance anomaly detection, and region-based anomaly detection methods utilize adversarial training to improve performance. Additionally, EDC is introduced to enhance feature extraction across various medical modalities. Space-aware Memory Queues capture normal patterns in imaging, while techniques that reconstruct images from masked regions

contribute to improved anomaly detection. This comprehensive review not only emphasizes key achievements but also addresses encountered challenges, setting the stage for our unique contributions to this crucial field. In conjunction with these advancements, the proposed method introduces a comprehensive framework that integrates feature inpainting, a memory matrix, and transformer-based augmentation, enhancing the detection of anomalies in medical imaging. This approach aims to effectively learn normal patterns while facilitating the identification of anomalies, ultimately improving diagnostic capabilities and clinical practice in medical imaging.

CHAPTER 3

SYSTEM ARCHITECHTURE

System Architecture focuses on designing and managing complex systems in a way that makes it easier to reason about their components and their interactions. It helps overcome the challenges of understanding, designing, and describing complex systems by providing a clear, structured approach. This is essential for ensuring that systems function properly, meet requirements, and can be scaled or modified as needed. Figure 3.1 shows the system architecture of SimSID algorithm.

3.1 DATASET

The dataset consists of training, test, and validation data in JPEG format. The training dataset includes 1024 normal images from the Chexpert Chest X-ray dataset [7], which are used to train the model to detect anomalies. The test dataset comprises 100 normal images and 100 abnormal images, which are used to evaluate the model's performance on unseen data. The validation dataset consists of 50 normal images and 50 abnormal images, used for model validation during training. Normal images represent healthy cases, while abnormal images contain anomalies such as pneumonia. The training data is used to teach the model the features of healthy chest radiographs, and the test and validation datasets are employed to assess the model's ability to detect abnormalities in chest radiograph images.

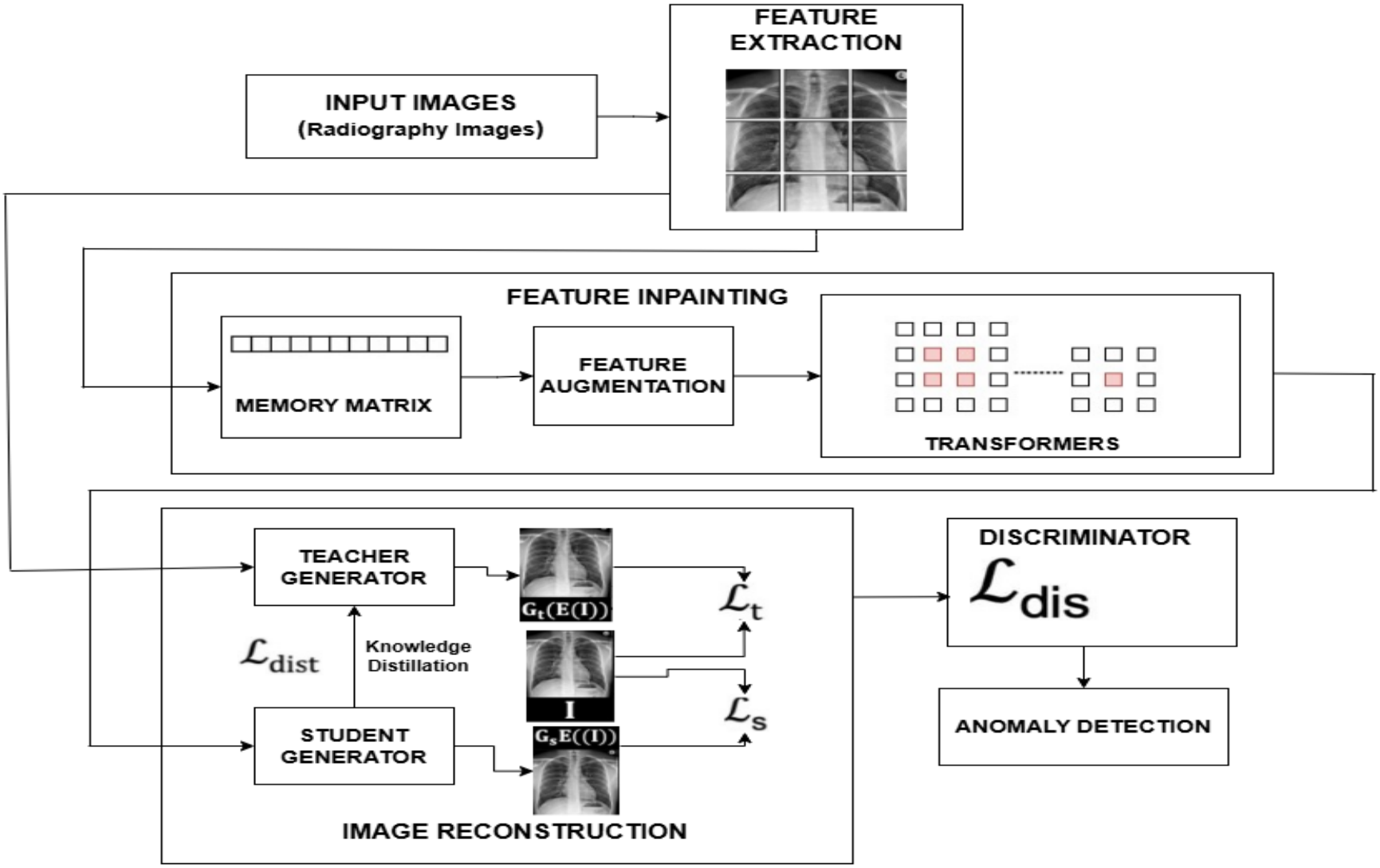

Figure 3.1: System Architecture of SimSID Algorithm

3.2 DATA COLLECTION AND PREPROCESSING

Data collection involves gathering relevant information from various sources, ensuring its accuracy and completeness. Preprocessing is the initial step in data analysis, addressing issues like missing values, outliers, and formatting inconsistencies. This step is essential to enhance data quality, making it suitable for further analysis and modeling. Effective data collection and preprocessing are crucial for obtaining reliable insights and maximizing the utility of the gathered information.

3.2.1 DATA COLLECTION

Data collection is the foundation of the research process, encompassing the systematic gathering of relevant information to address specific research questions or objectives. For this project, the data collection process focuses on gathering radiography images that represent both healthy and abnormal cases. Data collection is the initial step in the pipeline, where relevant images are gathered from various sources. In this case, the dataset used is from the Chexpert Chest X-ray dataset, which contains both normal (healthy) and abnormal (pneumonia) radiography images. The images are organized into different folders for training, testing, and validation. The training dataset includes 1024 normal images, which are used for model training. The test dataset consists of 100 normal images and 100 abnormal images, which are used to evaluate the model's performance. Additionally, the validation dataset consists of 50 normal images and 50 abnormal images, which are used to validate the model during training.

3.2.2 PREPROCESSING

Preprocessing is a crucial step in preparing the data for deep learning

models. For medical image analysis, such as anomaly detection in chest X-rays, common preprocessing steps include resizing images to a consistent resolution and normalizing pixel values. Below are the preprocessing steps followed in this project.

PREPROCESSING STEPS

• Load JPEG Images: The images are loaded from the dataset.

• Resize Images: The images are resized to a uniform size of 128x128 pixels. This step ensures that all input images have the same resolution, which is important for feeding them into neural networks.

• Convert to Tensor: The images are then converted to PyTorch tensors, which are required for input to a PyTorch model.

• Explore Preprocessed Data: Before moving on to the model training phase, it's essential to explore the preprocessed data to ensure that resizing and tensor conversion have been applied correctly. This step verifies that the preprocessing pipeline is functioning as expected and that the data is ready for use in the deep learning model.

• Train Model: The model is trained on the preprocessed images. It learns to detect anomalies in the images by analyzing the features extracted from both normal and abnormal X-ray images.

