A project report on

ANOMALY DETECTION USING TRANSGAN

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

NAYAN KHEMKA (20BCE1884)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April,2024

ANOMALY DETECTION USING TRANSGAN

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

NAYAN KHEMKA (20BCE1884)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2024

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "**Anomaly Detection using TransGAN**" submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of bona-fide work carried out by me under the supervision of **Dr. Premalatha M**.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"Anomaly Detection using TransGAN"** is prepared and submitted by **Nayan Khemka** (**20BCE1884**) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering programme** is a bona-fide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr. Premalatha M.

Date:

Signature of the Internal Examiner

Name:

Date:

Signature of the External Examiner

Name:

Date:

Approved by the Head of Department,
B.Tech. CSE

Name: Dr. Nithyanandam P

Date:

(Seal of SCOPE)

ABSTRACT

There is a huge number of industries increasingly depending upon systems that automatically deal with image data. This calls for the need of reliable and effective methods to deal with the data so as the system is not hindered by some anomalies in the dataset. This research paper investigates the efficacy of TransGAN that combines the power of vision transformers and Generative Adversarial Network (GAN) for anomaly detection in images. This paper makes use of some benchmark datasets such as MVTEC-AD and CIFAR-10 image for the same. Latent space mapping for the images and reconstruction differences are taken for anomaly detection in MVTecAD. And for CIFAR-10, two discriminators are used for the training the model to address the class imbalance along with weighted combination of several loss functions described in the paper. An accuracy of 0.9 for normal images and 0.6 for anomalous images in MVTEC-AD hazel-nut class was obtained through this methodology of latent space representation. The model yields an AUROC score of 0.83 on the testing set of the CIFAR-10 images. This research aims to contribute to the further enhancement in the field of anomaly detection in various domains such as medical imaging, surveillance and others.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to **Dr. Premalatha M**, **Associate Professor**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of **Deep Learning**.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, B.Tech. CSE and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: **Nayan Khemka**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

GAN	Generative Adversarial Net
ViT	Vision transformer
SAGAN	Self-Attention Generative Adversarial Net
CT-D2GAN	Convolutional Transformer-based Dual Discriminator GANs
MLP	Multiple Layer Perceptron
GELU	Gaussian Error Linear Unit
ReLU	Rectified Linear Unit
FID	Fréchet Inception Distance
AE	Autoencoders
VAE	Variational Auto Encoders
AUROC	Area Under the Receiver Operating Characteristic Curve
RNN	Recurrent Neural Network
OOD	Out of Distribution

CHAPTER – 1

INTRODUCTION

1.1 A BRIEF OVERVIEW

An increasing number of industries are depending on automated visual inspection systems which calls for reliable and effective anomaly detection methods for image data. In recent years, anomaly detection in image data has gained significant attention due to its wide-range applications in carious domains such as manufacturing, medical imaging and surveillance [1][2]. Conventional approaches hardly grasp the complexity of intricate image dataset since they rely on manually created features or statistical models. Using architectures is hence essential for obtaining superior performance in the field of anomaly detection in images a variety of.

In this study, two different approaches both focused on TransGAN [3]. Unlike traditional GANs [4], which primarily rely on convolutional layers, TransGAN utilizes TransformerEncoder layers to capture spatial dependencies. This enhances the quality of images that are produced and enable usage across several applications. Motivated by the iterative latent space mapping method of Ano-GAN [5] we train TransGAN for anomaly detection for MVTEC-AD data set [6]. Our goal is to detect deviations from the learned normal distributions and accurately identify anomalies. In contrast, we take a new approach for imbalanced data sets such as CIFAR-10 [7]. In this approach, we use ResNet50 [8] to extract features from real image to map it to the noise input of the generator. We also make use of two discriminators one for normal data and the other for anomalous data to address the imbalance in data. Through extensive experimentation and evaluation, we have tried to present the effectiveness of the framework in identifying anomalies in diverse image datasets, thus giving a direction to further research on the anomaly detection applications in the real-world.

1.2 PURPOSE AND MOTIVATION

This project revolves around progressing anomaly detection in image data using TransGAN architecture. With the ever-growing increase dependency on automated visual inspection systems across various industries, there’s a necessary need for robust anomaly detection methods fine-tuned for visual data. Anomaly detection in image data

has gained significant attention due to the widespread applications in manufacturing, medical imaging, surveillance and various other domains. Hence there's a compelling motivation to explore advanced architectures like that of TransGAN that makes use of transformers in the Generative Adversarial Networks to achieve superior performance in anomaly detection. By extending the application of TransGAN to anomaly detection, this project aims to further increase the usage and research on the topic and lead to improvement in more efficient anomaly detection capabilities across diverse image datasets.

1.3 OVERVIEW OF ARCHITECTURE

This project uses two main approaches utilizing TransGAN architecture for anomaly detection. Firstly, TransGAN is trained for anomaly detection on the MVTEC-AD dataset followed by leveraging of the iterative latent space mapping method of Ano-GAN. This approach focuses on detecting deviations from learned normal distributions and accurately identifying anomalies. In second approach for CIFAR-10 dataset, TransGAN is integrated with ResNet50 model to extract features from real images and to map them to the noise input of the generator. Additionally, two discriminators are used—one for normal data and one for anomalous data—to address data imbalance effectively. Through these approaches, this project aims to enhance the effectiveness and applicability of anomaly detection systems in diverse image datasets.

1.4 CURRENT CHALLENGES

Anomaly detection in visual data faces several challenges including complexity and diversity of the datasets. Addressing imbalanced data distribution, various anomaly types, and achieving real time performance is very important for anomaly detection. This field is advancing with a rapid pace and hence innovative approaches and advanced architectures need to be explored more to address the challenges this domain poses to provide more accurate and efficient anomaly detection capabilities.

Some challenges include:

- Dealing with occlusions, noise, and other forms of disruptions in the dataset that can affect anomalies.
- Scaling anomaly detection methods to large-scale datasets while maintaining efficiency and accuracy.

- Interpreting and explaining the decisions made by anomaly detection models to ensure reliability.
- Integrating and adapting anomaly detection techniques to new domains and application areas, each with its unique challenges and requirements.
- Addressing ethical and privacy concerns related to the collection and use of image data for anomaly detection purposes.

1.5 PROJECT STATEMENT

This project is aimed at making use of advancements and approaches for anomaly detection using TransGAN architecture. By leveraging advanced methods while addressing current challenges, goal of this project is to enhance effectiveness and applicability of the anomaly detection system in real-world use-cases. Through experimentation of different approaches, this project demonstrates the capability of TransGAN architecture for anomaly detection across different image datasets. Ultimately, this project aims to contribute to the increasing advancements in anomaly detection applications and provide practical solutions for real-world anomaly detection.

1.6 OBJECTIVES

The objectives of this study are very clearly to investigate, analyze and address key facets of anomaly detection in visual data using the TransGAN architecture. Through multiple approaches explored in the project, a comprehensive understanding of anomaly detection and its implications has been aimed to achieved. Specifically, the objectives that guide this research and constitute the criteria for successful completion of this project include:

- Train TransGAN for MVTecAD and CIFAR-10 datasets and assess its performance across anomaly types and severities.
- Explore the latent space representation of the Generator to detect anomalies.
- Integrating ResNet50 with the TransGAN and making use of two discriminators for addressing the imbalanced dataset.
- Conduct extensive experimentation and evaluation to assess the effectiveness and robustness of both the approaches used in the project.
- Integrating and adapting anomaly detection techniques to new domains and application areas, each with its unique challenges and requirements.

CHAPTER – 2

BACKGROUND

2.1 INTRODUCTION

The backdrop of this research is anchored in the dynamic and advancing domain of anomaly detection in visual data. The advancements are being driven by the growing demands in various industrial and domains. Recent studies have proposed several approaches using different technologies for the same. But the most promising among these seem to be the use of GANs. With the way GANs are advancing specifically, when combined with the power of transformers, several authors have demonstrated the potential of these technologies in various domains and applications. However, amidst these strides of advancements, there needs a critical need to enhance the accuracy of identifying different anomalies and address the limitations of the existing solutions. This research seeks to bridge the gap by implementing TransGAN in different approaches with different datasets to encourage further research and improvement in these approaches.

2.2 LITERATURE REVIEW

In many different fields of study and applications, deep learning-based anomaly detection has been thoroughly investigated. To give a systematic and thorough summary of the research in this area a number of surveys have been carried out. With a focus on discussing the advantages disadvantages and computational complexity of current methods these surveys attempt to classify them according to their fundamental presumptions and methods. In particular some surveys concentrate on posthoc anomaly detection methods to strengthen deep learning against adversarial and outlier distribution instances. These surveys address the relative benefits and drawbacks of various techniques and offer a taxonomy of them [5] [8].

Currently the state of art techniques used in anomaly detection are ADCNN [9] and Deep-Anomaly [10]. But the absence of labelled data turns out to be the bottleneck for these approaches. On one hand, the uneven distribution makes it difficult to gather abnormal images of both typical and anomalous data and on the other is that anomalous data cannot be clearly defined [11]. A number of learning-based methodologies have

been put forth to address these challenges [12]. These approaches discover the distribution of data and feature distribution of the normal data in terms of a predefined threshold is thought to include unusual objects.

It is suggested to use adversarial learned one-class classifiers for novelty detection when the novelty class data is missing. Using MNIST, CIFAR datasets the suggested method outperforms baseline and state-of-the-art approaches in anomaly and outlier detection in images and videos. [13] Attention-driven longrange dependency modeling for image generation tasks is made possible by Self-Attention Generative Adversarial Networks (SAGAN) [14]. SAGAN draws details using cues from all feature locations in contrast to conventional convolutional GANs that produce high-resolution details based on spatially localized points. It also guarantees that the images far-off regions retain the consistency of its highly detailed features. Rather than using local regions with fixed shapes the generator in SAGAN makes use of neighborhoods that correspond to object shapes. Furthermore, the GAN generator is subjected to spectral normalization which enhances the training dynamics. SAGAN attains cutting-edge outcomes on the ImageNet dataset exhibiting an increased Inception score and a decreased Frechet Inception distance. This model has been modified in various work to perform anomaly detection because of how well it generates faked data.

A novel approach has been proposed by Shangguan et al. for semi-supervised anomaly detection based on deep generative models with Transformers [15]. Their model uses generative adversarial networks (GAN) and autoencoders (AE) to distinguish between abnormal and normal images. While the discriminator in their model also relies on a Transformer to discern between generated and training set images the generator in their model extracts latent representations using an encoder-decoder structure based on a Transformer. By keeping the difference between the latent and original image spaces as small as possible the model is able to determine the normal distribution. Their approaches' effectiveness is demonstrated by experimental results on benchmark image datasets. Furthermore, Ruff and colleagues, additionally provide Deep SAD a semi-supervised anomaly detection technique that makes use of labeled normal and anomalous samples [16]. Their method is predicated on an information-theoretic framework wherein normal data is expected to have a lower entropy of the latent distribution than anomalous data. Even with a limited amount of labeled data

research on a wide range of datasets demonstrates that Deep SAD performs better than other shallow hybrid and deep schemes.

Multiple papers have suggested to use Transformer-based architectures and generative adversarial networks (GAN) for anomaly detection in images. The goal of these techniques is to separate outliers or unusual images from regular data. While local details are captured by GANs and normal from abnormal data is distinguished the use of Transformers enables the extraction of global semantic information. It has been demonstrated that autoencoders and GANs using Transformers as encoders and transposed convolutions as decoders can successfully distinguish anomalous from normal images [17]. In addition to industrial image anomaly detection performance has been enhanced by skip-connections and attention feature fusion based on encoder-decoder structures [15]. Convolutional Transformer-based Dual Discriminator GANs (CT-D2GAN) have also demonstrated encouraging results in unsupervised video anomaly detection [18] [19]. This is achieved by capturing typical spatial-temporal patterns and taking local consistency and global coherence of temporal dynamics into account.

2.3 COMPARATIVE ANALYSIS

S.No	Paper Title (APA Format)	Summary	Algorithms Used	Pros / Cons
[5]	Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U.M., & Langs, G. (2017). Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery.	The paper proposes AnoGAN, a deep convolutional generative adversarial network, for anomaly detection using unsupervised learning in images.	Novel anomaly scoring scheme based on mapping from image space to latent space	Pros: Unsupervised learning to identify anomalies in imaging data as candidates for markers. Cons: Is not very accurate about predicting the anomalous images.

	Information Processing in Medical Imaging.			
[8]	He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.	The paper trains deep neural networks using an approach called residual learning framework. The authors provide substantial evidence showing that these residual networks are easier to optimize and can gain accuracy from increased depth.	Residual Learning	Pros: Residual networks are easier to optimize and can gain accuracy from increased depth Cons: Computational costs.
[9]	Kwon, D., Natarajan, K., Suh, S.C., Kim, H., & Kim, J. (2018). An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks. 2018 IEEE 38th International Conference on Distributed	This paper evaluates different deep neural network models for network anomaly detection. Deeper CNN structures do not improve performance to other models.	Fully Connected Network (FCN), Variational Auto Encoder (VAE), Sequence to sequence with long-short term memory (Seq2Seq-LSTM),	Pros: Clear understanding of existing anomaly detection models. Cons: No new techniques explored here and no conclusive algorithm declared.

	Computing Systems (ICDCS), 1595-1598.		Convolutional Neural Networks (CNN)	
[10]	Sabokrou, M., Fayyaz, M., Fathy, M., Moayed, Z., & Klette, R. (2016). Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. Comput. Vis. Image Underst., 172, 88-97.	This paper suggests an efficient method for detecting and localizing anomalies using temporal data and FCNs. This architecture proposed in the paper achieves high performance in terms of computational speed and accuracy.	Fully Convolutional Neural Networks Cascaded detection algorithm	Pros: Efficient solution for detection and localization of anomalies.
[11]	Pang, G., Shen, C., Cao, L., & Hengel, A.V. (2020). Deep Learning for Anomaly Detection. ACM Computing Surveys (CSUR), 54, 1 - 38.	This paper gives a comprehensive study of deep anomaly detection algorithms and presents a taxonomy of the methods into 3 high level and 11 fine-grained categories. It discusses the intuition behind the different approaches and the challenges in the deep anomaly detection.	RDA, AnoGAN, Fast AnoGAN, EBGAN, ALAD, GANomaly, REPEN, E3Outlier, LSA, Deep SVDD, Deep SAD	Pros: Provides a taxonomy for different methods into categories. Covers wide range of algorithms making it easier for further research. Cons: The learned feature representations can be biased. The objective

				function of data reconstruction is designed for data compression and not anomaly detection.
[12]	Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., & Batra, D. (2016). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. International Journal of Computer Vision, 128, 336 - 359.	This paper presents a novel technique called Grad-CAM for producing visual explanations for the decisions made by a CNN based model. It uses the gradients of a target concept flowing into the final convolutional layer to produce localization map highlighting important regions in the image.	Gradient weighted Class Activation Mapping (novel proposed)	Pros: It makes CNN models more transparent and explainable providing justification for the decisions made by the model. It is applicable for a wide variety of models. It outperformed existing solutions in terms of interpretability and faithfulness. It identifies biases in the datasets. Cons: No clear implementation with multimodal inputs. The counterfactual explanations have limited power.