3.3 FLOW ARCHITECTURE

The architecture of the proposed system begins with data collection and preprocessing, followed by feature extraction, feature inpainting, generator

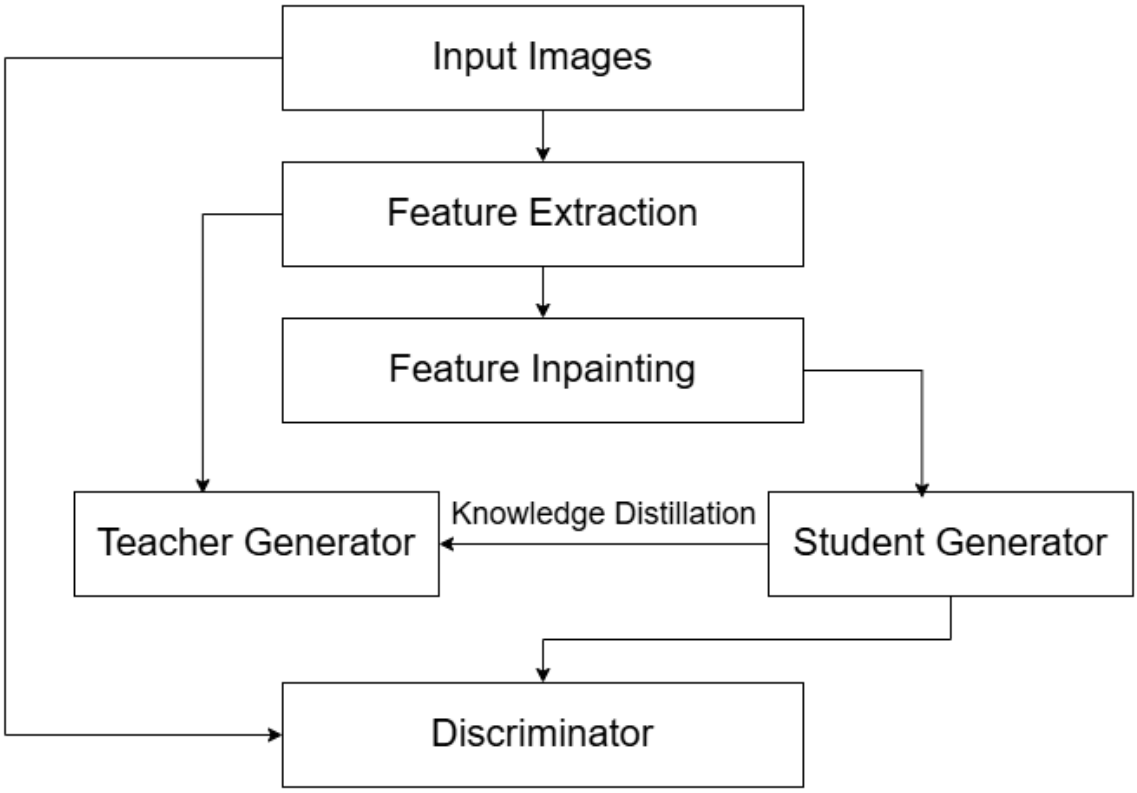

Figure 3.2: Architecture flow of SimSID Algorithm

and discriminator. The flow of the architecture can be outlined as follows: The input to the model is a chest X-ray image, which can be either normal or abnormal. The image is divided into non-overlapping patches, allowing the model to focus on localized regions, thereby enhancing both computational efficiency and the accuracy of anomaly detection. Each patch is passed through a CNN encoder to extract meaningful features representing anatomical information within the chest X-ray. The extracted features from each patch are stored in a Memory Matrix (MM) [8] [9], which serves as a repository of learned features from normal chest anatomy. To improve the feature representation, the extracted patch features are augmented using features from the Memory Matrix through a process called feature inpainting [10] [11]. This inpainting leverages the structural consistency of normal anatomy to create a more complete feature

representation. The model includes both a teacher generator (Gt) and a student generator (Gs): the teacher generator reconstructs the original image from the CNN-extracted features [12], providing a reference for the student generator. Meanwhile, the student generator reconstructs the image from the augmented features, guided by the teacher to produce more accurate reconstructions. The student generator is trained using knowledge distillation [13], where it learns from the teacher's reconstructions, enabling it to generalize better without overfitting to the training data. A discriminator is included to distinguish between real and generated images, ensuring that the student generator produces realistic and consistent outputs. The discriminator provides feedback to both the student and teacher generators, helping to improve their performance [14]. For anomaly detection, the reconstructed image is compared to the input image, and the discriminator calculates an anomaly score based on the differences observed. A high anomaly score indicates a significant deviation from the normal structure, suggesting the presence of an anomaly. Finally, based on the anomaly score, the model classifies the image as either normal or anomalous.

CHAPTER 4

IMPLEMENTATION

This chapter explains the implementation of the system and the various technologies that have been used.

4.1 FEATURE EXTRACTION

The input image is broken down into non-overlapping patches. Each patch is a fixed-size sub-image that contains specific information about a part of the whole image. By focusing on smaller sections of the image, the system can more easily identify local patterns or features, such as textures, edges, and shapes, which might be missed in a broader view. After dividing an image into patches, the next step is to extract meaningful features from these patches. This is achieved by using a Convolutional Neural Network (CNN), a specialized type of neural network designed to process images. CNNs are particularly good at identifying patterns in spatial data, such as the edges, textures, and shapes within each patch. Working with patches instead of the full image reduces computational complexity, as the model can process each patch separately, enabling parallel processing and faster analysis. CNNs apply filters to the patches, scanning for patterns or features like edges, gradients, or specific textures, with each layer gradually extracting more complex features as the network goes deeper. Pooling layers are used after convolution to reduce the size of the feature maps, making the model more efficient while preserving the key features of the image. The network employs hierarchical feature learning, where early layers may detect simple features like edges, while deeper layers may recognize more complex features, such as shapes, objects, or patterns.

Algorithm 4.1 Images to Patches Algorithm

Input: Chest Radiography Images
Output: Non-overlapping patches

1. **function** IMAGES_TO_PATCHES(image, patch_size)
2. SET patch_height, patch_width FROM patch_size
3. EXTRACT patches using sliding windows with size (patch_height, patch_width)
4. CONVERT patches to contiguous memory
5. RESHAPE patches to (batch_size, channels, num_patches, patch_height, patch_width)
6. **return** patches
7. **end function**

Algorithm 4.2 Feature Extraction

Input: Non-overlapping patches
Output: Feature information

1. **Class PatchBasedCNN:**
2. **Function __init__():**
3. INITIALIZE encoder with convolutional layers, batch norm, ReLU, and max pooling:
4. Conv(1→32), BatchNorm, ReLU, MaxPool
5. Conv(32→64), BatchNorm, ReLU, MaxPool
6. Conv(64→128), BatchNorm, ReLU, MaxPool
7. Conv(128→256), BatchNorm, ReLU, MaxPool
8. SET $fc = \text{None}$ and $fc\_input\_dim = 0$
9. **Function forward(x):**
10. Pass x through encoder
11. Compute fc_input_dim as total features per batch
12. Flatten x to shape $(\text{batch\_size}, fc\_input\_dim)$
13. If $fc == \text{None}$:
14. Initialize fc as Linear($fc\_input\_dim \rightarrow 64$)
15. Pass x through fc
16. Return x

The algorithm 4.1 defines a function image to patches that extracts patches from an image using a sliding window with a specified patch size. The image is divided into non-overlapping patches of the given shape. These patches are reshaped and returned for further processing.

The algorithm 4.2 implements a 'PatchBasedCNN' class, which consists of a series of convolutional layers for feature extraction. The encoder includes four convolutional blocks with batch normalization, ReLU activations, and max pooling. After encoding, the output is flattened and passed through a fully connected layer, initialized dynamically based on the encoder's output dimensions. The final output is returned for downstream tasks like anomaly detection.