[13]	Chalapathy, R., & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. ArXiv, abs/1901.03407.	It provides a comprehensive overview of the research methods in deep-learning based anomaly detection across different industries and domains. It categorizes the current anomaly detection methods based on the underlying assumptions and usage. It discusses variations, advantages and limitations of the methods and outlines open research issues in adopting these detection methodologies.	CNN, LSTM, AE, SVM, SCDD, GAN, SPN, etc	Pros: Provides a structured and contextual understanding of different approaches. Cons: No specific implementation details are provided. Drawbacks of adopting the techniques in real-world scenarios are not clearly mentioned.
[14]	Bulusu, S., Kailkhura, B., Li, B., Varshney, P.K., & Song, D.X. (2020). Anomalous Example Detection in Deep Learning: A Survey. IEEE Access, 8, 132330-132347.	This survey reviews anomaly detection techniques in deep learning applications. It provides a better understanding of the techniques and highlight unsolved research challenges in applying these techniques in DL systems.	Out-of-distribution (OOD) and adversarial algorithms in DL systems. Deep autoencoders combined with CNNs, and RNNs	Pros: The paper highlights their relative strengths and weaknesses. It also discusses the application of anomaly detection techniques in various domains, such as Android malware detection, fake news detection, and intrusion detection. Cons: The paper acknowledges that current anomaly

				detection techniques may not be effective against complex and correlated anomalies.
[15]	Sabokrou, M., Khalooei, M., Fathy, M., & Adeli, E. (2018). Adversarially learned one-class classifier for novelty detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3379-3388)	This paper proposes an end-to-end architecture for one-class classification and novelty detection in visual media. It proposes a model with two deep learning networks – one Reconstructor and one Discriminator which are trained adversarially. The proposed approach achieves state of the art performance in anomaly detection in videos and images without any samples from novelty class during training.	Two deep networks in supervised and semi-supervised setting.	Pros: The presented system is capable of detecting samples not belonging to target class. Cons: Computationally expensive and requires large amount of training data to achieve optimal performance.
[16]	Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019, May). Self-attention generative adversarial networks. In	This paper proposes SAGAN for image generation tasks, allowing attention-driven, long-range dependency modeling. It takes cues from all feature locations and ensures consistency. Spectral Normalization is	SAGAN (proposed novel approach) spectral normalizati on, two time-scale update rule.	Pros: Allows attention-driven long-range dependency for image generation. Generates high resolution pictures.

	International conference on machine learning (pp. 7354-7363). PMLR.	used in generator for higher quality images.		
[17]	Shangguan, W., Fan, W., & Chen, Z. (2022, March). Semi-supervised anomaly detection based on deep generative models with transformer. In Proceedings of the 2022 6th International Conference on Innovation in Artificial Intelligence (pp. 172-177).	The paper presents a semi-supervised anomaly detection techniques based on deep generative models with transformers for classifying abnormal and normal images. It combines autoencoders and GANs and extracts latent representations using encoder-decoder structured generator while discriminator uses transformer to distinguish the images. The model minimizes the gap between original image space and latent space during training.	Autoencoder, GAN, Transformers	Pros: Combines strengths of AE, GANs and transformers and is able to capture long-range dependencies. It is versatile as it can deal with both labelled and un-labelled data. Cons: The performance depends highly on quality of the data used for training. The computational complexity is a strain on the resources and time of the system.
[18]	Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K. R., & Kloft, M. (2019). Deep	The paper proposes a Deep SAD, an end-to-end semi-supervised anomaly detection algorithm. It uses information-theoretic framework which is based on the assumption that	Novel deep methodology for general semi-supervised anomaly	Pros: Promising results on large and complex datasets. Can utilize labeled samples of both data. Cons: Complex and resource exhausting.

	semi-supervised anomaly detection. arXiv preprint arXiv:1906.02694.	entropy of latent distribution of normal data is much less than that of anomalous images. Extensive experimentation with different datasets demonstrates the Deep SAD performs better than hybrid and deep competitors.	detection.	
[19]	Yang, C., Lan, S., Huang, W., Wang, W., Liu, G., Yang, H., ... & Li, P. (2022, September). A transformer-based gan for anomaly detection. In International Conference on Artificial Neural Networks (pp. 345-357). Cham: Springer Nature Switzerland.	The paper proposes a Transformer-based architecture for anomaly detection in images. The method combines the advantages of self-attention mechanism and modified skip-connections to capture both local and global information. It outperforms CNN based models.	Transforme r based models with self-attention mechanism and modified skip connections .	Pros: Captures both local and global information. Mitigates limited repetitive field issue. Cons: May not always converge on certain datasets due to instability of GANs.

Table 1: Comparative analysis of existing solutions

2.4 RESEARCH GAP

Talking about the anomaly detection, especially when we discuss the improvements and changes in the integration of technologies such as ADCNN, Deep-Anomaly, SAGAN, and Transformer-based models, we need to properly assess the

current limitations and potential areas for further development. With the existing solutions, the following gaps and opportunities for advancement have been identified:

1. Scarcity of Labeled Data: Anomaly detection models highly depends on the availability of the data that is labeled, especially the ones that use supervised or semi-supervised learning. Unsupervised approaches can remove this dependency on extensively labeled datasets and hence represent a significant opportunity.

2. Generalization across diverse usage: Anomaly detection models are usually developed and experimented for specific domains. Hence models that can deal with diverse data in different domains in various settings, are still needed.

3. Real-time Detection: The computational cost and complexity of recent GAN and transformer-based models are very high and hence limit real-time detection. Further research to balance the computational cost with detection efficiency is very important for several applications.

4. Interpretability and Explainability: The advanced deep learning models sometimes fail due to lack of interpretability. The reason behind why a model identifies certain data points as anomalous is important for understanding the insights, especially in domains such as healthcare and security. This leads to an opportunity of further research in enhancing model transparency and model explanations for the decisions it makes.

5. Integration of multimodal data: Anomaly detection in environments that generate data in different modes (images, video, audio, sensor, etc.) is still very much an unexplored domain.

6. Scalability and Efficiency: As we advance, we generate more and more data, hence the scalability of the model becomes of high importance. Research into such methods that can process large-scale data without compromises in its efficiency and accuracy is very important.

7. Data privacy and Security: Anomaly detection involves several kinds of data and often they might be sensitive hence there is a need of methods that can ensure that the data is secured and the privacy is not compromised.

CHAPTER – 3

PROPOSED SYSTEM

3.1 SYSTEM OVERVIEW

As a pilot study, TransGAN is a transformer-based GAN model that has no convolutions at all. TransGAN is primarily composed of a transformer-based generator that reduces feature resolution over time and is memory-friendly as well as a transformer-based patch-level discriminator [3]. In the original work, a series of techniques are combined such as data augmentation, modified normalization, and relative position encoding, all of which helps to address the instability in the training of GAN. In this project, we implement data augmentation [22] and relative position encoding. We train our model on CIFAR-10 and MVTecAD dataset use the trained TransGAN in two different approaches to solve the problem of anomaly detection. The architecture pipeline of TransGAN is shown below as in the original paper [3] in the Figure 1.

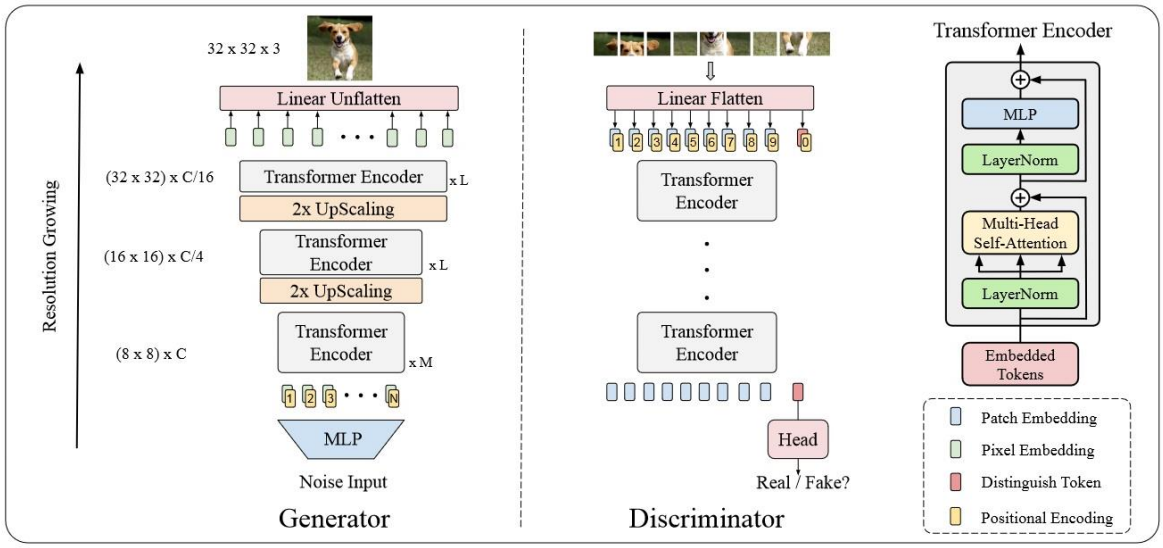

Figure 1: Pipeline Architecture of TransGAN

For MVTecAD dataset we train TransGAN for anomaly detection using an approach called AnoGAN [5]. After training the model on MVTecAD data subset, we get a generator capable of generating fake images, and a discriminator that tries its best to separate real from fake images. Now, we make use of latent space where the trained images are represented in, to find anomalies. Given a new image x, we're going to

compute the representation the image in the TransGAN latent space, and if it is on the zone of normal images, we label it as normal, anomaly otherwise. For CIFAR-10 dataset we explore the anomaly detection architecture which takes its starting point from the paper: [23]. Since CIFAR-10 is an imbalanced dataset, we make use of two discriminators and different losses as discussed in Table 2 and the training pipeline shown in Figure 2 in the next subsections.

3.2 TRANSGAN MODEL

In the TransGAN model, transformer encoder [24] is the basic block of the model. The encoder is made up of multi-head self-attention module and the feed-forward Multiple layer Perceptron (MLP) with GELU non-linearity. For this model we are using layer normalization [25]. The Generator is made up of multiple stages with transformer blocks. It takes random noise input and passes it through the MLP layer. The output along with positional embeddings are fed to transformer encoders recursively to calculate the relation between the tokens. The discriminator takes the patches of images as input which are converted to 1D sequence of token embeddings. Addition of position encoding is followed by feeding it to the transformer encoder. The classification head of the discriminator then outputs the real/fake prediction.

3.2.1 GENERATOR

In the case of TransGAN, the generator comprises multiple layers of transformer blocks at different stages instead of the commonly used CNN-based iterative upscaling of the resolution. Each stage involves increasing of the feature mapping until the target resolution is achieved. The generator takes random noise as input and passes it through MLP. The output vector is reshaped into a feature map and is treated as a sequence of C-dimensional tokens. These token along with positional embeddings are given to transformer encoders as input.

3.2.2 DISCRIMINATOR

The discriminator used here takes the patches of an image as inputs. The image is split into patches and treated as words and hence is converted to a one-dimensional sequence of token embeddings through a linear flatten layer. Post this, the position embedding is added and tokens pass through the transformer encoder. Then this is sent to classification module to classify it as real/fake image.

3.2.3 TRANSFORMER-ENCODER

Transformer encoder [24] is used as the basic block just like described in the original paper. An encoder consists of two parts. The first part is made up of multi-head self-attention module and the other part is a GELU non-linearly activated feed-forward MLP layer.

3.3 LATENT SPACE EXPLORATION

For each class of MVTecAD dataset we are provided with a train. Test and ground-truth (labels) folder. We first train our TransGAN model on non-defective images of any class and test the model on defect classes as well. So, we train the GAN by providing random noise to the generator which generates a fake image from it and we use this non-defective real images and fake generated images as input to discriminator to train that. We make use of reconstruction and adversarial loss of the discriminator to train the GAN to perform better. Once we have our trained GAN, we try to find for a given image 'img' the closest representation of it in the latent space by picking a random point in the latent space and use gradient descent to walk through the space for the purpose. The loss function that we use here is the L1 loss between the generated and given input image given by:

$$\mathcal{L}_R(z_\gamma) = \sum |x - G(z_\gamma)|$$

We also compare the representation of real and fake images in the feature space of the discriminator. Since each layer of the discriminator while training has captured a lot of features, we use it to find the perceptual difference between the real and fake images that discriminator uses to distinguish them. AnoGAN authors use L1 loss as shown in their paper:

$$\mathcal{L}_D(z_\gamma) = \sum |f(x) - f(G(z_\gamma))|$$

where f (·) represents the intermediate layers of the discriminators. For convenience in this research, we only consider the first few layers of the discriminator for the calculation of the discriminator loss.

Finally, we balance both the losses to find the overall loss and use it to find the

latent representation that minimizes the loss. From this we get the closest real image that we can generate. Now we test it with defects and see the reconstructed images are not that great because they do not fall in the manifold of normal image representations. We use the losses of images with which we trained our model as threshold and if these loses are more for test image then it is labeled anomaly, normal otherwise.

3.4 IMBALANCED DATASET ANOMALY DETECTION

CIFAR-10 dataset is an imbalanced dataset for anomaly detection. Hence, we need to deal with the problem as discussed in Table 2 below. Since we divide the dataset originally consisting of 10 classes, into 2 classes as normal and abnormal by setting one class as normal and others as abnormal, the dataset is clearly unbalanced. So, we use k-means sampling to get the number of samples for anomaly class while subsampling.

The two discriminators are used only in the training, both of which are completely identical. They are used to treat anomalous and normal data separately. The generator used takes noise as input and gives image as output but for this purpose we include a feature mapper module before the generator so that the input image is mapped to the latent vector that can be accepted by the generator as the input. In the feature mapping we use a pre-trained ResNet50 as our feature extractor and use Linear layers with RELU for mapping the extracted feature to the latent space of the generator. The training pipeline of the model is shown in Figure 2 below.

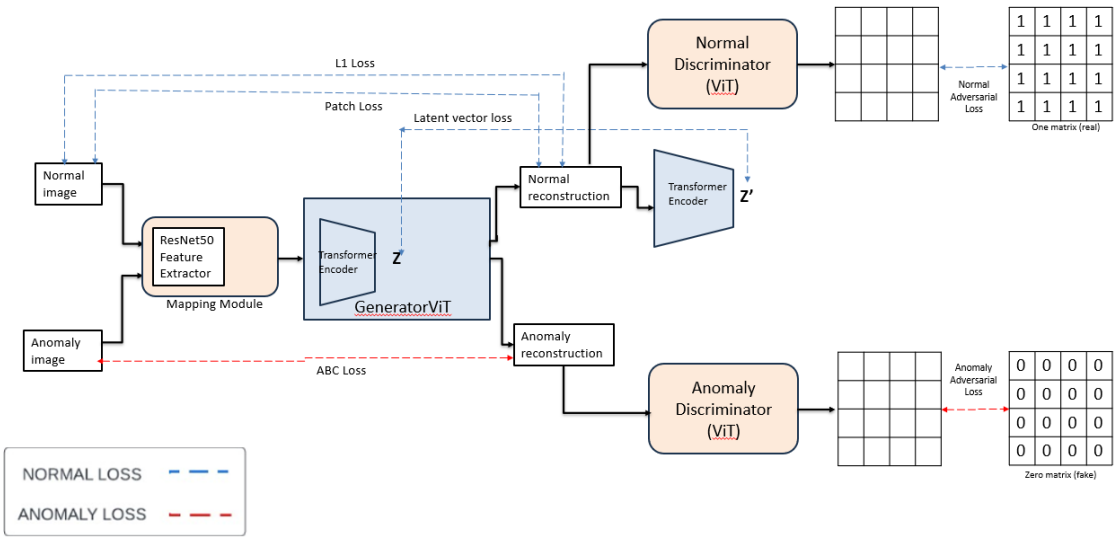

Figure 2: Training Pipeline for anomaly detection in imbalanced dataset

For the training we use a weighted combination of several different losses to update the weights, separately for normal and anomalous images. The loses that we use here include six loses for the generator and one for each of the discriminator. For each discriminator we make use of loss function from LSGAN [27], where a and b are labels of fake and real data shown in Eqn. 1.

$$\min_D V_{\text{LSGAN}}(D) = [(D(x) - b)^2] + [(D(G(x)) - a)^2] \quad (1)$$

The six loses used to train the TransGAN generator include four losses for the normal or good images and two for anomalous images.