4.2 FEATURE INPAINTING

In feature inpainting, the goal is to predict the missing or corrupted feature values using the context provided by the surrounding uncorrupted features. This allows for more robust inpainting, as it works with abstract, high-level information rather than dealing with pixel-level information. The process involves masking out certain parts of the feature map, and the model learns to fill in these gaps by leveraging the surrounding context and patterns. This can be particularly useful in anomaly detection tasks, where abnormal regions of an image need to be identified by comparing the inpainted features to expected normal patterns.

Algorithm 4.3 Feature Inpainting

Input: Patch features
Output: Augmented features

Explanation: Algorithm 4.3 takes patch features as input, performs feature augmentation by substituting or enhancing missing or incomplete features using

a learned memory matrix or by using inpainting mechanisms, and outputs augmented features that preserve spatial and contextual consistency.

Algorithm 4.4 SpaceAwareMemory

Class SpaceAwareMemory:
  Function __**init**__(*num_blocks*, *feature_dim*, *similarity_threshold*)**:**
    INITIALIZE *memory* as zero tensor of shape (*num_blocks*, *feature_dim*)
    STORE *similarity_threshold*
  Function update_memory(*features*, *patch_locations*)**:**
    For each *feature*, *location* **in** *features*, *patch_locations*:
      RETRIEVE *existing_feature* FROM *memory*[*location*]
      COMPUTE *similarity* USING cosine similarity between *feature* and *existing_feature*
      If *similarity* $<$ *similarity_threshold*:
        UPDATE *memory*[*location*] $=$ *feature*
  Function retrieve_features(*patch_feature*, *top_k*)**:**
    COMPUTE *scores* $=$ dot product of *patch_feature* and *memory*
    APPLY *gumbel_softmax*(*scores*) TO compute probabilities
    SELECT *top_k_indices* USING probabilities
    RETRIEVE and sum top-k memory features
    Return retrieved_features
  Function gumbel_softmax(*logits*, *temperature*)**:**
    ADD Gumbel noise to *logits*
    Return softmax(*logits*/*temperature*)
  Function sample_gumbel(*shape*)**:**
    Return random Gumbel noise tensor of given *shape*

Explanation: Algorithm 4.4 defines the SpaceAwareMemory class, which manages a memory matrix with a focus on spatial relationships between image patches. The update memory function updates memory based on cosine similarity between incoming features and existing features in memory. If the similarity is below a predefined threshold, the feature is stored in the memory. The retrieve function computes a similarity score using dot products, applies Gumbel-Softmax for selection, and retrieves the top-k most relevant memory features. The Gumbel-Softmax function adds Gumbel noise to logits and applies softmax, while sample Gumbel generates random Gumbel noise.

Algorithm 4.5 Hierarchical Memory

Function __**init**__(*num_levels*, *initial_blocks*, $feature\_dim$):
  INITIALIZE *memory* AS list of SpaceAwareMemory instances, one per level
  STORE $current\_blocks = initial\_blocks$
  STORE $feature\_dim$
Function store_in_hierarchy($features$, *positions*):
  For *level* **in range** *num_levels*:
    If $level == 0$:
      CALL $memory[level].update\_memory(features, positions)$
    Else If $level == 1$:
      GROUP $features$ INTO blocks
      COMPUTE mean feature PER block
      CALL $memory[level].update\_memory(features, correspondingpositions)$
    Else If $level == 2$:
      COMPUTE max pooled feature ACROSS all features
      CALL $memory[level].update\_memory(maxpooledfeature, singleposition)$
Function retrieve_from_hierarchy($patch\_feature$, top_k):
  For *memory_instance* **in** *memory*:
    RETRIEVEfeaturesUSING $retrieve\_features(patch\_feature, top\_k)$
  Return concatenated features from all levels
Function expand_memory(*additional_blocks*):
  For *memory_instance* **in** *memory*:
    EXPAND *memory_instance* by *additional_blocks*
Function adjust_memory(*patch_locations*):
  COMPUTE $max\_loc = \max(patch\_locations)$
  If $max\_loc \geq current\_blocks$:
    COMPUTE $new\_blocks = (max\_loc - current\_blocks) + 1$
    CALL $expand\_memory(new\_blocks)$

Explanation: Algorithm 4.5 introduces the HierarchicalMemory class, which extends SpaceAwareMemory into a multi-level structure. It stores memory across different levels, allowing for different types of feature aggregation at each level. The store function stores features at different levels based on the hierarchy, adjusting how features are grouped or pooled. The retrieve hierarchy function collects features from all levels and concatenates them. The expand memory function allows for dynamic memory expansion, while the adjust memory function ensures that memory can scale according to the number of patch locations.

Algorithm 4.6 Inpainting Transformer

Class InpaintingTransformer:
  Function __**init**__($feature\_dim$, num_heads, num_layers)**:**
    INITIALIZE $linear\_proj$ as Linear layer ($input\_dim \rightarrow output\_dim = feature\_dim$)
    INITIALIZE TransformerEncoder layers
    INITIALIZE $output\_layer$ as Linear layer ($feature\_dim \rightarrow feature\_dim$)
  Function forward($neighborhood\_features$, $center\_patch$)**:**
    PROJECT $neighborhood\_features$ USING $linear\_proj$
    CONCATENATE $neighborhood\_features$ WITH $center\_patch$
    FEED combined input THROUGH $TransformerEncoder$ layers
    Return $output\_layer(combined\_input)$

The algorithm 4.6 defines the Inpainting Transformer class, which is responsible for inpainting features in the memory space. It consists of a linear projection layer to adjust the feature dimensions, followed by Transformer encoder layers for processing the features. The forward function projects the neighborhood features, concatenates them with a center patch, and feeds the combined input through the Transformer layers, returning the inpainted feature.

The algorithm 4.7 introduces the Feature Inpainting class, which utilizes the Inpainting Transformer for inpainting features within a grid of patches.The apply mask and inpaint function extracts the neighborhood features based on a center mask and inpaints the center patch using the inpaint center patch method, which uses the transformer to generate the final inpainted patch. It describes the extract patch feature with inpainting function, which processes image patches through a model and applies feature inpainting using hierarchical memory. The function works as follows: It first initializes the model and memory system.The image is then split into patches.Features are extracted for each patch.he memory is adjusted and updated with these features.For each patch location, feature inpainting is performed using the hierarchical memory system. The inpainted features are collected into a list.Finally, the function returns the inpainted patch features as a tensor.

Algorithm 4.7 Feature Inpainting

Class FeatureInpainting:
  Function __init__(transformer):
    STORE $transformer$
    DEFINE $center\_mask$ FOR selecting neighborhood features
  Function apply_mask_and_inpaint($feature\_grid$**):**
    EXTRACT $neighborhood\_features$ USING $center\_mask$
    EXTRACT $center\_patch$
    Return $inpaint\_center\_patch(neighborhood\_features, center\_patch)$
  Function inpaint_center_patch($neighborhood\_features$**,** $center\_patch$**):**
    Return $transformer.forward(neighborhood\_features, center\_patch)$
Function extract_patch_features_with_inpainting($loader$**,** $model$**,** $patch\_size$**,** $hierarchical\_memory$**):**
  SET $model$ TO evaluation mode
  INITIALIZE $patch\_features$ AS empty list
  INITIALIZE $total\_patches\_extracted = 0$
  COMPUTE num_rows, num_cols USING $calculate\_num\_cols(image\_size, patch\_size)$
  each batch $(images, \_)$ IN $loader$:
    SPLIT $images$ INTO patches USING $image\_to\_patches$
    RESHAPE patches TO appropriate shape
    EXTRACT features USING $model(patches)$
    INCREMENT $total\_patches\_extracted$ BY number of patches
    COMPUTE $positions$ AS indices of patches
    CALL $hierarchical\_memory.adjust\_memory(positions)$
    CALL $hierarchical\_memory.store\_in\_hierarchy(features, positions)$
    loc IN range(number of patches):
      CALL $hierarchical\_memory.perform\_feature\_inpainting(loc)$
      APPEND inpainted feature TO $patch\_features$
  Return all $patch\_features$ as a tensor

4.3 GENERATOR

The feature extraction step serves as the foundation for the reconstruction process, facilitated by two distinct generators: the teacher generator and the student generator. The teacher generator uses the extracted features to reconstruct the original image, combining localized information from patches to create a cohesive output that mirrors the input image while capturing its spatial relationships. In contrast, the student generator takes

an inpainted version of the image as input and strives to replicate the teacher generator's reconstruction capabilities. Through knowledge distillation, the student generator learns from the intermediate outputs of the teacher, gaining insights into the reconstruction process and applying them to refine its understanding. This enables the student generator to produce high-quality reconstructions that effectively mimic the teacher's functionality, resulting in an efficient and collaborative learning approach that enhances overall performance.

Algorithm 4.8 Generator

Class Generator:
  Function __init__(input_dim):
    Initialize layers:
      fc: Fully connected layer (4096 → 256 * 8 * 8)
      deconv0: Transpose Conv2D (256 → 128)
      bn0: BatchNorm for deconv0
      deconv1: Transpose Conv2D (128 → 64)
      bn1: BatchNorm for deconv1
      deconv2: Transpose Conv2D (64 → 32)
      bn2: BatchNorm for deconv2
      deconv3: Transpose Conv2D (32 → 1)
  Function forward(x, return_intermediate=False):
    Reshape x → Flatten → Pass through fc → Reshape to (batch, 256, 8, 8)
    x0 = Pass x through deconv0 → bn0 → ReLU
    x1 = Pass x0 through deconv1 → bn1 → ReLU
    x2 = Pass x1 through deconv2 → bn2 → ReLU
    x3 = Pass x2 through deconv3
    Return (x0, x1, x2, x3) if return_intermediate else x3

Explanation: Algorithm 4.8 defines the Generator class, which uses a fully connected layer followed by three transpose convolutional layers with batch normalization and ReLU activations, ending with a final transpose convolutional layer to generate an image from a latent input, with an option to return intermediate outputs.

Algorithm 4.9 Teacher Generator

Input: Patch features
Output: Reconstructed Images
Initialize:
  patch_cnn = Pre-trained PatchBasedCNN (for feature extraction)
  teacher_gen = Generator (for image reconstruction)
  optimizer = Adam optimizer for teacher_gen parameters, learning rate
for each epoch: **do**
    Set patch_cnn to evaluation mode
    Set teacher_gen to training mode
    Initialize total_loss to 0
    for each batch of images in train_loader: **do**
      Split images into patches of size (4, 16)
      Reshape patches to (batch_size * num_patches, 1, N, N)
      Use patch_cnn to extract features from patches
      Reshape extracted features for input into teacher_gen
      Use teacher_gen to reconstruct the original image from features
      Compute Mean Squared Error (MSE) loss between reconstructed image and original image
      Perform backpropagation:
        Zero gradients
        Compute gradients using loss.backward()
        Update teacher_gen parameters using optimizer.step()
      Add batch loss to total_loss
    end for
    Compute average loss for the epoch

Explanation: Algorithm 4.9 outlines the teacher training loop, where a pre-trained PatchBasedCNN is used for feature extraction and a Generator (teacher generator) is trained for image reconstruction. In each epoch, the patch_cnn is set to evaluation mode, and teacher_gen is set to training mode. For each batch in the training data, images are split into $N \times N$ patches and reshaped. Features are extracted from these patches using the patch_cnn, then passed to the teacher_gen to reconstruct the original image. The Mean Squared Error (MSE) loss is computed between the reconstructed and original images, followed by backpropagation to zero gradients, compute new gradients, and update the teacher_gen parameters using the Adam optimizer. The batch loss is accumulated, and the average loss is computed.

Algorithm 4.10 Student Generator

Input: Augmented features

Output: Reconstructed Images

```
Initialize:
    student_gen, optimizer
for each epoch: do
    Set student_gen to training mode
    Reset total_reconstruction_loss, total_distillation_loss
    for each batch of images in train_loader: do
        Extract patches from images and reshape them
        Use patch_cnn (no gradient) to extract features
        Perform hierarchical inpainting for features
        Stack inpainted features into a tensor
        Use teacher_gen (no gradient) to get teacher intermediate outputs
        Use student_gen to get student intermediate outputs
        Compute:
            L_s (Reconstruction Loss): MSE between final student output and original images
            L_dist (Distillation Loss): MSE between teacher and student intermediate outputs
        Perform backpropagation:
            Zero optimizer gradients
            Compute gradients and update student_gen
        Accumulate total_reconstruction_loss, total_distillation_loss
    end for
    Compute average loss
```

Explanation: Algorithm 4.10 outlines the student training loop, where the student_gen is trained for image reconstruction and distillation from the teacher. In each epoch, student_gen is set to training mode, and total_reconstruction_loss and total_distillation_loss are reset. For each batch of images, patches are extracted and reshaped, with features extracted using the pre-trained patch_cnn (without gradients). Hierarchical inpainting is then applied to the features, and inpainted features are stacked into a tensor. The teacher's intermediate outputs are obtained (without gradients), and the student's intermediate outputs are computed. The Reconstruction Loss (L_s) is calculated as the MSE between the final student output and the original image, while the Distillation Loss (L_{dist}) is computed as the MSE between

the teacher's and student's intermediate outputs. Gradients are computed using backpropagation, and the optimizer updates the student_gen parameters. Finally, the total reconstruction and distillation losses are accumulated, and the epoch averages for both losses are computed.

4.4 DISCRIMINATOR

The discriminator plays a crucial role in distinguishing between real and generated images, acting as a classifier that assesses the quality of the images produced by the teacher and student generators. It receives both the original and generated images as inputs and attempts to classify them as real or fake. By comparing generated images to real ones, the discriminator learns to identify subtle differences, providing feedback to guide the generators in improving their output. It is trained to detect anomalies and inconsistencies, ensuring that generated images become more realistic over time. In addition, the discriminator improves the system's ability to identify small irregularities, requiring the generators to produce highly detailed and coherent images. Through this adversarial process, the discriminator promotes more accurate and realistic image generation in the overall system, ultimately refining the model's capacity for anomaly detection.

The algorithm 4.11, defines a discriminator that distinguishes between real and fake images generated by a student model. The discriminator is trained using a loss function to minimize the difference between real and fake image scores. The anomaly score for an image is computed by comparing the real and generated images' scores, normalizing the difference, and applying a sigmoid function for the final result.

Algorithm 4.11 Discriminator

Input: Student Reconstructed Images

Output: Anomaly score

Initialize Discriminator model (**D**)
Set optimizer_d = Adam(discriminator.parameters(), lr=0.0001)
for each epoch: **do**
    Set **D** to training mode
    Initialize total_d_loss = 0
    for each batch (batch_idx, images) in train_loader: **do**
        Set real_labels = Tensor of ones (size=batch_size)
        Set fake_labels = Tensor of zeros (size=batch_size)
        Training on Real Images:
        Zero the gradients: optimizer_d.zero_grad()
        Compute real_scores = D(images)
        Training on Fake Images:
        Extract patches = image_to_patches(images, patch_size=(4, 4))
        Compute fake_features = patch_cnn(patches)
        Reconstruct reconstructed_images = student_gen(fake_features)
        Compute fake_scores = D(reconstructed_images)
        Discriminator Loss:
        Compute $L_{dis} = -\text{Mean}(\log(\text{real\_scores}) + \log(1 - \text{fake\_scores}))$
        Backpropagate L_{dis}: L_dis.backward()
        Update weights: optimizer_d.step()
        Accumulate total_d_loss += L_{dis}
    end for
    Compute avg_d_loss = total_d_loss / len(train_loader)
end for
Save the trained Discriminator model (**D**)
Anomaly Detection for Given Image:
Compute encoded_features = Encoder(image)
Flatten encoded_features to match generator input size
Expand using a fully connected layer (Linear layer)
Compute reconstructed_image = student_gen(encoded_features)
Compute discriminator_output = D(reconstructed_image)
Normalize discriminator_output:
score $= (\text{discriminator\_output} - \mu)/\sigma$
Apply sigmoid function: anomaly_score = expit(score)
Return anomaly_score for the image =0

The processes used to develop the anomaly detection system for radiograph images are explained by detailing the algorithms and techniques implemented to achieve accurate and effective results. This implementation chapter outlines the methodology for developing the anomaly detection system, highlighting the various approaches, including feature extraction, hierarchical memory inpainting, and adversarial frameworks, as well as the models utilized throughout the project.