L1 loss (Eqn. 2) which is the reconstruction loss where x is the original input image and G(x) is the generated image from the generator. It penalizes the generator based on how different the generated image is from the original input

$$\mathcal{L}_{\text{recon}} = \|x - G(x)\|_1 \quad (2)$$

The patch loss (Eqn. 3) where we divide both the images into multiple patches and select the average of the biggest reconstruction errors among all the patches.

$$\mathcal{L}_{\text{patch}} = f_{\text{avg}}(n)(\|x_{\text{patch }(i)} - G(x_{\text{patch }(i)})\|_1), i = 1, \dots, m \quad (3)$$

The latent vector loss (Eqn. 4) [26] is calculated between real and generated image after passing them through the encoder.

$$\mathcal{L}_{\text{enc}} = \|G_E(x) - G_E(G(x))\|_1 \quad (4)$$

The adversarial losses for normal and anomalous images (Eqn. 5 and 6) are used to update the generator. Here 1 is the actual label for images after reconstruction which belong to normal class and 0 is the label for the images that should be classified as anomalous.

$$\min_G V_{\text{LSGAN}}(G) = [(D(G(x)) - 1)^2] \quad (5)$$

$$\min_G V_{\text{LSGAN}}(G) = [(D(G(x)) - 0)^2] \quad (6)$$

And the ABC loss (Eqn. 7) is used to maximize L1 loss for anomalous images, which is reconstruction error modified by adding the exponential and log function to

solve the imbalance due to large difference between the reconstruction of good and anomalous images.

$$\mathcal{L}_{\mathrm{ABC}} = -\log\left(1 - e^{-\mathcal{L}_{\theta}(x_i)}\right) \quad (7)$$

Challenge	Approach
Unbalanced class	K-means sampling
Loss scale imbalance	Loss function weight search
Discriminator bias	Two discriminators
Size imbalance between defect and object	Reconstruction-based methods

Table 2: Imbalance Challenges and solutions

CHAPTER – 4

EXPERIMENTS & RESULTS

In this section, we perform substantial experiments to evaluate our model in a way of leave-one-class-out anomaly detection on commonly used datasets - CIFAR-10 and MVTecAD using two different approaches. We log the different losses and try to present the effects of those losses by plotting them in different graphs to understand how they affect the training and anomaly detection.

4.1 DATASETS

Datasets used for the experimentation includes:

- MVTecAD: MVTecAD is one of the most popular bench-mark datasets used for the task of anomaly detection which focuses on different inspections. It contains 5000 images of fifteen different objects such as ‘hazelnut’, ‘metal nut’, ‘screw nut’, ‘bottle’, etc. Every category contains a collection of flawless training pictures, along with a test set that includes images with a range of defects and some that are defect-free. We perform anomaly detection on each class training the model on the ‘good’ subset of the images and perform anomaly detection using latent walk on all the test images of the class.

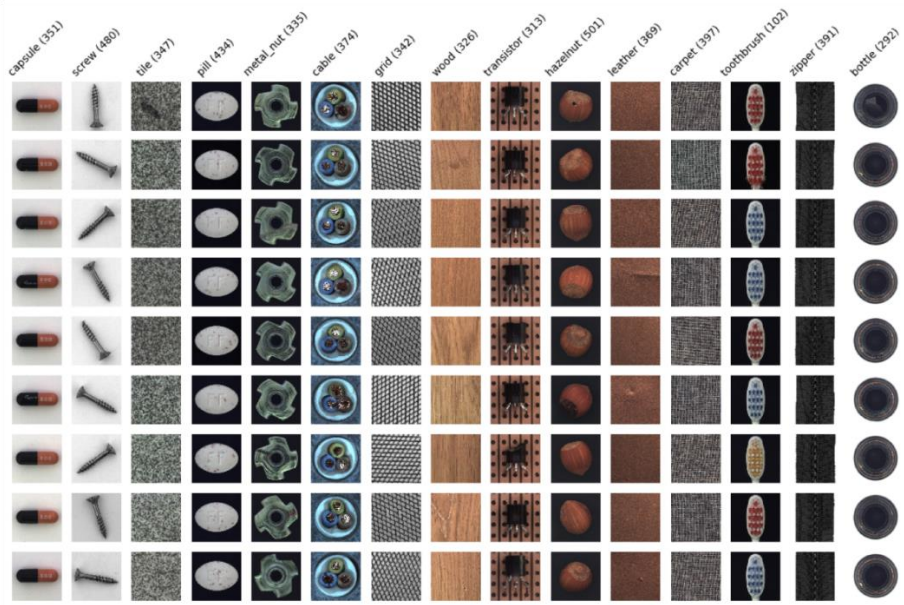

Figure 3: MVTecAD Dataset

- CIFAR-10: CIFAR-10 is another widely used dataset which comprises color images of 32x32 pixels from 10 classes of objects. Each class has 5000 training images of a particular object. Classes in the dataset include: airplane, automobile, bird, cat, horse etc. We select one class of images as anomaly and other images as normal data. Then we train our model on both normal and anomalous samples using separate discriminators to deal with each type and then perform anomaly detection on them using only the generator.

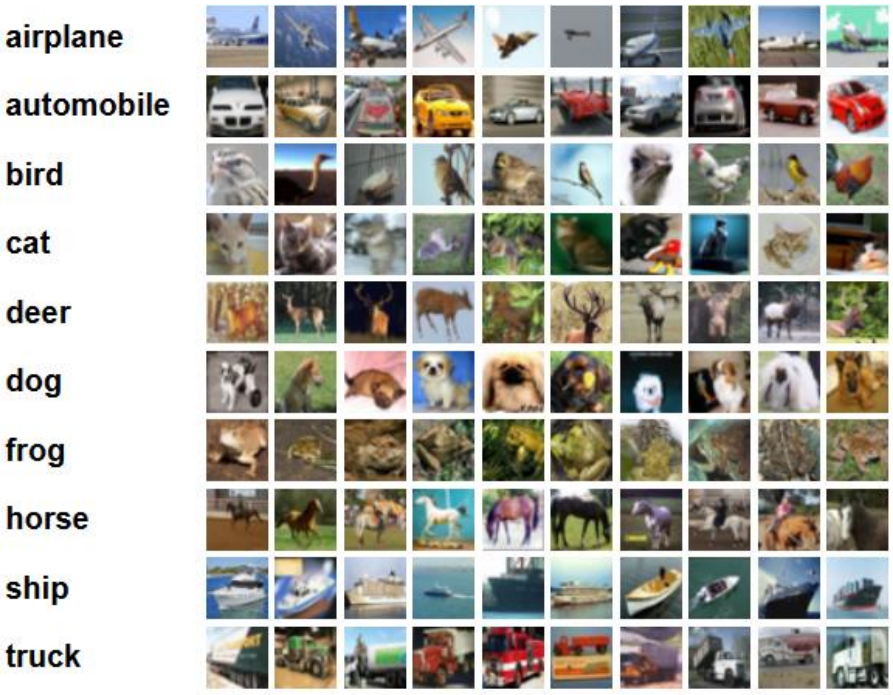

Figure 4: CIFAR-10 Sample dataset

4.2 TRAINING DETAILS:

Our experiments are performed on GPU T4x2 with 15GB Memory with Pytorch framework on Google Colab and Kaggle platforms. In the training process we use Adam optimizers with momentum b1=0.001 and beta2=0.99. We experiment with learning rate of 0.0001 for both generator and discriminator for different batch sizes for each of the datasets.

We try training the model with each class and different parameters to see which yields better results. The training architecture of each dataset is different and require varied time for training and testing.

4.3 ANOMALY DETECTION IN MVTEC-AD:

We use the hyperparameters as shown in the Table 3 for the training of the MVTec-AD dataset. We train for different class of objects and try to visualize the anomalous region after performing the latent walk to define the thresholds to find anomalous images.

Image Size	Epochs	Batch Size	Learning Rate	Latent Dim	Number of Iterations
64	350	8	0.0001	1024	1000

Table 3: Hyperparameters for training of MVTec-AD anomaly detection

We see in Figure 5, as epochs increase the images that are generated transform from just looking like a noise to more closely like the hazelnut images. Due to complexity of our model and hardware limitations we observe that the images are no more noise but is not producing the most realistic fake images. We think this only goes on to present further need of research and improvements that are still underway to make this model production grade.

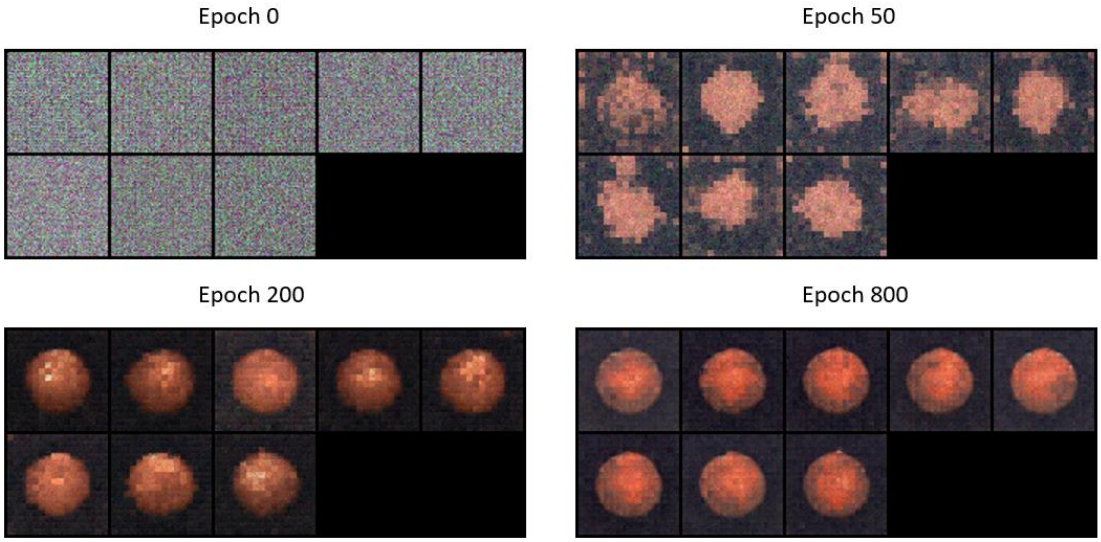

Figure 5: Generated Images of Hazelnut Dataset

Similarly, we train the model on other classes of MVTec-AD dataset as well and here are few examples of how the training went for few of those classes such as metal-nut, toothbrush, cable-wire, bottle and capsule in the Figures 6,7,8,9 and 10 respectively. We can also see a batch of generated images as well as the graph of

discriminator and generator loss after running for 300 epochs.