CHAPTER 5

RESULTS AND ANALYSIS

This chapter should provide implementation details of your work with results and analysis.

5.1 PERFORMANCE METRICS

The performance of the model is evaluated by metrics such as Mean Squared Error(MSE) and Kullback–Leibler (KL) divergence.

Mean Squared Error(MSE) Mean Squared Error (MSE) is a measure used to evaluate the performance of a model by calculating the average squared difference between predicted and actual values. It quantifies how well the predicted values match the actual outcomes.The formula for MSE is calculated using Equation 5.1:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (5.1)$$

Kullback–Leibler (KL) divergence KL Divergence is a type of statistical distance that measures how much one probability distribution Q (the model distribution) differs from another probability distribution P (the true distribution). Specifically, in image processing it measures how much the pixel intensity distribution (or other feature distributions) of one image differs from another. A lower KL Divergence indicates that the distributions are more similar, and a higher value indicates greater differences between the imagesThe

formula for KL Divergence is given by Equation 5.2:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \qquad (5.2)$$

5.2 PERFORMANCE METRICS FOR TEACHER GENERATOR

The Teacher Generator demonstrates a significant improvement in reconstructing images, producing outputs that closely resemble the original inputs. The loss function is calculated using the formula as shown in Equation 5.3:

$$L_t = \|I - G_t(E(I))\|^2 \qquad (5.3)$$

where I is the original image, E(I) is the encoded image, and Gt is the generated image after decoding. This formula measures the difference between the original and generated images.

```
Epoch [1/1000], Loss: 0.3495
Epoch [2/1000], Loss: 0.0608
Epoch [3/1000], Loss: 0.0444
Epoch [4/1000], Loss: 0.0394
Epoch [5/1000], Loss: 0.0372
Epoch [6/1000], Loss: 0.0361
Epoch [7/1000], Loss: 0.0353
Epoch [8/1000], Loss: 0.0348
Epoch [9/1000], Loss: 0.0344
Epoch [10/1000], Loss: 0.0342
Epoch [11/1000], Loss: 0.0339
Epoch [12/1000], Loss: 0.0337
Epoch [13/1000], Loss: 0.0336
Epoch [14/1000], Loss: 0.0334
Epoch [15/1000], Loss: 0.0333
Epoch [16/1000], Loss: 0.0331
Epoch [17/1000], Loss: 0.0329
Epoch [18/1000], Loss: 0.0328
Epoch [19/1000], Loss: 0.0326
Epoch [20/1000], Loss: 0.0323
```

```
Epoch [981/1000], Loss: 0.0029
Epoch [982/1000], Loss: 0.0030
Epoch [983/1000], Loss: 0.0029
Epoch [984/1000], Loss: 0.0028
Epoch [985/1000], Loss: 0.0028
Epoch [986/1000], Loss: 0.0028
Epoch [987/1000], Loss: 0.0027
Epoch [988/1000], Loss: 0.0027
Epoch [989/1000], Loss: 0.0027
Epoch [990/1000], Loss: 0.0027
Epoch [991/1000], Loss: 0.0027
Epoch [992/1000], Loss: 0.0028
Epoch [993/1000], Loss: 0.0028
Epoch [994/1000], Loss: 0.0028
Epoch [995/1000], Loss: 0.0029
Epoch [996/1000], Loss: 0.0029
Epoch [997/1000], Loss: 0.0029
Epoch [998/1000], Loss: 0.0029
Epoch [999/1000], Loss: 0.0029
Epoch [1000/1000], Loss: 0.0029
```

Figure 5.1: Teacher Loss

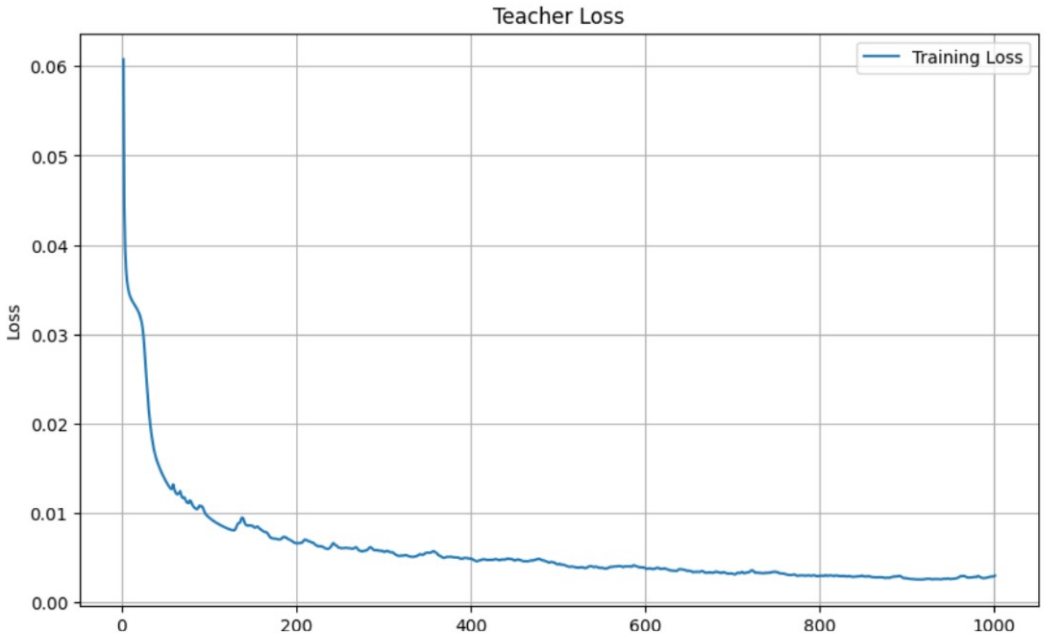

Figure 5.2: Teacher Loss Graph

The loss graph in Figure 5.1 and 5.2 shows that the loss decreases steadily during training, indicating that the generator is learning to improve its image reconstruction over time.

```
Image 1, KL Divergence Score: 0.0095
Image 2, KL Divergence Score: 0.0074
Image 3, KL Divergence Score: 0.0088
Image 4, KL Divergence Score: 0.0081
Image 5, KL Divergence Score: 0.0059
```

Figure 5.3: Kullback-Leibler (KL) divergence

Additionally, KL divergence, shown in Figure 5.3, is calculated to compare the distribution of the generated images with the original images. This measure helps assess how closely the generated images match the true images in terms of overall patterns. A lower KL divergence indicates that the generated images are more similar to the original data in their distribution.