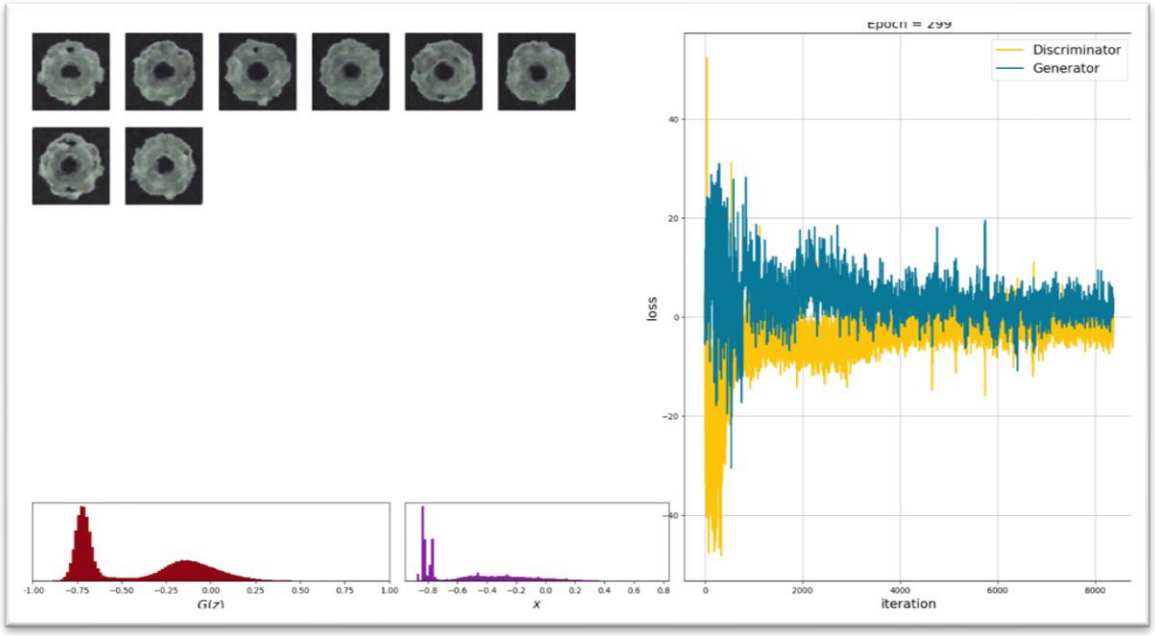

Figure 6: Training for Metal-nut

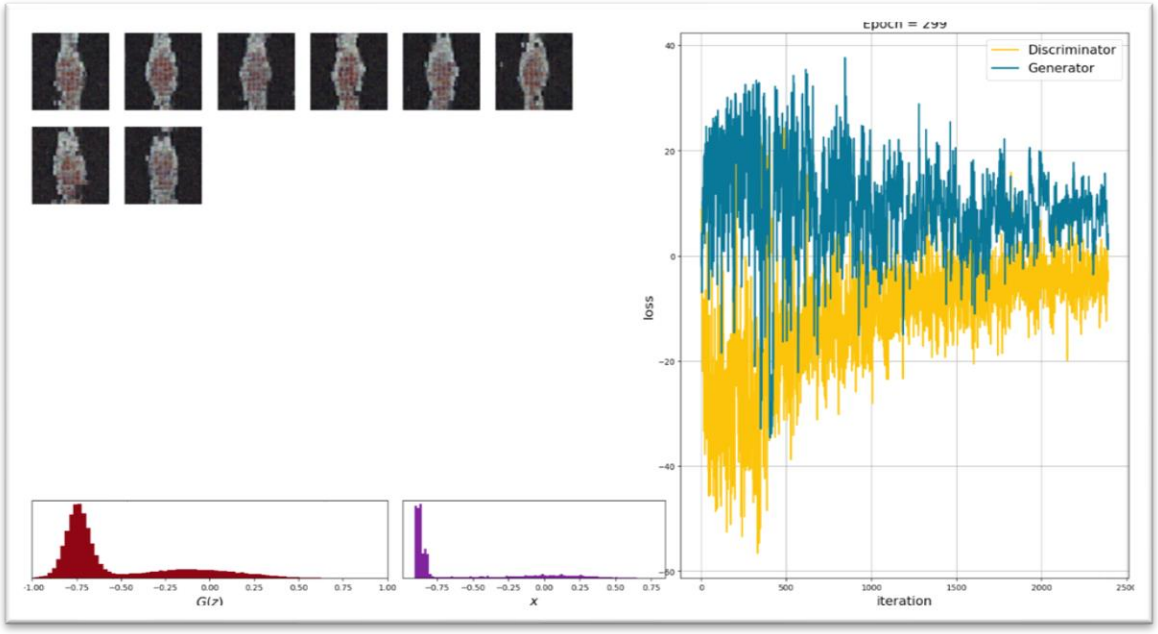

Figure 7: Training for Toothbrush

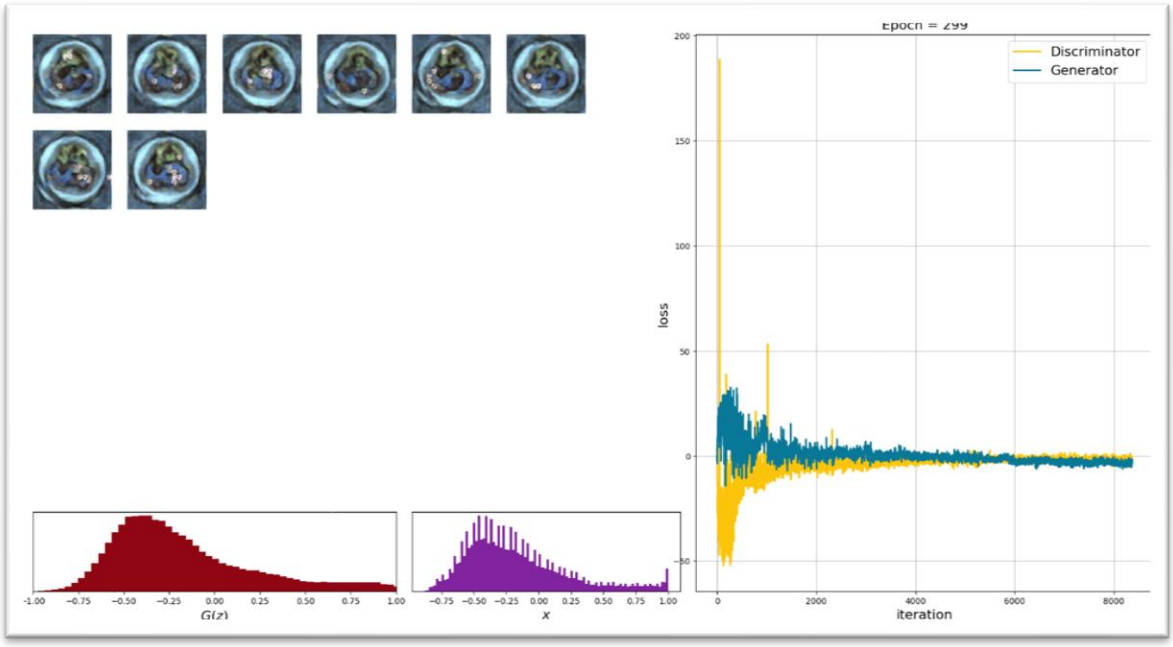

Figure 8: Training for Cable Wire

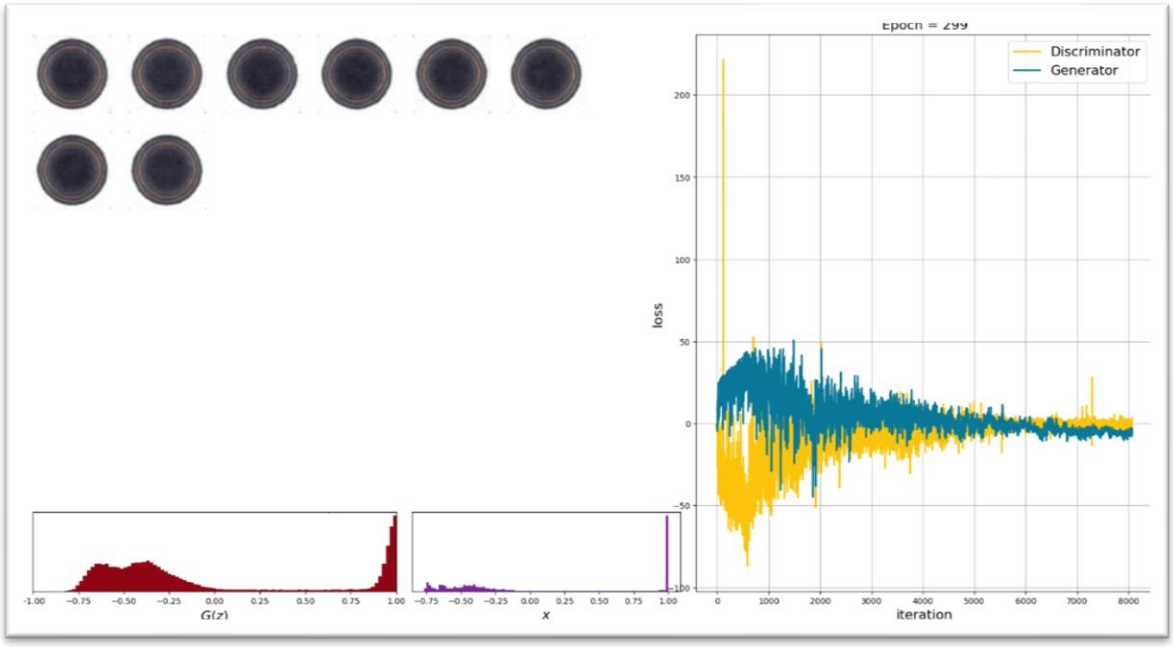

Figure 9: Training for Bottle

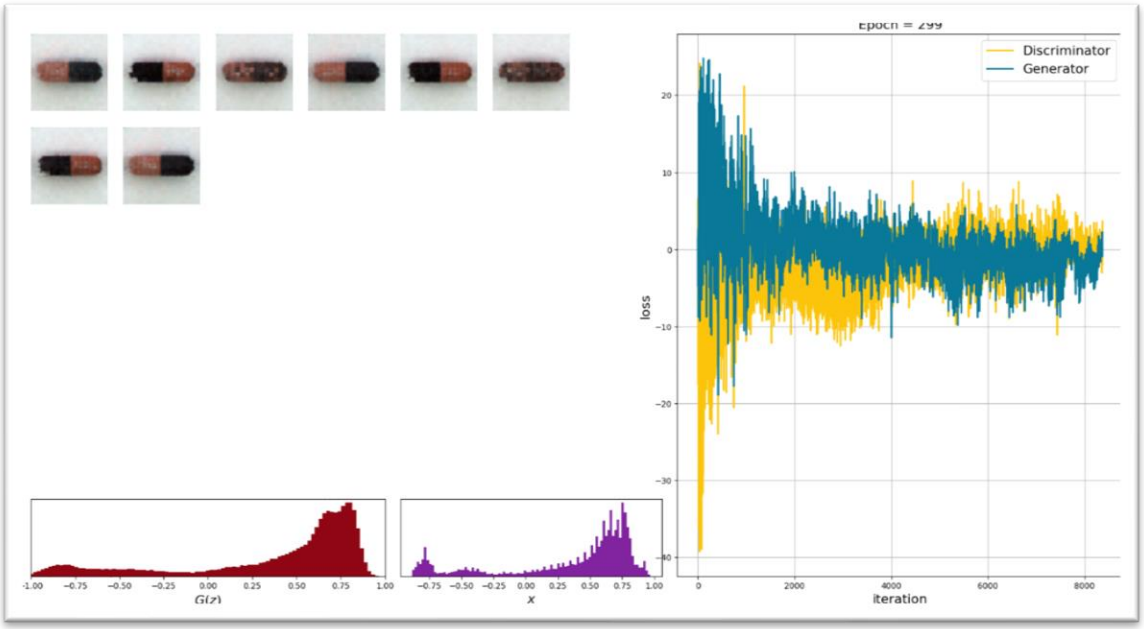

Figure 10: Training for Capsule

Once we had finished training our model on our data-set we make use of latent space representation of the generator to find the image that looks closest to a given real image. We do that by fixing a real image. Generated Images of Hazelnut and take random noise vector z. Then we make use of L1 loss as discussed earlier to see the difference between G(z), the generated image from the noise vector z to the generator, and the real image. We use Adam optimizer to optimize z with different learning rates(lr) for different number of iterations and for lr= 2e-2 and 1000 epochs, we see the loss decreasing as seen in Figure 11, even though it is not the way we expected it to be. There could be several reasons for that including learning rate, some noisy generated images, and the fact that due to hardware restrictions we use small batch size which restricts the model from learning the gradients better.

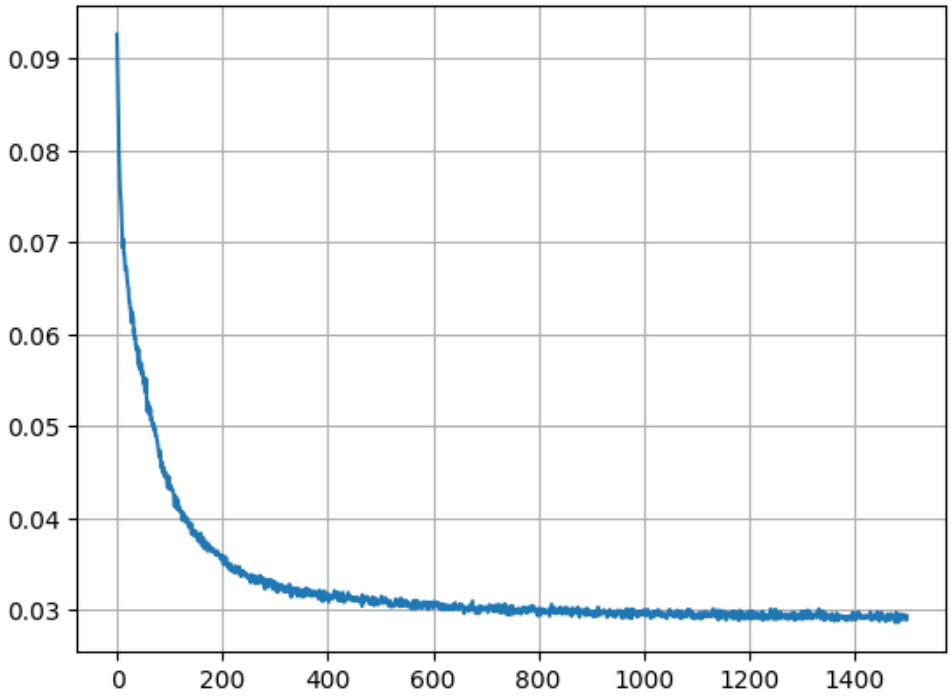

Figure 11: Z-loss optimization graph

With this we also include the perception loss from discriminator and experiment with different weights to balance the combined losses. Now we perform the latent walk on a test real image with defects to generate an image. We see that the generated image is not very great and the reconstruction is very poor. We explore this high reconstruction and discriminator loss, and identify anomalous regions in images. We compute a “residual image” - the difference between the input test image and reconstructed image - and areas of the residual images with large values should correspond to anomalous regions, since the original and reconstructed image are maximally different in these regions. We then plot these images side by side as you can see in Figure 12.

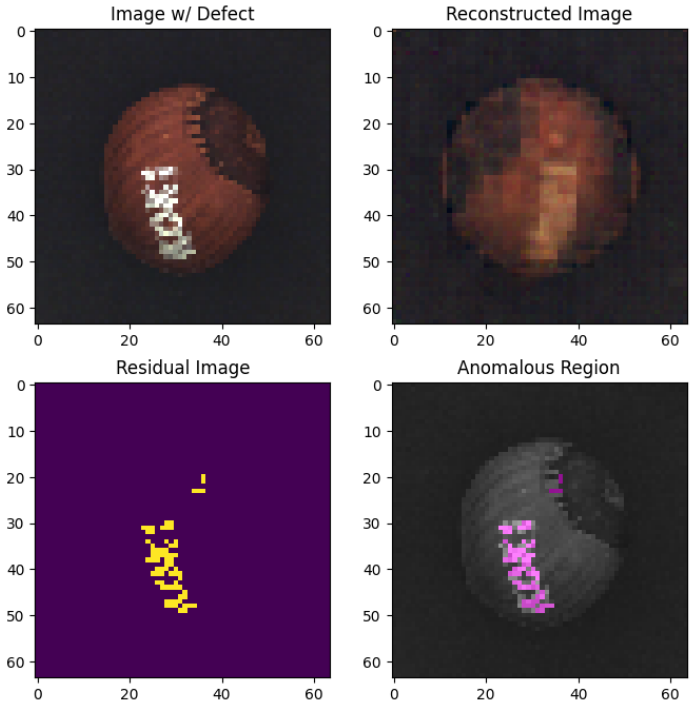

Figure 12: Anomaly Reconstruction and highlighting of Hazelnut

Similarly, we do the same with other classes of MVTec-AD dataset and the anomaly highlighting can be seen in the Figures 13-17 of some of those classes such as metal-nut, toothbrush, cable wire, bottle and capsule respectively.

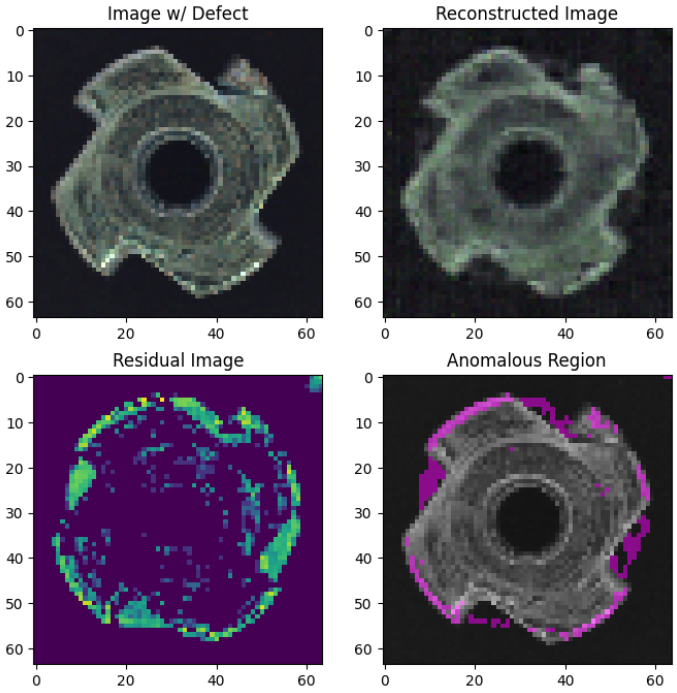

Figure 13: Anomaly Reconstruction and highlighting of Metal-Nut

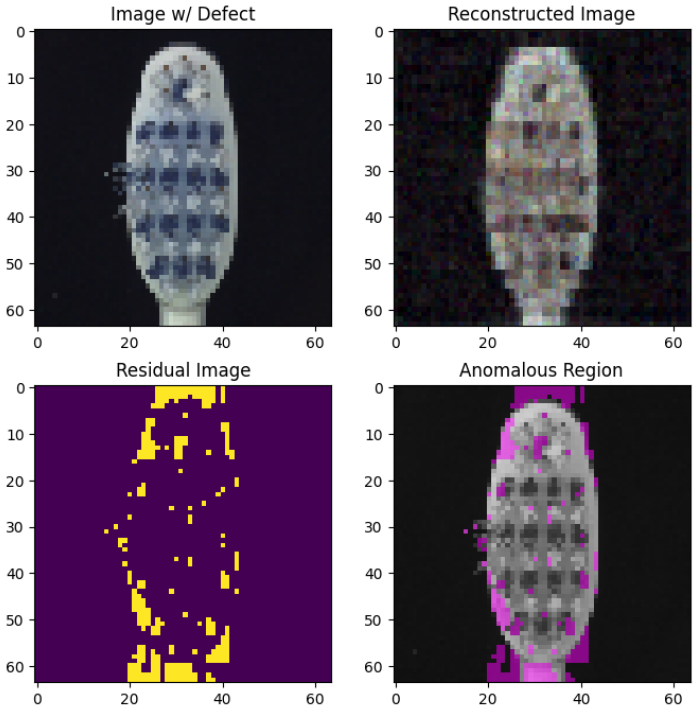

Figure 14: Anomaly Reconstruction and highlighting of Toothbrush

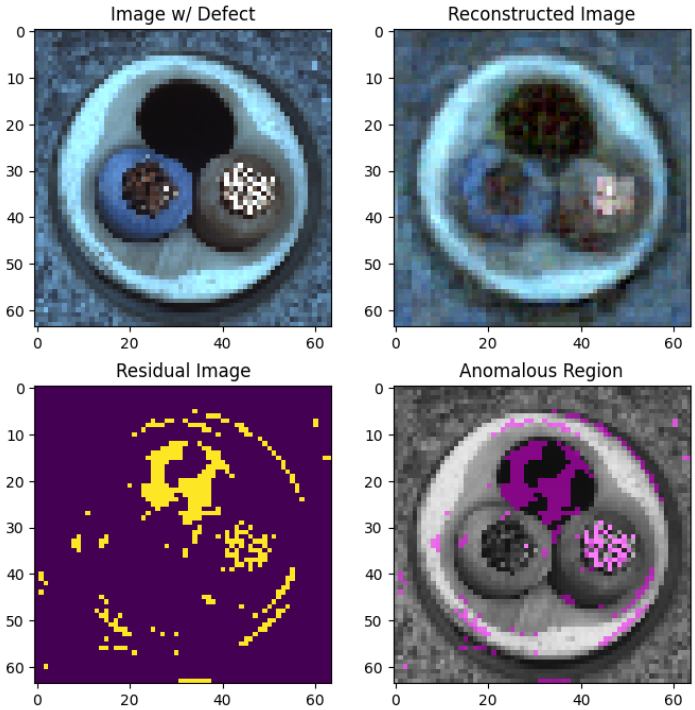

Figure 15: Anomaly Reconstruction and highlighting of Cable wire

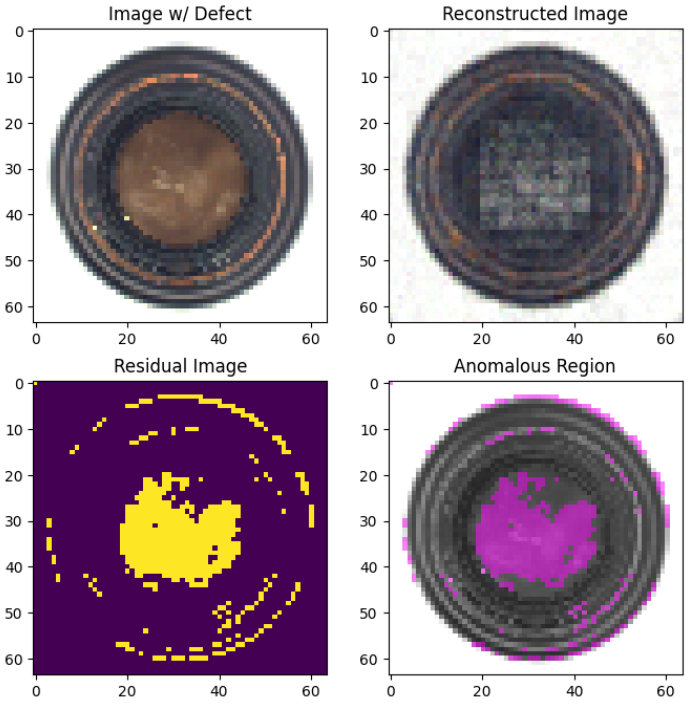

Figure 16: Anomaly Reconstruction and highlighting of Bottle

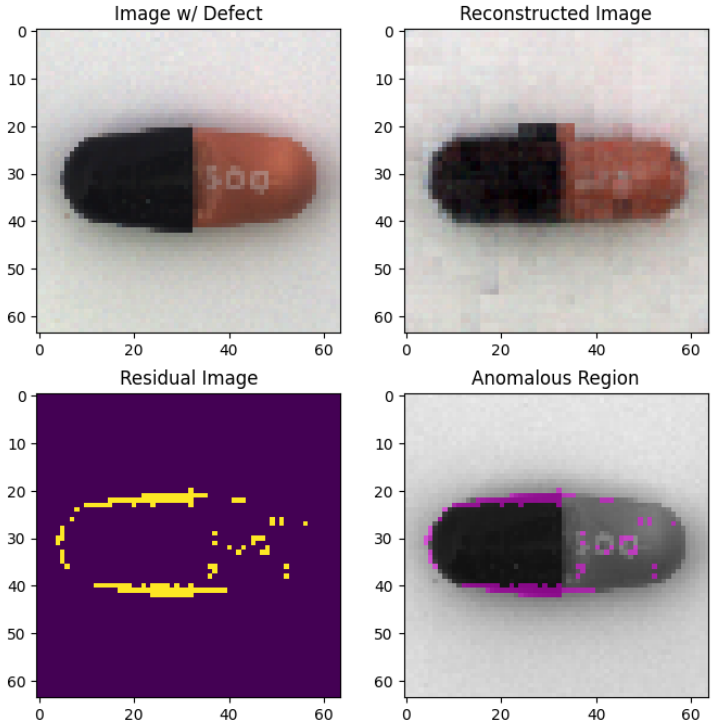

Figure 17: Anomaly Reconstruction and highlighting of Capsule

We now run this for our training dataset and try to fix a residual and perception threshold to find the anomalies in the images of the test dataset that includes normal hazelnuts as well as ones with the defects. We plot the residual loss and discriminator

loss of our data as shown in Figure 18 below, but unfortunately, we see a medium to high reconstruction loss for our normal images as well, which should not have been. But we think this is because of the complexity and hardware requirements and can be improved with better hardware. However, we do see a pattern with our discriminator - the images with defects have a high discriminator loss.

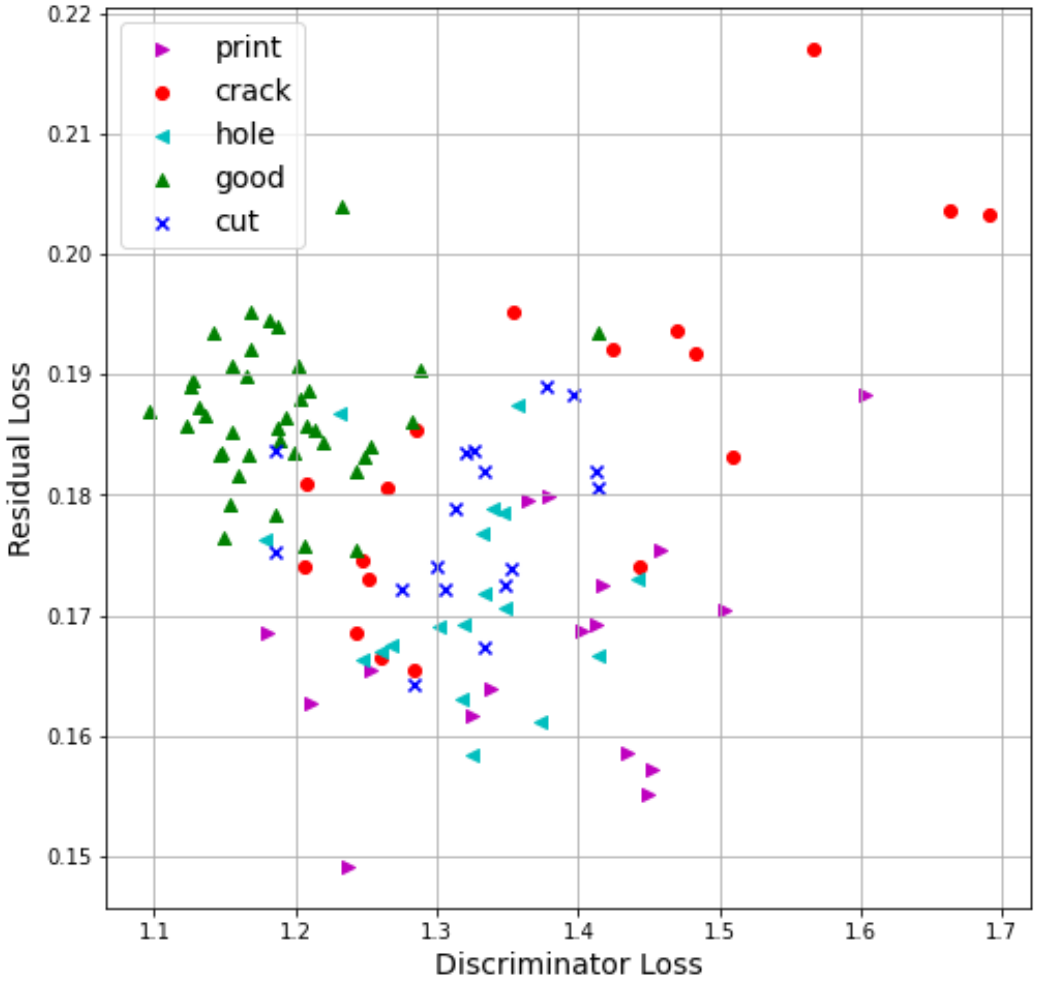

Figure 18: Residual and discriminator loss of test images of 'hazelnut'

We take this inference and use discriminator loss for anomaly detection. We classified images into anomaly class if their reconstruction and discriminator loss exceeded their set threshold. We set the discriminator threshold to be 1.5 and reconstruction to be 0.5. When we run this on our test samples of 'good' data we get an accuracy of 0.9 and for defects an accuracy of 0.6. When we compare this to the original paper [6] the accuracy obtained by AnoGAN for hazelnut data for good images was 0.83 and for anomalous 0.16.

We do similar mapping of the reconstruction and discriminator losses for other classes and set the threshold and classify the images of those class into ‘good’ and ‘anomalous’ classes. The accuracy obtained for each of those classes for good and anomalous images has been given below in the Table 4.

Class	**Accuracy**	
	Good Images	**Anomalous Images**
Bottle	0.91	0.5
Cable	0.95	0.41
Capsule	0.91	0.52
Carpet	0.8	0.4
Grid	0.9	0.55
Hazelnut	0.9	0.6
Leather	0.91	0.34
Metal nut	0.86	0.44
Pill	0.95	0.47
Screw	0.5	0.5
Tile	0.95	0.33
Toothbrush	1.00	0.31
Transistor	0.95	0.65
Wood	0.92	0.55
Zipper	0.8	0.45

Table 4: Accuracy of TransGAN in anomaly detection for MVTec-AD dataset

4.4 ANOMALY DETECTION IN CIFAR-10:

We use the hyperparameters as shown in the Table 5 below for the CIFAR-10 dataset training on the TransGAN model for anomaly detection.

Image Size	Epochs	Batch Size	Learning Rate	Latent Dim
32	10	1	0.0001	1024

Table 5: Hyperparameters for training of MVTec-AD anomaly detection

Just like on MVTecAD dataset we perform several transformations on the CIFAR-10 dataset when we load the dataset from pytorch. We train the dataset and with each epoch we can see in the Figure 19 the images getting better.

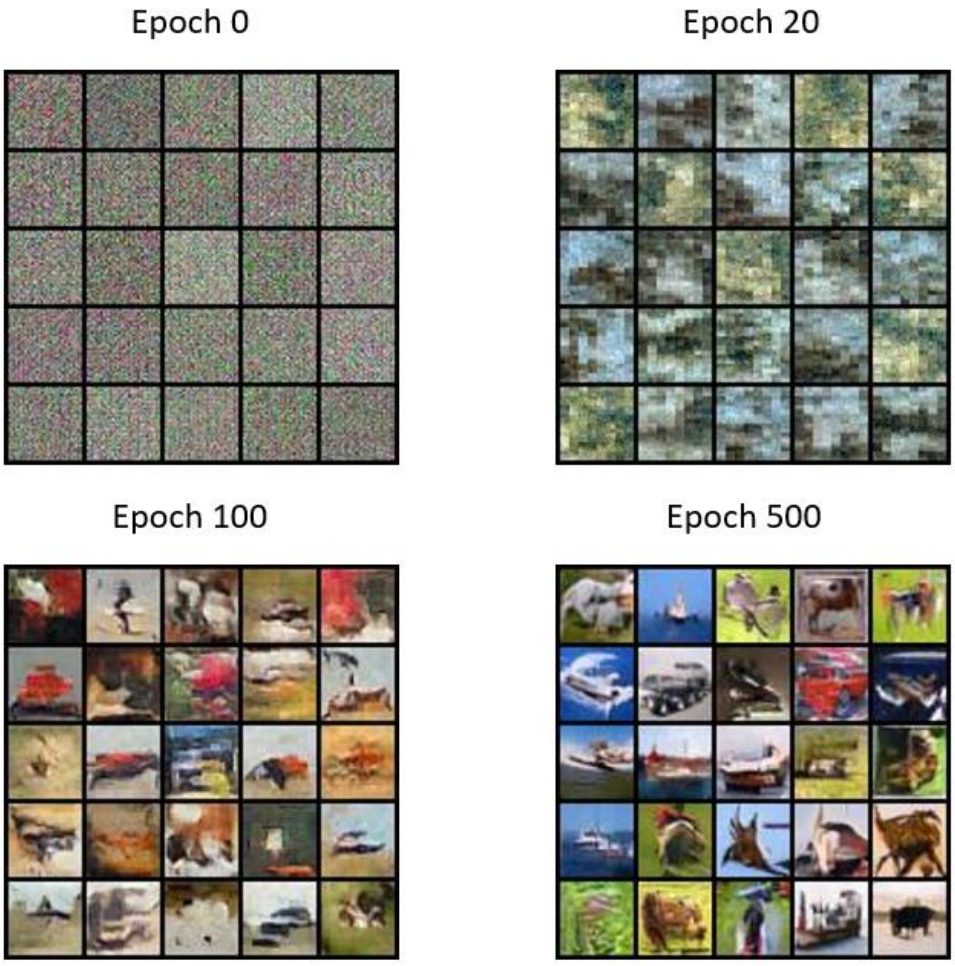

Figure 19: CIFAR-10 model after training

Now when we give a random noise, we can clearly see that this model gives images in the batch that resembles the real images of the CIFAR-10 dataset. But for the approach we are experimenting with this dataset, we want to pass real image as input to the generator. To achieve our goal, we utilize cutting-edge techniques in deep learning.

Initially, we employ the ResNet50 model, which is pre-trained and represents the pinnacle of current technology, to extract intricate features from our image dataset. These features are then processed through a mapping function that converts them into the latent space of the generator. This mapping is performed using a linear function, enabling the transformation of image representations into a format compatible with the generator's latent space.

We refine this mapping process by minimizing the reconstruction loss, as depicted in Figure 20, where a noticeable decrease in loss is evident. Due to the large volume of images and the complexity of our model, training is limited to just few epochs before integrating the model into an anomaly detection algorithm.

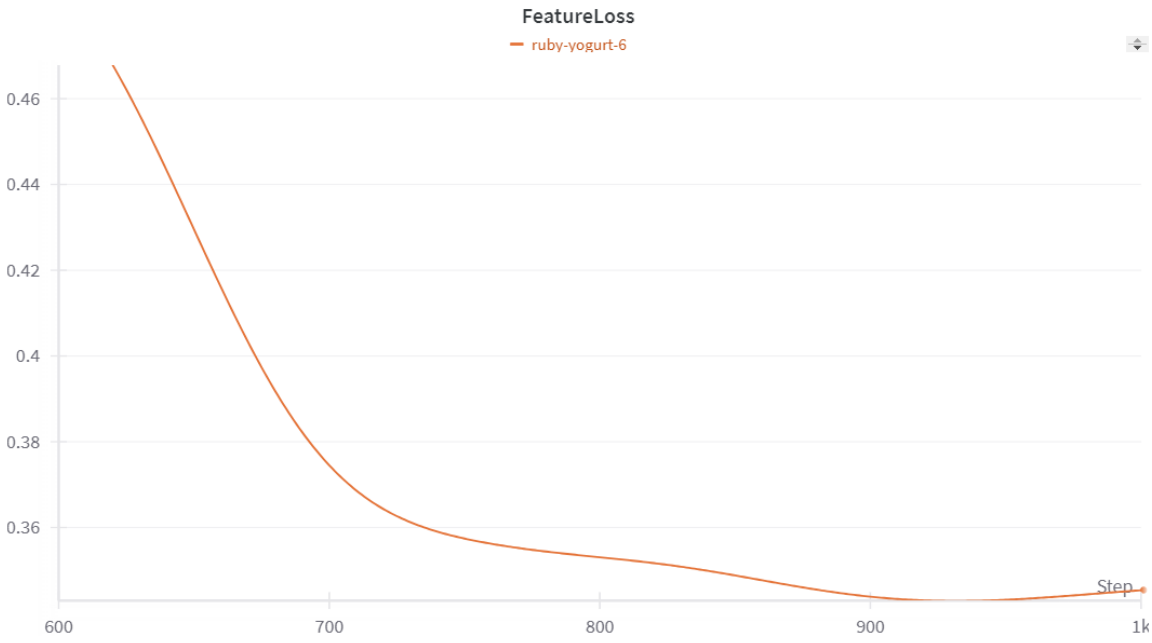

Figure 20: L1 loss during feature extraction training

This integration involves incorporating the Mapping module alongside TransGAN, utilizing identical discriminators. However, the training objective is adjusted to differentiate between anomalous and normal images. When an image is flagged as anomalous, the reconstructed image is directed to the anomaly detector, where the ABC loss is computed. Conversely, for normal images, the reconstruction

loss and latent loss are computed by the normal detector. Additionally, we calculate patch loss between the input and reconstructed images, as well as adversarial losses for both anomalous and normal images, as defined in Equations 1 to 7.