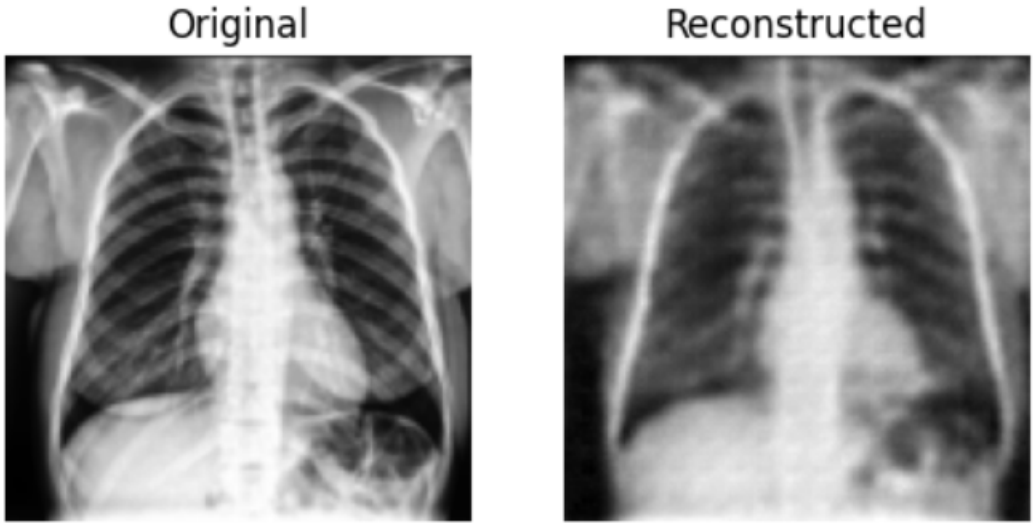

Figure 5.4: Teacher Reconstructed Image

The reconstructed images in Figure 5.4 demonstrate how well the generator can recreate the original images. The results are clear, with minimal distortion and a good match to the original images. The teacher generator strikes a balance between reducing loss and maintaining image quality. While there may be slight differences in images due to their complexity, the generator consistently performs well, making it a valuable part of the overall anomaly detection system.

5.3 VISUALISATION OF INPAINTING PATCHES

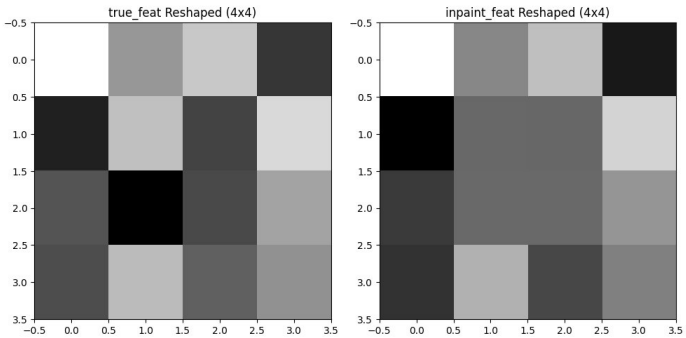

Figure 5.5: 4x4 Patches

Figure 5.5 shows a comparison between the original patches and the inpainted patches. The inpainted patches are displayed alongside their corresponding original patches, highlighting how closely the inpainted patches resemble the originals, with minimal distortion and a strong match in terms of texture and features.

```
true_feature [[ 0.02402875  0.00491205  0.01381195 -0.01281444]
 [-0.01673663  0.01248153 -0.01052233  0.01691908]
 [-0.00741485 -0.0228887  -0.00948721  0.00712124]
 [-0.00866988  0.01148966 -0.00528781  0.00380326]]
inpainted_feature [[ 2.4028752e-02  4.9120514e-03  1.3811950e-02 -1.2814444e-02]
 [-1.6736627e-02 -8.1696344e-05 -2.5298496e-04  1.6919078e-02]
 [-7.4148462e-03  2.0976333e-04 -1.5155703e-05  7.1212426e-03]
 [-8.6698802e-03  1.1489660e-02 -5.2878149e-03  3.8032578e-03]]
```

Figure 5.6: Patch Features

Figure 5.6 shows the tensor values for both the original and inpainted patches. The tensor values provide a numerical representation of the feature characteristics of each patch, offering insight into the consistency of the features before and after the inpainting process. By comparing the tensor values, we can observe how closely the inpainted patches match the original patches in terms of their underlying features. The minimal differences in the tensor values further confirm the effectiveness of the inpainting process in reconstructing the missing information while preserving the overall feature structure

5.4 PERFORMANCE METRICS FOR STUDENT GENERATOR

Figure 5.7 and 5.8 shows the reconstruction loss of the generator. The loss for the student generator is calculated using the following formula as shown in Equation 5.4:

$$L_s = \|I - G_s(E(I))\|^2 \qquad (5.4)$$

where I is the original image, E(I) represents the encoded image, and G_s is the reconstructed image from the Student Generator.

```
Epoch [0]: L_s (Reconstruction): 0.0317, L_dist: 0.6160
Epoch [1]: L_s (Reconstruction): 0.0264, L_dist: 0.6065
Epoch [2]: L_s (Reconstruction): 0.0212, L_dist: 0.6011
Epoch [3]: L_s (Reconstruction): 0.0184, L_dist: 0.5989
Epoch [4]: L_s (Reconstruction): 0.0167, L_dist: 0.5967
Epoch [5]: L_s (Reconstruction): 0.0154, L_dist: 0.5947
Epoch [6]: L_s (Reconstruction): 0.0144, L_dist: 0.5928
Epoch [7]: L_s (Reconstruction): 0.0136, L_dist: 0.5913
Epoch [8]: L_s (Reconstruction): 0.0130, L_dist: 0.5900
Epoch [9]: L_s (Reconstruction): 0.0124, L_dist: 0.5890
Epoch [10]: L_s (Reconstruction): 0.0119, L_dist: 0.5879
Epoch [11]: L_s (Reconstruction): 0.0115, L_dist: 0.5869
Epoch [12]: L_s (Reconstruction): 0.0111, L_dist: 0.5860
Epoch [13]: L_s (Reconstruction): 0.0108, L_dist: 0.5853
Epoch [14]: L_s (Reconstruction): 0.0105, L_dist: 0.5846
Epoch [15]: L_s (Reconstruction): 0.0102, L_dist: 0.5839
Epoch [16]: L_s (Reconstruction): 0.0101, L_dist: 0.5834
Epoch [17]: L_s (Reconstruction): 0.0102, L_dist: 0.5831
Epoch [18]: L_s (Reconstruction): 0.0100, L_dist: 0.5827
Epoch [19]: L_s (Reconstruction): 0.0099, L_dist: 0.5821
Epoch [20]: L_s (Reconstruction): 0.0099, L_dist: 0.5815
```

```
Epoch [231]: L_s (Reconstruction): 0.0033, L_dist: 0.4830
Epoch [232]: L_s (Reconstruction): 0.0033, L_dist: 0.4824
Epoch [233]: L_s (Reconstruction): 0.0033, L_dist: 0.4818
Epoch [234]: L_s (Reconstruction): 0.0033, L_dist: 0.4815
Epoch [235]: L_s (Reconstruction): 0.0033, L_dist: 0.4815
Epoch [236]: L_s (Reconstruction): 0.0032, L_dist: 0.4807
Epoch [237]: L_s (Reconstruction): 0.0033, L_dist: 0.4805
Epoch [238]: L_s (Reconstruction): 0.0033, L_dist: 0.4805
Epoch [239]: L_s (Reconstruction): 0.0033, L_dist: 0.4800
Epoch [240]: L_s (Reconstruction): 0.0032, L_dist: 0.4796
Epoch [241]: L_s (Reconstruction): 0.0032, L_dist: 0.4795
Epoch [242]: L_s (Reconstruction): 0.0033, L_dist: 0.4789
Epoch [243]: L_s (Reconstruction): 0.0033, L_dist: 0.4785
Epoch [244]: L_s (Reconstruction): 0.0033, L_dist: 0.4780
Epoch [245]: L_s (Reconstruction): 0.0033, L_dist: 0.4776
Epoch [246]: L_s (Reconstruction): 0.0032, L_dist: 0.4774
Epoch [247]: L_s (Reconstruction): 0.0032, L_dist: 0.4768
Epoch [248]: L_s (Reconstruction): 0.0032, L_dist: 0.4767
Epoch [249]: L_s (Reconstruction): 0.0032, L_dist: 0.4761
Epoch [250]: L_s (Reconstruction): 0.0032, L_dist: 0.4758
```