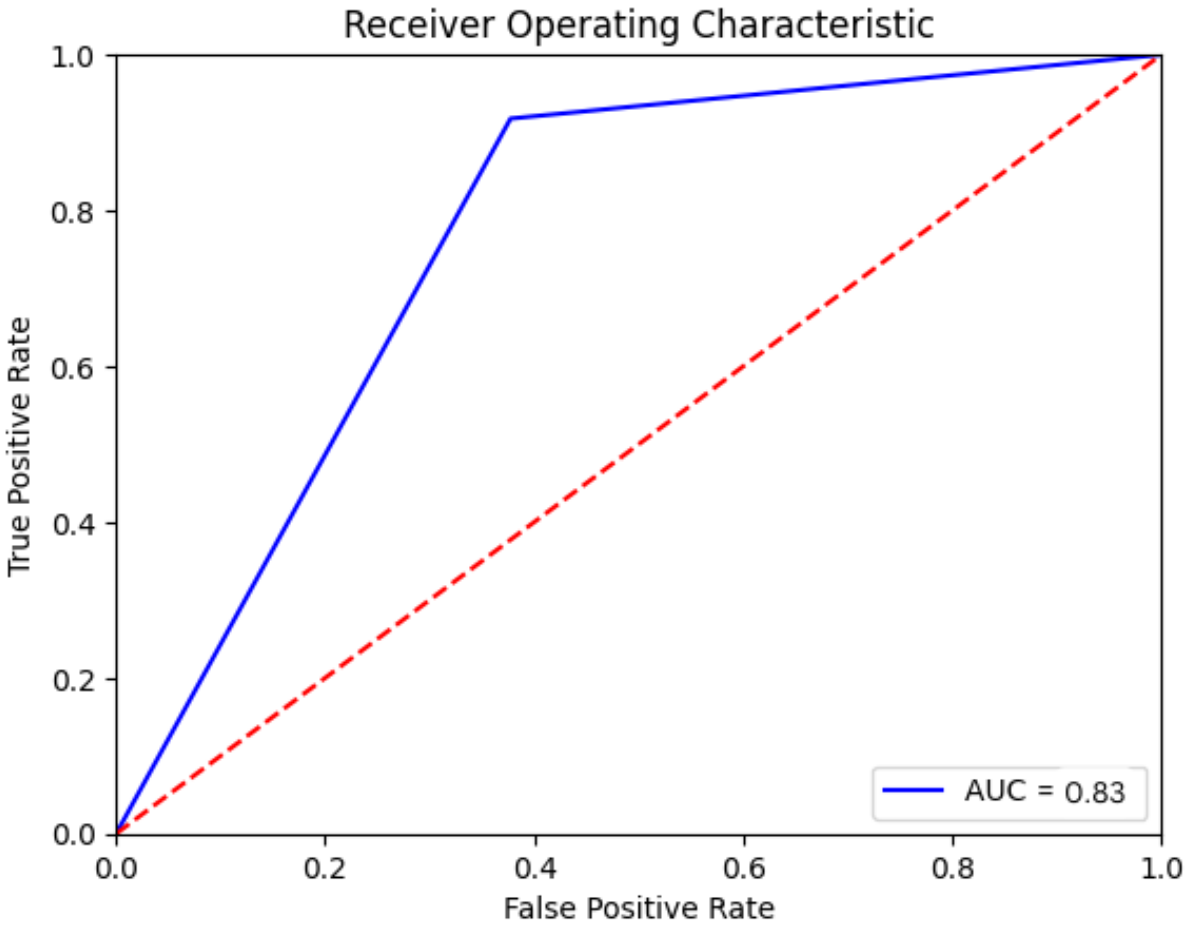

Figure 21: AUROC score when normal class = 8

Training the model spans ten epochs, employing the Adam optimizer with a batch size of 1 and a threshold of 0.5. The model's effectiveness depends on the weights and may vary depending on the designated normal class. To assess the model's performance, we designate class 8 as the normal class, achieving a commendable Area Under the Receiver Operating Characteristic (AUROC) score of 0.83, as illustrated in Figure 21.

The Table 6 below gives the auroc score obtained for each class of the dataset in this experiment.

Class	0	1	2	3	4	5	6	7	8	9	**AVERAGE**
Score	0.81	0.82	0.74	0.72	0.76	0.75	0.82	0.8	0.83	0.81	**0.79**

Table 6: Auroc score for the model on CIFAR-10 dataset

CHAPTER – 5

CONCLUSION & FUTURE WORK

As part of our experimentation into anomaly detection techniques we set out to utilize the capabilities of the TransGAN model a state-of-the-art transformer-based Generative Adversarial Network (GAN) with a reputation for producing lifelike images from noise input. There were two primary stages to our project both focused on clarifying TransGANs capabilities in anomaly detection tasks on various datasets. Using the MVTecAD and CIFAR-10 datasets we aimed to advance anomaly detection techniques by incorporating cutting-edge generative models into our approaches.

With its intricate architecture TransGAN offered both possibilities and difficulties. Although we were excited by its potential to generate realistic images there were substantial obstacles due to the training process' inherent complexity and resource requirements. Our preliminary tests demonstrated that careful training procedure optimization is necessary to reduce computational overhead and hasten convergence. Some promising directions for further research and development include the application of techniques like transfer learning and progressive training which may help improve model performance and expedite training.

The importance of discriminator architecture in anomaly detection was one of the most important lessons learned from our TransGAN experiments. Our understanding of the complexities involved in differentiating between normal and anomalous cases deepened our understanding of the significance of creating discriminators that can pick up on minute details that point to anomalies. It has been suggested that one way to improve the discriminative power and robustness of models is to incorporate attention mechanisms or customize discriminators for particular anomaly detection tasks.

In spite of our most useful makes an attempt there were huge difficulties in precisely appropriate among ordinary and anomalous photographs in our checks using the MVTecAD dataset. sudden patterns within the discriminator mistakes ended in misclassifications exceptionally while there were baby anomalies that have been invisible at 64x64 image resolution. This verified how important hardware substances and image resolution are to aberration detection tasks emphasizing the necessity for extra resilient hardware infrastructure and higher- resolution image statistics to get around these constraints.

The CIFAR-10 dataset presented similar difficulties in our investigation as we were only able to obtain an average AUROC score of 0. 79 which was below the standards set by other anomaly detection algorithms. Our TransGAN implementation was a novel approach but it also showed that in order to achieve competitive performance further research and refinement are needed. The limitations we experienced during our experiments highlighted the intrinsic complexity of anomaly detection tasks and the necessity for creative solutions to effectively address them.

The success of our project stems from investigating new methods for anomaly detection with cutting-edge generative models like TransGAN. We have shown the versatility of TransGAN for anomaly detection tasks by testing the limits of anomaly detection techniques and experimenting with various datasets and training regimens. These achievements were however somewhat dampened by a number of difficulties and restrictions faced along the route.

Looking ahead our work establishes the foundation for generative model-based anomaly detection advancements in the future. We can contribute to the larger field of machine learning and artificial intelligence as well as further advance the state-of-the-art in anomaly detection by addressing the limitations that have been identified and building upon our achievements. To fully realize the potential of generative models in anomaly detection we plan to conduct ongoing research and development to improve model architectures streamline training processes and assess performance on a variety of datasets.

To sum up our project has shed important light on the difficulties and possibilities involved in applying TransGAN to anomaly detection tasks. Although we have made progress in proving the approaches viability and identifying possible areas for development there are still major obstacles to overcome. The project has brought to light the need for additional study and development in order to improve training protocols assess performance across a variety of datasets and fine-tune the model architecture. Our work establishes the groundwork for future research in this field despite obstacles like scarce computational resources and challenges in identifying subtle anomalies.

APPENCICES

APPENDIX 1: Understanding Generative Adversarial Network

Generative Adversarial Networks (GAN) is one of the most significant deep learning models. It comprises two neural networks, generally CNN, that act as two parts of the GAN – the generator and the discriminator. The generator strives to produce synthetic data samples that can be considered to be close enough to the real data, while the task of the discriminator is to distinguish real and fake samples.

Through adversarial training, the generator learns to generate more and more realistic data, while the discriminator becomes more capable of distinguishing the real and fake samples. The working of a general GAN for image data can be illustrated as shown in Figure 22.

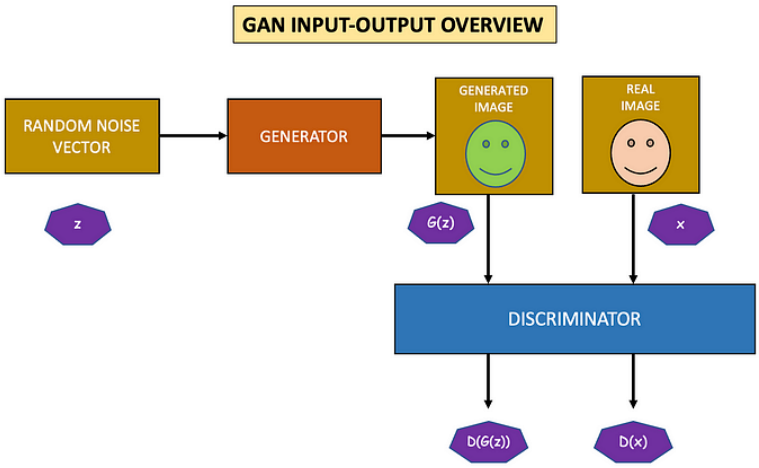

Figure 22: GAN input output workflow

Here we can see the generator takes a noise vector (z) and learns to generate samples (G(z)) from that. The discriminator can take either the generated sample as input or a real sample (x) and give an output D. When generator is trained the weights and biases of the discriminator are fixed and vice versa.

A GAN can be explained as a minimax game played between the generator and discriminator where both models are adversarially trained to defeat each other's purpose. Different loss functions are used to update the weights and biases of both the generator and discriminator. As training progresses generator gets better at generating samples making it

harder for the discriminator to tell apart real from fake samples.

GANs have several advantages and disadvantages and can be used for various domains. The advantages of GANs include – Synthetic data generation, high quality results, unsupervised learning, versatility among various others. Whereas the disadvantages include training instability, model drop, computational cost, overfitting, bias and interpretability and accountability.

GANs can be used for various tasks such as:

- Image Synthesis and Generation: GANs are frequently utilized for the synthesis and generation of images. These models have the capability to produce new, realistic images that closely resemble the training data by learning the underlying distribution of the dataset. The advancement of lifelike avatars, high-resolution photographs, and original artwork has been significantly facilitated by the utilization of these generative networks.

- Image-to-Image Translation: GANs can be applied to tasks involving image-to-image translation, where the objective is to convert an input image from one domain to another while preserving its essential characteristics. For example, GANs can be employed to transform images from day to night, convert sketches into realistic images, or alter the artistic style of an image while maintaining its content.

- Text-to-Image Synthesis: GANs have been leveraged for generating visual content from textual descriptions. By taking a text input, such as a phrase or a caption, GANs can generate images that correspond to the provided description. This application holds promise for revolutionizing the creation of realistic visual content based on textual instructions.

- Data Augmentation: GANs offer the capability to augment existing data, thereby enhancing the robustness and generalizability of machine learning models. By generating synthetic data samples, GANs contribute to expanding the diversity and richness of the training dataset, leading to improved model performance across various tasks.

- Data Generation for Training: GANs play a crucial role in enhancing the resolution and quality of low-resolution images. Through training on pairs of low-resolution and high-resolution images, GANs can generate high-resolution images from low-resolution inputs. This capability facilitates improvements in image quality across diverse applications,

including medical imaging, satellite imaging, and video enhancement.

GANs is a cutting-edge approach to generative modeling and can leverage different deep learning models like CNN. Hence understanding how it works is very important to understand how we use it to perform anomaly detection. There are several different types of GANs using different deep learning models and for various functionalities. We try to implement Transformer based GAN in this project for anomaly detection.

APPENDIX 2: Understanding Transformers

Transformers were originally introduced for natural language processing tasks. Since then, they have revolutionized various domains and industries within machine learning. Unlike traditional RNNs or CNNs, transformers make use of self-attention mechanisms to capture long range dependencies with sequential data. Transformers consist of an encoder and a decoder which work simultaneously to process the input and give us the output. The architecture of the transformer can be seen in the Figure 23 below.

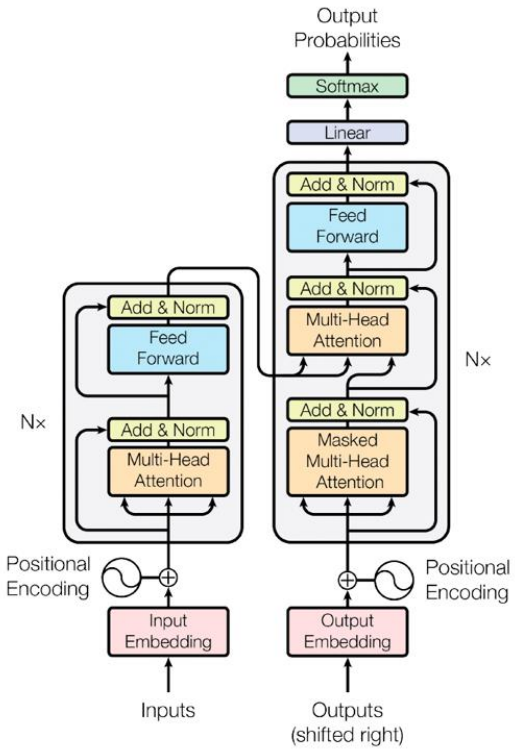

Figure 23: Transformer Architecture

The encoder block takes words as input and converts it into abstract numerical representation and saves them in memory. The decoder generates word one by one referring to

the generated output and the memory through attention.

In this project we make use of vision transformers (ViT). ViT was introduced in 2021 and have extensible applications in popular image processing tasks such as object detection, classification, image segmentation, etc. Just like normal transformer blocks, in ViTs, images are provided as sequences and class labels of the image are predicted, Input images are taken as sequence of patches as a flattened vector. The architecture of the ViT can be seen in the Figure 24 below.

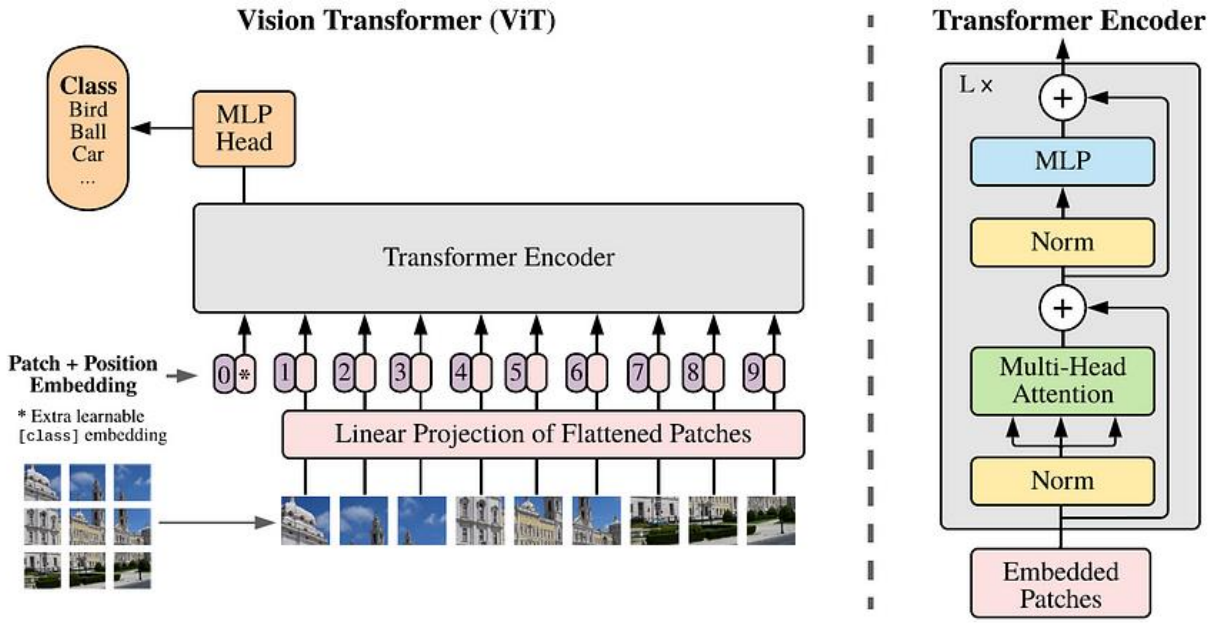

Figure 24: Vision Transformer Architecture

The image is first split into patches which are then flattened. Lower dimensional linear vector embeddings are then produced from these flattened patches. Each patch has a position which is added to these embeddings as position embeddings and then this sequence is fed as input to the standard transformer encoder.

As you can see there are multiple layers in the ViT encoder and each block consist of three major units – Layer Norm, Multi-head Attention Network (MSP) and Multi-Layer Perceptron (MLP).

Layer Norm is used to keep the training process on track and helps the model to adapt to the diverse varied training images. MSP is a network responsible for generating the attention maps from the input embeddings. MLP is a two-layer classification network at the end. The

MLP head is used as output of the transformer block.

With ever increasing data and development of various models, it is important that the architectures are more efficiently trained. ViT is one such model which has several prominent applications such as image classification, image captioning, segmentation, anomaly detection and countless others.

APPENDIX 3: USE CASE – Anomaly Detection in medical images

There is great potential for improving patient outcomes developing diagnostic capabilities and revolutionizing medical research through the use of transformer-based models and generative adversarial networks (GANs) in medical imaging. Specifically focusing on anomaly detection and image synthesis tasks we examine a compelling use case of our project in the field of medical image analysis in this section.

Diagnostic tools for a wide range of illnesses and ailments include CT (Computed Tomography) MRI (Magnetic Resonance Imaging) ultrasound and X-rays. But deciphering these images can be difficult especially when trying to find subtle abnormalities or anomalies that might point to underlying health problems. An effective way to overcome this difficulty is to combine transformer-based models and GANs for anomaly detection in medical images. Healthcare providers can improve patient care and their diagnostic abilities by utilizing these models' capacity to learn intricate patterns and produce lifelike images.

GANs can be trained on a dataset of both normal and abnormal MRI images for instance in the context of MRI tumor detection. While the discriminator discerns between real and synthetic images the generator learns to produce synthetic images that closely resemble actual MRI scans. This allows radiologists or oncologists to identify anomalies or abnormalities that differ from the learned distribution and mark them as possible areas of concern for follow-up work.

Additionally, by improving the resolution and quality of medical images transformer-based models like TransGAN can be used to increase diagnostic accuracy. TransGAN can produce detailed high-quality images from low-resolution inputs by training the model on pairs of low- and high-resolution medical images. When obtaining high-resolution images is difficult or expensive as in remote or resource-constrained healthcare settings this capability is especially helpful.

In addition to anomaly detection, realistic synthesized medical images can be used for research and training purposes. Synthetic data can be used to augment existing datasets thereby making the dataset more diverse and varied. This approach helps improve the robustness and generalizability of diagnosis.

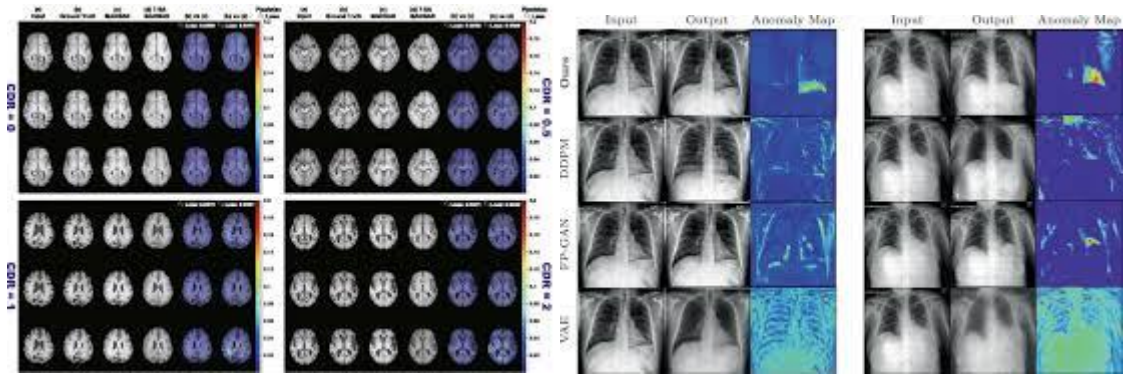

Figure 25: Anomaly Detection for MRI and X-Ray Scan

APPENDIX 4: Ethical Considerations and Implications

1. Data Privacy and Consent:

   Concerns regarding data privacy and consent are brought up by the generation and analysis of data using transformer-based models and GANs. Researchers must get informed consent from people whose data is used to train models and make sure that privacy-preserving measures are taken to safeguard sensitive data due to ethical considerations. Furthermore, steps must be taken to reduce the possibility of re-identification and illegal access to personal information.

2. Bias and Fairness:

   An unfair or discriminatory result may result from biases in training data spreading through GANs and transformer-based models. In order to ensure fairness and equity researchers must actively identify mitigate and transparently disclose biases in model training data. This is required by ethical considerations. Furthermore, to lessen algorithmic bias attempts should be made to create inclusive diverse datasets that faithfully capture the range of human diversity.

3. Accountability and Transparency:

   Accountability and comprehension of AI models decision-making processes may be hampered by their opacity especially when it comes to GANs. Researchers should

emphasize openness in the creation use and results of their models giving concise justifications for the actions and choices made by the models as required by ethical principles. In order to promote accountability and trust in AI systems procedures for auditing and informing stakeholders about model predictions should also be put in place.

In conclusion even though transformer-based models and GANs have the potential to revolutionize a variety of fields it is crucial to discuss the ethical ramifications and issues in order to make sure that these technologies are created and applied appropriately. We can harness the power of AI for the good of humanity while respecting moral standards and values by putting data privacy and consent first reducing bias encouraging transparency and accountability taking societal impact into account and being cautious about misuse.

APPENDIX 5: Code Snippets

1. TransGAN model:

```
class MLP(nn.Module):
    def __init__(self, in_feat, hid_feat=None, out_feat=None,
                 dropout=0.):
        super().__init__()
        if not hid_feat:
            hid_feat = in_feat
        if not out_feat:
            out_feat = in_feat
        self.fc1 = nn.Linear(in_feat, hid_feat)
        self.act = nn.GELU()
        self.fc2 = nn.Linear(hid_feat, out_feat)
        self.droprateout = nn.Dropout(dropout)

    def forward(self, x):
        x = self.fc1(x)
        x = self.act(x)
        x = self.fc2(x)
        return self.droprateout(x)
```

```python
class Attention(nn.Module):
    def __init__(self, dim, heads=4, attention_dropout=0., proj_dropout=0.):
        super().__init__()
        self.heads = heads
        self.scale = 1./dim**0.5

        self.qkv = nn.Linear(dim, dim*3, bias=False)
        self.attention_dropout = nn.Dropout(attention_dropout)
        self.out = nn.Sequential(
            nn.Linear(dim, dim),
            nn.Dropout(proj_dropout)
        )

    def forward(self, x):
        b, n, c = x.shape
        qkv = self.qkv(x).reshape(b, n, 3, self.heads, c//self.heads)
        q, k, v = qkv.permute(2, 0, 3, 1, 4)

        dot = (q @ k.transpose(-2, -1)) * self.scale
        attn = dot.softmax(dim=-1)
        attn = self.attention_dropout(attn)

        x = (attn @ v).transpose(1, 2).reshape(b, n, c)
        x = self.out(x)
        return x

class ImgPatches(nn.Module):
    def __init__(self, input_channel=3, dim=768, patch_size=4):
        super().__init__()
        self.patch_embed = nn.Conv2d(input_channel, dim,
                                     kernel_size=patch_size, stride=patch_size)

    def forward(self, img):
        patches = self.patch_embed(img).flatten(2).transpose(1, 2)
        return patches
def UpSampling(x, H, W):
        B, N, C = x.size()
        assert N == H*W
        x = x.permute(0, 2, 1)
        x = x.view(-1, C, H, W)
        x = nn.PixelShuffle(2)(x)
        B, C, H, W = x.size()
        x = x.view(-1, C, H*W)
        x = x.permute(0,2,1)
        return x, H, W

class Encoder_Block(nn.Module):
    def __init__(self, dim, heads, mlp_ratio=4, drop_rate=0.):
        super().__init__()
        self.ln1 = nn.LayerNorm(dim)
        self.attn = Attention(dim, heads, drop_rate, drop_rate)
        self.ln2 = nn.LayerNorm(dim)
        self.mlp = MLP(dim, dim*mlp_ratio, dropout=drop_rate)
    def forward(self, x):
        x1 = self.ln1(x)
        x = x + self.attn(x1)
        x2 = self.ln2(x)
        x = x + self.mlp(x2)
        return x
```

```python
class TransformerEncoder(nn.Module):
    def __init__(self, depth, dim, heads, mlp_ratio=4, drop_rate=0.):
        super().__init__()
        self.Encoder_Blocks = nn.ModuleList([
            Encoder_Block(dim, heads, mlp_ratio, drop_rate)
            for i in range(depth)])
    def forward(self, x):
        for Encoder_Block in self.Encoder_Blocks:
            x = Encoder_Block(x)
        return x
```

```python
class Generator(nn.Module):
    """docstring for Generator"""
    def __init__(self, depth1=5, depth2=4, depth3=2, initial_size=8, dim=384, heads=4, mlp_ratio=4, drop_rate=0.
        super(Generator, self).__init__()
        self.initial_size = initial_size
        self.dim = dim
        self.depth1 = depth1
        self.depth2 = depth2
        self.depth3 = depth3
        self.heads = heads
        self.mlp_ratio = mlp_ratio
        self.droprate_rate =drop_rate

        self.mlp = nn.Linear(1024, (self.initial_size ** 2) * self.dim)
        self.positional_embedding_1 = nn.Parameter(torch.zeros(1, (8**2), 384))
        self.positional_embedding_2 = nn.Parameter(torch.zeros(1, (8*2)**2, 384//4))
        self.positional_embedding_3 = nn.Parameter(torch.zeros(1, (8*4)**2, 384//16))
        self.TransformerEncoder_encoder1 = TransformerEncoder(depth=self.depth1, dim=self.dim,heads=self.heads,
        self.TransformerEncoder_encoder2 = TransformerEncoder(depth=self.depth2, dim=self.dim//4, heads=self.hea
        self.TransformerEncoder_encoder3 = TransformerEncoder(depth=self.depth3, dim=self.dim//16, heads=self.he

        self.linear = nn.Sequential(nn.Conv2d(self.dim//16, 3, 1, 1, 0))

    def forward(self, noise):

        x = self.mlp(noise).view(-1, self.initial_size ** 2, self.dim)

        x = x + self.positional_embedding_1
        H, W = self.initial_size, self.initial_size
        x = self.TransformerEncoder_encoder1(x)

        x,H,W = UpSampling(x,H,W)
        x = x + self.positional_embedding_2
        x = self.TransformerEncoder_encoder2(x)

        x,H,W = UpSampling(x,H,W)
        x = x + self.positional_embedding_3

        x = self.TransformerEncoder_encoder3(x)
        x = self.linear(x.permute(0, 2, 1).view(-1, self.dim//16, H, W))

        return x
```

```python
class Discriminator(nn.Module):
    def __init__(self, diff_aug, image_size=32, patch_size=4, input_channel=3, num_classes=1,
                 dim=384, depth=7, heads=4, mlp_ratio=4,
                 drop_rate=0.):
        super().__init__()
        if image_size % patch_size != 0:
            raise ValueError('Image size must be divisible by patch size.')
        num_patches = (image_size//patch_size) ** 2
        self.diff_aug = diff_aug
        self.patch_size = patch_size
        self.depth = depth
        # Image patches and embedding layer
        self.patches = ImgPatches(input_channel, dim, self.patch_size)

        # Embedding for patch position and class
        self.positional_embedding = nn.Parameter(torch.zeros(1, num_patches+1, dim))
        self.class_embedding = nn.Parameter(torch.zeros(1, 1, dim))
        nn.init.trunc_normal_(self.positional_embedding, std=0.2)
        nn.init.trunc_normal_(self.class_embedding, std=0.2)

        self.droprate = nn.Dropout(p=drop_rate)
        self.TransfomerEncoder = TransformerEncoder(depth, dim, heads,
                                                    mlp_ratio, drop_rate)
        self.norm = nn.LayerNorm(dim)
        self.out = nn.Linear(dim, num_classes)
        self.apply(self._init_weights)

    def _init_weights(self, m):
        if isinstance(m, nn.Linear):
            nn.init.trunc_normal_(m.weight, std=.02)
            if isinstance(m, nn.Linear) and m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.LayerNorm):
            nn.init.constant_(m.bias, 0)
            nn.init.constant_(m.weight, 1.0)
```

```python
    def forward(self, x):
        x = DiffAugment(x, self.diff_aug)
        b = x.shape[0]
        cls_token = self.class_embedding.expand(b, -1, -1)

        x = self.patches(x)
        x = torch.cat((cls_token, x), dim=1)
        x += self.positional_embedding
        x = self.droprate(x)
        x = self.TransfomerEncoder(x)
        x = self.norm(x)
        x = self.out(x[:, 0])
        return x
```

2. ImageMapper Module

```
# Define loss function
criterion = nn.MSELoss()

# Load pre-trained transformer (e.g., ResNet50)
feature_extractor = resnet50(pretrained=True)

# Freeze transformer parameters
for param in feature_extractor.parameters():
    param.requires_grad = True

# Define mapping function
class ImageToNoiseMapper(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(ImageToNoiseMapper, self).__init__()
        self.linear = nn.Linear(input_dim, output_dim)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.linear(x)
        x = self.relu(x)
        return x

# Define optimizer for mapper
optimizer = optim.Adam(feature_extractor.parameters(), lr=0.001)

# Define transformations for CIFAR-10 images
transform = transforms.Compose([
    transforms.Resize(size=(32, 32)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
# Define CIFAR-10 train dataset and data loader
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=50, shuffle=True)
losses = []
# Define device (CPU or GPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
num_epochs = 100
# Training loop
# b = len(trainloader)
```

```
for epoch in range(num_epochs):
    j = 0
    for input_image, _ in trainloader:
        print(input_image.shape)
        j += 1
        # Move input image to device
        input_image = input_image
        # Forward pass through feature extractor
        features = feature_extractor(input_image)

        batch_size, features_dim = features.shape[:2]  # Extract batch size and features dimension
        flattened_features = features.view(batch_size, -1)
        # Forward pass through generator
        output_dim = 1024  # Assuming noise vector size
        mapper = ImageToNoiseMapper(features_dim, output_dim)
        noise_vector = mapper(flattened_features)
        generated_image = G(noise_vector.to(device))

        # Compute loss between generated image and input image
        loss = criterion(generated_image.to('cpu'), input_image.to('cpu'))
        loss.backward()
        optimizer.step()
        # Backpropagation
        optimizer.zero_grad()

        losses.append(loss.item())
```