Figure 5.7: Student Loss

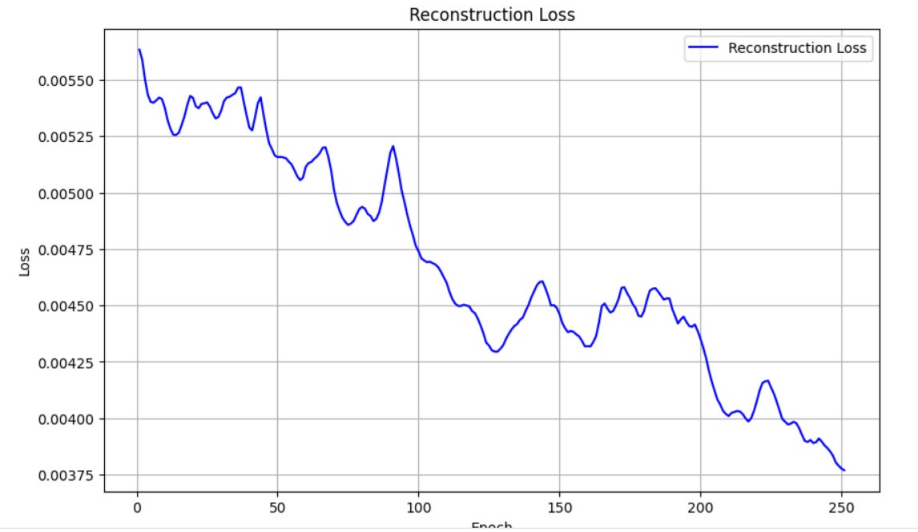

Figure 5.8: Student Loss Graph

The loss graph for L_s indicates a steady decline throughout the training process, demonstrating the generator's ability to minimize the difference between the original and reconstructed images, thereby improving reconstruction quality over time.

The distillation loss L_dist is computed as shown in Equation 5.5,

$$L_{\text{dist}} = \sum_{i=1}^{l} \left(z_t^i - z_s^i\right)^2 \qquad (5.5)$$

Here, z_t and z_s denote the features from the teacher and student generators, respectively.

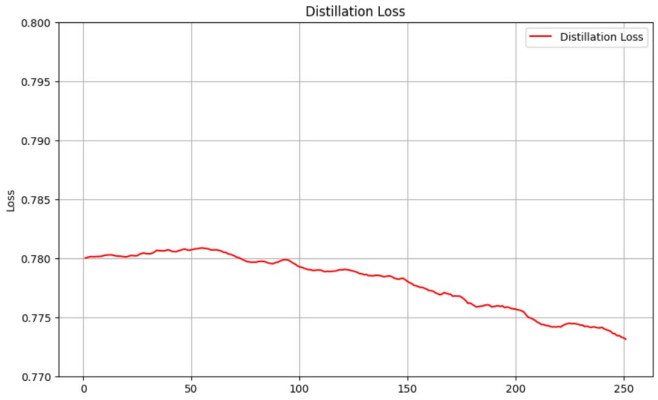

Figure 5.9: Distillation Loss Graph

Figure 5.9 shows the loss graph for L_dist how the student generator progressively aligns its feature representations with those of the teacher generator. This ensures the consistency of feature distribution between the two generators, further enhancing the model's overall performance.

```
Image 1, Student-Original KL Divergence Score: 0.4290
Image 2, Student-Original KL Divergence Score: 0.2276
Image 3, Student-Original KL Divergence Score: 0.1855
Image 4, Student-Original KL Divergence Score: 0.2067
Image 5, Student-Original KL Divergence Score: 0.2044
```

Figure 5.10: Kullback-Leibler (KL) divergence

Additionally, KL divergence, shown in Figure 5.10, is calculated to compare the distribution of the student-generated images with the original

images. This metric evaluates how well the reconstructed images align with the true data distribution. A consistent reduction in KL divergence during training indicates that the student generator effectively learns to produce outputs that closely mimic the original image distribution, even when working with augmented features.

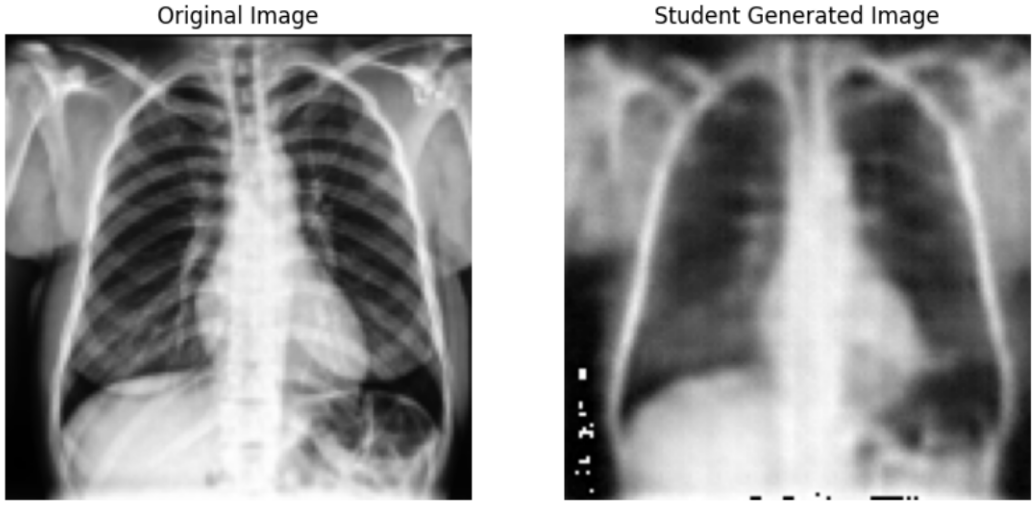

Figure 5.11: Student Reconstructed Image

Figure 5.11 shows the reconstructed images created by the student generator alongside the original images. The reconstructed images are very similar to the originals, even though the generator uses augmented features during the process.The images keep important details and textures intact, showing that the inpainting block and feature substitution work well in maintaining the quality of the images. While there might be small differences in some areas, the overall quality is good, proving that the student generator can produce realistic and accurate results.This demonstrates the student generator's ability to handle augmented features effectively, making it a key part of the anomaly detection system.

5.5 PERFORMANCE METRICS FOR DISCRIMINATOR

Figure 5.12 shows the performance of the discriminator, which helps distinguish between real and generated images. The loss function used for discrimination is defined as Equation 5.6:

$$L_{\text{dis}} = \log(D(I)) + \log(1 - D(G_s(E(I)))) \tag{5.6}$$

This formula helps assess how well the discriminator can differentiate between real images from the dataset and the images generated by the student generator.

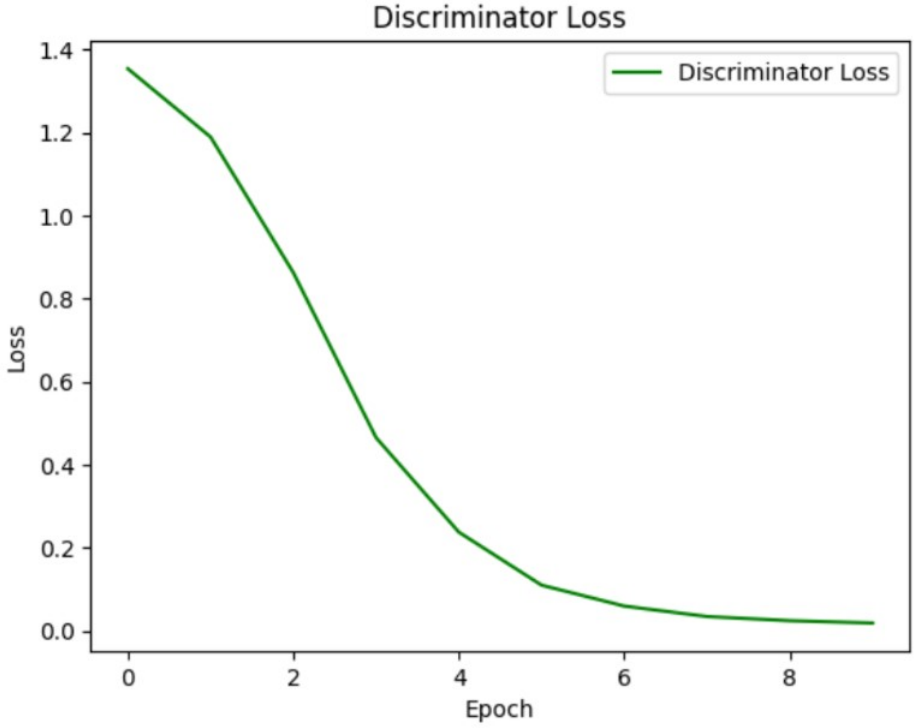

Figure 5.12: Discriminator Loss Graph

As the loss decreases, the discriminator improves in identifying real and fake images, enhancing the model's anomaly detection capabilities. A lower discrimination loss indicates that the discriminator can accurately classify real and generated images, thus contributing to the overall success of the anomaly detection process.

5.6 ANOMALY SCORE

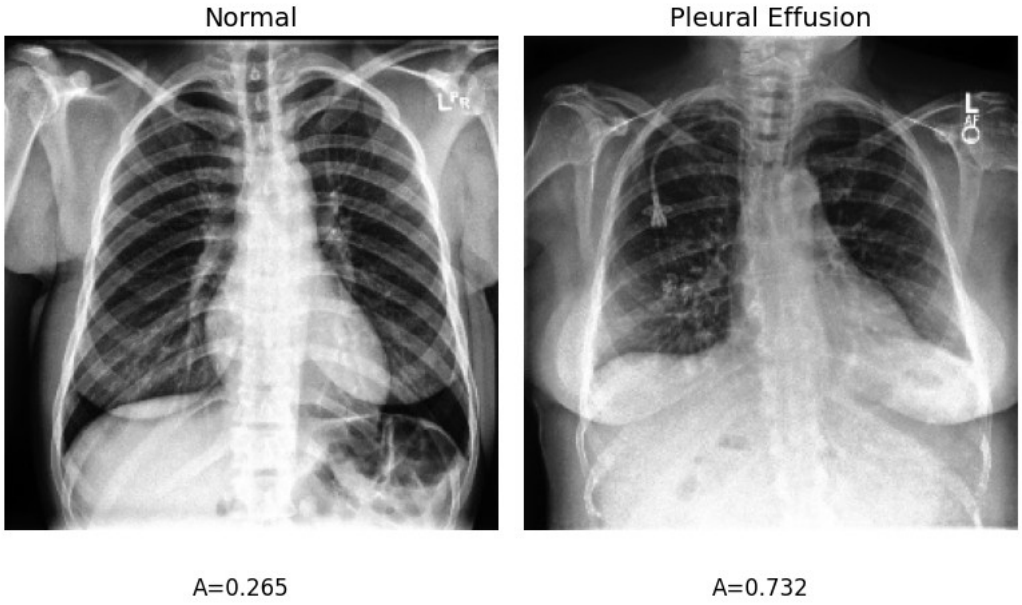

Figure 5.13: Anomaly Score

Figure 5.13 presents the anomaly scores for the images, which help assess how well the model can detect anomalies. The anomaly score (A) is calculated using the formula in Equation 5.7 ,

$$A = \phi\left(\frac{D(G_s(E(I))) - \mu}{\sigma}\right) \qquad (5.7)$$

where ϕ is the Sigmoid function, $D(G_s(E(I)))$ is the discriminator's output for the generated image, and μ and σ represent the mean and standard deviation of anomaly scores calculated on all training samples. Higher anomaly scores indicate that the image significantly deviates from normal patterns, suggesting a potential anomaly. Lower anomaly scores indicate that the image closely matches the normal data, signaling no anomaly. These scores are crucial for the anomaly detection process, as they provide a quantitative measure for distinguishing between normal and anomalous images.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In conclusion, the anomaly detection system developed for radiograph images leverages innovative techniques in feature extraction, hierarchical memory inpainting, and adversarial learning to enhance the detection of anomalies with precision and reliability. By utilizing patch-based feature extraction and inpainting methodologies, the system effectively captures spatial details and restores missing or corrupted features, ensuring robustness in anomaly identification. The integration of hierarchical memory and transformer-based inpainting allows for the preservation of spatial relationships and feature consistency, improving the overall model's capability to discern subtle irregularities in radiographs.

The adversarial framework, comprising teacher-student generators and a discriminator, further refines the model's performance by distinguishing between normal and anomalous patterns, enabling a comprehensive evaluation of anomalies. This approach contributes to an accurate and efficient anomaly detection pipeline, demonstrating the potential of combining advanced memory mechanisms and adversarial learning for medical image analysis. As a result, this system provides a reliable tool for assisting radiologists in identifying anomalies, ultimately supporting improved diagnostic outcomes.

6.2 FUTURE WORK

The future work of the anomaly detection system lies in enhancing feature inpainting and expanding the dataset for better performance while addressing resource utilization challenges. Refining the feature inpainting technique, such as integrating multi-scale inpainting and attention mechanisms, could improve anomaly localization, especially at pixel-level precision. However, the increased complexity of inpainting and larger datasets may lead to higher resource consumption, making it crucial to explore optimized approaches that reduce computational costs while maintaining high performance. This balance between enhanced model capabilities and efficient resource utilization will be key to the system's success in real-world deployment.

REFERENCES

[1] J. Wolleb, F. Bieder, R. Sandkühler, and P. C. Cattin. Diffusion models for medical anomaly detection. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 35–45. Springer, 2022.

[2] Muzaffer Özbey, Onat Dalmaz, Salman U. H. Dar, Hasan A. Bedel, Şaban Özturk, Alper Güngör, and Tolga Çukur. Unsupervised medical image translation with adversarial diffusion models. *IEEE Transactions on Medical Imaging*, 42(12):3524–3539, 2023.

[3] T Schlegl, P Seeböck, S. M Waldstein, G Langs, and U Schmidt-Erfurth. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, vol.54:30–44, 2019.

[4] B Nguyen, A Feldman, S Bethapudi, A Jennings, and C. G Willcocks. Unsupervised region-based anomaly detection in brain mri with adversarial image inpainting. In *Proceedings of the IEEE 18th International Symposium on Biomedical Imaging*, pages 1127–1131, 2021.

[5] J Guo, S Lu, L Jia, W Zhang, and H Li. Encoder-decoder contrast for unsupervised anomaly detection in medical images. *IEEE Transactions on Medical Imaging*, 43(3):1102–1112, March 2024.

[6] Tiange Xiang, Yixiao Zhang, Yongyi Lu, Alan L Yuille, Chaoyi Zhang, Weidong Cai, and Zongwei Zhou. Squid: Deep feature in-painting for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 23890–23901, 2023.

[7] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 590–597, 2019.

[8] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages pp.1705–1714, 2019.

[9] Kang Zhou, Jing Li, Yuting Xiao, Jianlong Yang, Jun Cheng, Wen Liu, Weixin Luo, Jiang Liu, and Shenghua Gao. Memorizing structure-texture

correspondence for image anomaly detection. *IEEE transactions on neural networks and learning systems*, 33(6):2335–2349, 2021.

[10] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[11] V Zavrtanik, M Kristan, and D Skočaj. Reconstruction by inpainting for visual anomaly detection. *Pattern Recognition*, 112:Art. no. 107706, 2021.

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. *Biometrika*, 71(599-607):6, 1986.

[13] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4183–4192, 2020.

[14] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.