3. Anomaly Detection Architecture using 2 discriminators

```
class GAN_SOTA(pl.LightningModule):
    def __init__(self, lr=0.0002, display=False, display_end=False, b1=0.5, b2=0.999):
        super().__init__()
        self.lr = lr
        self.display = display
        self.display_end = display_end

        self.save_hyperparameters()
        self.G = G
        self.Enc = ENCODER()
        self.D_norm = Discriminator(diff_aug = "translation,cutout,color", image_size=32, patch_size=4, input_ch
                dim=384, depth=7, heads=4, mlp_ratio=4,
                drop_rate=0.)
        self.D_anom = Discriminator(diff_aug = "translation,cutout,color", image_size=32, patch_size=4, input_ch
                dim=384, depth=7, heads=4, mlp_ratio=4,
                drop_rate=0.)
        self.D_anom.load_state_dict(checkpoint['discriminator_state_dict'])
        self.D_norm.load_state_dict(checkpoint['discriminator_state_dict'])
        self.G.load_state_dict(checkpoint['generator_state_dict'])
        self.automatic_optimization = False
        self.test_step_outputs = [[], []]  # two dataloaders
        self.auroc = 0
        self.last_batch = None
        wandb.watch(self.G)
        wandb.watch(self.D_norm)
        wandb.watch(self.D_anom)

    def forward(self, x):
        return self.G(ImgNoise(x))

    def print_generator_parameters(self):
        for name, param in self.G.named_parameters():
            print(f"Generator Parameter Name: {name}")
            print(f"Parameter Value:\n{param}")
```

```
    def patch_loss(self, X, generated_X, n=3):

        # Calculate the L1 loss for each patch
        patch_errors = torch.abs(X - generated_X)
        patch_errors_reshaped = patch_errors.view(patch_errors.size(0), patch_errors.size(1), -1)

        # compute loss for each patch
        mean_patch_errors = patch_errors_reshaped.mean(dim=-1)

        # Select the top n patches
        sorted_patch_indices = torch.argsort(mean_patch_errors, dim=-1, descending=True)
        top_patch_indices = sorted_patch_indices[:, :n]

        # Calculate the average of the top n patch errors
        top_patch_errors = torch.gather(mean_patch_errors, dim=-1, index=top_patch_indices)
        avg_top_patch_errors = top_patch_errors.mean()
        return avg_top_patch_errors

    # Utilities to print batches during training
    def show_batch_image(self, batch):
        X, Y = batch
        image = X[0].detach().cpu().numpy()
        plt.figure(figsize=(5, 5))
        plt.title("Label: {}".format(Y[0]))
        plt.imshow(image[0], cmap='gray')
        plt.axis('off')
        plt.show()
```

```
    #The generator is trained to output 1 from normal data and 0 from anomaly data
    def training_step(self, batch, batch_idx):
#         g_opt, d_norm_opt, d_anom_opt = self.optimizers()

        X, Y = batch
        self.last_batch = batch
        batch_size = X.shape[0]

        real_label = torch.ones((batch_size, 1), device=self.device)
        fake_label = torch.zeros((batch_size, 1), device=self.device)

        errD_anomal = 0
        errD_normal = 0

        # Patch embedding
        patch_size = 4
        embed_dim = 384
#         patch_embedding = PatchEmbedding(patch_size, 1, embed_dim)

        x = ImgNoise(X)

        # Normal Case
        if (Y.squeeze() == 1):
            x = ImgNoise(X)
            generated_X = self.G(x)

            # print("---\t", X.shape, generated_X.shape)
            # print("\n")
            discriminated_X = self.D_norm(X)
            discriminated_G = self.D_norm(generated_X)
            encoded_X = self.Enc(X)
            # Optimize Discriminator #
            d_norm_opt.zero_grad()
```

```
#             print(discriminated_G.shape, discriminated_X.shape,fake_label.shape, real_label.shape)
            # NORM ADV LOSS
            Norm_adv_loss = (((discriminated_X - real_label)**2) + ((discriminated_G - fake_label)**2))

            errD_normal = (Norm_adv_loss.mean())

            errD = errD_normal
            self.manual_backward(errD_normal.mean())
            d_norm_opt.step()

            # Optimize Generator #
            g_opt.zero_grad()
            x = ImgNoise(X)
            generated_X = self.G(x)
            # print("---\t", X.shape, generated_X.shape)
            discriminated_X = self.D_norm(X)
            discriminated_G = self.D_norm(generated_X)
            encoded_X = self.Enc(generated_X)

            # L1 RECONSTRUCTION ERROR
            l1_loss = F.l1_loss(X, generated_X)

            # PATCH L1 LOSS
            patch_loss = self.patch_loss(X, generated_X)

            # LATENT VECTOR LOSS
            latent_loss = F.l1_loss(encoded_X, self.Enc(generated_X))

            # ADVERSARIAL LOSS
            norm_adv_loss = ((discriminated_G - real_label)**2)

            errG_normal = patch_loss*(1.5) + (norm_adv_loss)*(0.5) + (latent_loss)*(0.5) + (l1_loss)*(1.5)
            errG = errG_normal.mean()
            self.manual_backward(errG_normal.mean())
            g_opt.step()
            wandb.log({"g_loss": torch.tensor(errG).mean(), "d_normal_loss": torch.tensor(errD).mean()})
            self.log_dict({"g_loss": torch.tensor(errG).mean(), "d_normal_loss": torch.tensor(errD).mean()}, pro

        # Anomal Case
        else:
            x = ImgNoise(X)
            generated_X = self.G(x)
            # print("---\t", X.shape, generated_X.shape)
            # print("\n")
            discriminated_X = self.D_anom(X)
            discriminated_G = self.D_anom(generated_X)

            # Optimize Discriminator #
            d_anom_opt.zero_grad()

            # ANOM ADV LOSS
#             print(discriminated_G.shape, discriminated_X.shape,fake_label.shape, real_label.shape )
            Anom_adv_loss = ((((discriminated_X - real_label)**2) + ((discriminated_G - fake_label)**2)))

            errD_anomal = (Anom_adv_loss.mean())

            errD = errD_anomal
            self.manual_backward(errD_anomal.mean())
            d_anom_opt.step()
```

```
            # Optimize Generator #
            g_opt.zero_grad()
            x = ImgNoise(X)
            generated_X = self.G(x)
            discriminated_X = self.D_anom(X)
            discriminated_G = self.D_anom(generated_X)
            encoded_X = self.Enc(X)

#             print(discriminated_G.shape, discriminated_X.shape,fake_label.shape, real_label.shape )
            # ANOM ADVERS LOSS
            anom_adv_loss = ((discriminated_G - fake_label)**2)

            # ABC LOSS
            abc_loss = -torch.log(1 - torch.exp(-F.l1_loss(generated_X, X)))

            errG_anomal = anom_adv_loss*(1) + (abc_loss)*(0.5)
            errG = errG_anomal.mean()
            self.manual_backward(errG_anomal.mean())
            g_opt.step()

            self.log_dict({"g_loss": errG, "d_anomal_loss": errD}, prog_bar=True)
            wandb.log({"g_loss": errG, "d_anomal_loss": errD})
            wandb.log({"input_images": [wandb.Image(X[i]) for i in range(batch_size)]})
            wandb.log({"generated_images": [wandb.Image(to_pil_image(generated_X[i].cpu())) for i in range(batc

        return {"g_loss": errG, "d_norm_loss": errD_normal, "d_anom_loss": errD_anomal}
```

```
    def test_step(self, batch, batch_idx):

        X, Y = batch
        x = ImgNoise(X).to("cuda")
        generated_X = self.G(x)

        # Calculate reconstruction error
        reconstruction_error = F.mse_loss(generated_X, X, reduction='none')
        reconstruction_error = reconstruction_error.view(reconstruction_error.size(0), -1).mean(dim=1)
        self.test_step_outputs[0].append(reconstruction_error)
        self.test_step_outputs[1].append(Y)

        if self.display:
            print("generated_X:", generated_X)
            print("reconstruction_error:", reconstruction_error)
            self.show_batch_image(batch)

        wandb.log({"reconstruction_error": reconstruction_error})
        return {"reconstruction_error": reconstruction_error, "true_labels": Y}


    def on_test_epoch_end(self):
        all_reconstruction_errors = torch.cat(self.test_step_outputs[0])
        all_true_labels = torch.cat(self.test_step_outputs[1])

        anomaly_threshold = 0.5
        predicted_labels = (all_reconstruction_errors < anomaly_threshold).float().cpu().numpy()
        true_labels = all_true_labels.squeeze().float().cpu().numpy()

        # Calculate AUROC
        auroc = roc_auc_score(true_labels, predicted_labels)

        self.log("auroc", (auroc), prog_bar=True)
        self.auroc = auroc
```

4. Anomaly Detection using latent walk

```
def train(noise, generator, discriminator, optim_gen, optim_dis,
          epoch, writer, schedulers, img_size=image_size, latent_dim=latent_dim,
          n_critic=n_critic,
          gener_batch_size=gener_batch_size, device=device):

    writer = writer_dict['writer']
    gen_step = 0

    generator = generator.train()
    discriminator = discriminator.train()

    transform = transforms.Compose([transforms.Resize(size=(img_size, img_size)), transforms.RandomHorizontalFlip(
                                    transforms.ToTensor(),
                                    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

    train_loader = data_loader

    for index, img in enumerate(train_loader):

        global_steps = writer_dict['train_global_steps']

        real_imgs = img.type(torch.cuda.FloatTensor)

        noise = torch.cuda.FloatTensor(np.random.normal(0, 1, (img.shape[0], latent_dim)))

        optim_dis.zero_grad()
        real_valid = discriminator(real_imgs)
        fake_imgs = generator(noise).detach()

        fake_valid = discriminator(fake_imgs)
```

```
        if loss_type == 'hinge':
            loss_dis = torch.mean(nn.ReLU(inplace=True)(1.0 - real_valid)).to(device) + torch.mean(
                nn.ReLU(inplace=True)(1 + fake_valid)).to(device)
        elif loss_type == 'wgangp_eps':
            gradient_penalty = compute_gradient_penalty(discriminator, real_imgs, fake_imgs.detach(), phi)
            loss_dis = -torch.mean(real_valid) + torch.mean(fake_valid) + gradient_penalty * 10 / (phi ** 2)

        loss_dis.backward(); losses[0].append(loss_dis.item())
        optim_dis.step()

        writer.add_scalar("loss_dis", loss_dis.item(), global_steps)

        if global_steps % n_critic == 0:

            optim_gen.zero_grad()
            if schedulers:
                gen_scheduler, dis_scheduler = schedulers
                g_lr = gen_scheduler.step(global_steps)
                d_lr = dis_scheduler.step(global_steps)
                writer.add_scalar('LR/g_lr', g_lr, global_steps)
                writer.add_scalar('LR/d_lr', d_lr, global_steps)

            gener_noise = torch.cuda.FloatTensor(np.random.normal(0, 1, (gener_batch_size, latent_dim)))

            generated_imgs = generator(gener_noise)
            fake_valid = discriminator(generated_imgs)

            gener_loss = -torch.mean(fake_valid).to(device)
            gener_loss.backward(); losses[1].append(gener_loss.item())
            optim_gen.step()
            writer.add_scalar("gener_loss", gener_loss.item(), global_steps)

            gen_step += 1
```

```python
import torch.optim as optim
from tqdm import tqdm

def walk_latent_space(G, D, im_tensor, n_iter=1500, lambd=0.001, lr=2e-2, device='cuda'):
    try:
        f_x = {}  # Precompute feature values for all layers of D
        def hook(name):
            def hook_fn(module, input, output):
#                 if name not in f_x:
                    f_x[name] = output
            return hook_fn

        # Register hooks to the first 10 layers
        j = 0
        for name, module in D.named_children():
            if  j < 10:
                module.register_forward_hook(hook(name))
                j += 1

        D.eval()
        _ = D(im_tensor.unsqueeze(0).to('cuda'))

        d_x = {}

        for i in f_x.keys():
            d_x[i] = f_x[i].detach()

        z = torch.randn(1, 1024, device='cuda', dtype=torch.float, requires_grad=True)
        opt=optim.Adam([z], lr= lr)

        losses = []
        loss = 0
        for i in tqdm(range(n_iter)):
            fake = G(z)
            loss_r = combined_l1_l2_loss(fake.squeeze(0).to('cpu'), im_tensor)  # Residual loss
            loss_d = get_d_loss(d_x, fake, D)  # Discriminator loss
            loss = (1 - lambd) * loss_r + lambd * loss_d  # Total loss

            loss.backward()
            torch.nn.utils.clip_grad_norm_([z], max_norm=1.0)

#             z.grad.zero_()
            opt.step()

            # Zero gradients
            opt.zero_grad()
            G.zero_grad()
            D.zero_grad()

            losses.append(loss.item())
    except RuntimeError as e:
            # If other runtime error, raise it
            raise e
    return {'z': z, 'loss': loss.item(), 'loss_r': loss_r.item(), 'loss_d': loss_d.item(), 'losses': losses}
```

```python
def anomaly_detector(im, G, D, d_loss_thresh, r_loss_thresh):
    '''Decides if image im is anomalous'''
    im_tensor = toTens(im)
    res=walk_latent_space(G, D, im_tensor, n_iter=1000, lambd=0.1, lr=2e-2, device='cuda')
    if res['loss_d']>d_loss_thresh or res['loss_r']>r_loss_thresh: return True
    else: return False
```

REFERENCES

[1]. Ghazal, M., Vázquez, C., & Amer, A. (2007). Real-time automatic detection of vandalism behavior in video sequences. 2007 IEEE International Conference on Systems, Man and Cybernetics, 1056-1060.

[2]. Mould, N., Regens, J.L., Jensen, C., & Edger, D.N. (2014). Video surveillance and counterterrorism: the application of suspicious activity recognition in visual surveillance systems to counterterrorism. Journal of Policing, Intelligence and Counter Terrorism, 9, 151 - 175.

[3]. Jiang, Y., Chang, S., & Wang, Z. (2021). TransGAN: Two Transformers Can Make One Strong GAN. ArXiv, abs/2102.07074.

[4]. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., & Bengio, Y. (2014). Generative Adversarial Nets. Neural Information Processing Systems.

[5]. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U.M., & Langs, G. (2017). Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. Information Processing in Medical Imaging.

[6]. Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2019). MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 9584-9592.

[7]. Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images.

[8]. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[9]. Kwon, D., Natarajan, K., Suh, S.C., Kim, H., & Kim, J. (2018). An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 1595-1598.

[10]. Sabokrou, M., Fayyaz, M., Fathy, M., Moayed, Z., & Klette, R. (2016). Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. Comput. Vis. Image Underst., 172, 88-97.

[11]. Pang, G., Shen, C., Cao, L., & Hengel, A.V. (2020). Deep Learning for Anomaly Detection. ACM Computing Surveys (CSUR), 54, 1 - 38.

[12]. Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., & Batra, D. (2016). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. International Journal of Computer Vision, 128, 336 - 359.

[13]. Chalapathy, R., & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. ArXiv, abs/1901.03407.

[14]. Bulusu, S., Kailkhura, B., Li, B., Varshney, P.K., & Song, D.X. (2020). Anomalous Example Detection in Deep Learning: A Survey. IEEE Access, 8, 132330-132347.

[15]. Sabokrou, M., Khalooei, M., Fathy, M., & Adeli, E. (2018). Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3379-3388)

[16]. Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019, May). Self-attention generative adversarial networks. In *International conference on machine learning* (pp. 7354-7363). PMLR.

[17]. Shangguan, W., Fan, W., & Chen, Z. (2022, March). Semi-supervised anomaly detection based on deep generative models with transformer. In *Proceedings of the 2022 6th International Conference on Innovation in Artificial Intelligence* (pp. 172-177).

[18]. Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K. R., & Kloft, M. (2019). Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*.

[19]. Yang, C., Lan, S., Huang, W., Wang, W., Liu, G., Yang, H., ... & Li, P. (2022, September). A transformer-based gan for anomaly detection. In *International Conference on Artificial Neural Networks* (pp. 345-357). Cham: Springer Nature Switzerland.

[20]. Zhang, L., Dai, Y., Fan, F., & He, C. (2022). Anomaly Detection of GAN Industrial Image Based on Attention Feature Fusion. *Sensors*, *23*(1), 355.

[21]. Feng, X., Song, D., Chen, Y., Chen, Z., Ni, J., & Chen, H. (2021, October). Convolutional transformer based dual discriminator generative adversarial networks for video anomaly detection. In *Proceedings of the 29th ACM International Conference on Multimedia* (pp. 5546-5554).

[22]. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[23]. Kim, J., Jeong, K., Choi, H., & Seo, K. (2020). GAN-Based Anomaly Detection In Imbalance Problems. ECCV Workshops.

[24]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems (p./pp. 5998--6008).

[25]. Ba, J., Kiros, J., & Hinton, G. Layer Normalization.

[26]. Akcay, S., Atapour-Abarghouei, A., Breckon, T.P.(2018) Ganomaly: Semi-supervisedanomaly detection via adversarial training. In: Asian conference on computer vision. pp. 622–637. Springer.

[27]. Mao, X., Li, Q., Xie, H., Lau, R., Zhen, W., & Smolley, S.. (2017). Least Squares Generative Adversarial Networks. 2813–2821