B.Tech Project Report

on

Design and Analysis of GAN Architecture for Anomaly Detection

By

Aditi, 180001002

Jay Bangar, 180001022

Yanamadala Aravind, 180001063

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

Design and Analysis of GAN Architecture for Anomaly Detection

A Project Report

Submitted in partial fulfilment of
the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Aditi, 180001002
Jay Bangar, 180001022
Yanamadala Aravind, 180001063

Guided by:

Dr. Aruna Tiwari
Professor
Computer Science and Engineering
IIT Indore

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"Design and Analysis of GAN Architecture for Anomaly Detection"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science and Engineering' completed under the supervision of **Dr. Aruna Tiwari, Professor, Computer Science and Engineering, IIT Indore** is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Aditi (26/5/2022) Jay 26/5/2022 Y. Sai Aravind 26-May-2022

Aditi
Jay Bangar
Yanamadala Aravind

CERTIFICATE BY BTP GUIDE

It is certified that the above statement made by the students is correct to the best of my knowledge.

26/5/2022

Dr. Aruna Tiwari
Professor
Computer Science and Engineering
Indian Institute of Technology Indore

PREFACE

This report on **"Design and Analysis of GAN Architecture for Anomaly Detection"** is prepared under the supervision of **Dr. Aruna Tiwari, Professor, Computer Science and Engineering, Indian Institute of Technology, Indore.**

Through this report, we have tried to provide a detailed description of our approach, design, and implementation of the method to address the problem of Anomaly Detection. We have tried to implement a vanilla GAN as well as a Wasserstein GAN to train our model and detect abnormal events.

Aditi
Jay Bangar
Yanamadala Aravind
B.Tech. IV Year
Computer Science and Engineering
IIT Indore

ACKNOWLEDGEMENTS

We wish to thank Dr. Aruna Tiwari for her kind support and valuable guidance throughout the duration of the project, giving us an opportunity to work at our own pace along our own lines, while providing us with very useful directions whenever necessary. We would also like to thank all the faculty members of the Discipline of Computer Science and Engineering for their invaluable support and constructive feedback during the presentations. We wish to express our sincere gratitude to Mr. Rituraj for his guidance throughout the project and for helping us at every stage.

Finally, we offer our sincere thanks to everyone else who knowingly or unknowingly helped us complete this project.

Aditi
Jay Bangar
Yanamadala Aravind
B.Tech. IV Year
Computer Science and Engineering
IIT Indore

ABSTRACT

In order to maintain public safety, surveillance cameras are progressively being utilized in public spaces like roadways, crossroads, banks, etc. Video footage from these cameras helps avoid criminal or unwanted activities. A large number of people have been employed to monitor the surveillance system in which unexpected events occur once in a while. It also needs manual operations to assess the tape for any unlikely event. This thesis work is meant to provide the initial solution for this use case using machine learning techniques to avoid using human resources in monitoring any anomalous activities in surveillance system recordings.

We use video frames from the past and present to identify anomalous activity and predict unprecedented future events. We model a reconstruction-based, One-Class Classifier(OCC) using the Generative Adversarial Network(GAN) architecture to classify a given dynamic image frame as anomalous or normal. The prediction model, with one generator and two discriminators, is trained with only normal samples and the abnormal or anomalous samples are considered outliers. Thus when the model receives an abnormal sample(outlier), it is expected that the reconstructed result would be poor, helping us in detecting the abnormal frame.

Contents

1 Introduction 1

1.1 Motivation . . . 1

1.2 Objective . . . 3

2 Literature Review 4

2.1 One Class Classification (OCC) . . . 4

2.2 Generative Adversarial Networks . . . 7

2.2.1 Preliminary Concepts . . . 7

2.2.2 Conditional GANs (CGANs) . . . 10

2.2.3 Wasserstein GANs (WGANs) . . . 10

3 Design Of GAN Architecture 13

3.1 Generator Architectures . . . 14

3.1.1 Constrained CNN Encoder-Decoder Generator . . . 15

3.1.2 CNN Encoder-Decoder Generator . . . 17

3.1.3 Constrained CNN Encoder-Decoder Generator with different kernel size and latent space . . . 18

3.2 Discriminator Architectures . . . 19

3.2.1 Discriminator for the constrained Encoder-Decoder Generator . . . 20

3.2.2 Discriminator for the CNN Encoder-Decoder Generator . . . 21

3.2.3 Discriminator for Constrained CNN Encoder-Decoder Generator with different kernel size and latent space . . . 21

3.3 Proposed Adversarial Training for the GAN . . . 22

4 Experimentation Details 27

4.1 System Specifications . . . 27

4.2 Dataset Description . . . 27

4.2.1 UCSD Peds Dataset . . . 28

4.2.2 CUHK Avenue Dataset . . . 29

4.2.3 Data Pre-Processing 30
4.3 Training of proposed GAN based OCC 31
4.4 Testing of proposed GAN based OCC 32

5 Results and Discussion **36**
5.1 Results 36
5.1.1 Results with constrained encoder-decoder Generator and the corresponding Discriminator 36
5.1.2 Results with CNN Encoder-Decoder Generator and the corresponding Discriminator 37
5.1.3 Results with constrained encoder-decoder Generator from section 3.1.3 and the corresponding Discriminator 37
5.2 Discussion 38

6 Conclusion and Future Work **39**

Bibliography **41**

Chapter 1

Introduction

Video surveillance is one of the key areas that people have begun to use for security and monitoring purposes all around the world. Keeping a watch on unusual or anomalous activities is a vital responsibility in video monitoring. Since abnormal events are infrequent compared to normal events, a machine learning-based solution to detect them can save a lot of time and manpower. As a result, we propose a Generative Adversarial Network-based One-Class Classifier for anomaly detection in surveillance footage. We further discuss the motivation for the work in the subsequent section followed by the objectives of the work.

1.1 Motivation

Surveillance cameras are becoming more common in public settings such as roadways, junctions, banks, retail malls, private residences, etc. They help in monitoring the activities of both the commercial and residential places so that we shall avoid criminal or unwanted activities. However, law enforcement agencies' monitoring capabilities have not kept up. The number of security cameras compared to humans watching them is quite large, resulting in a significant underutilization of video footage from surveillance cameras and an impractical camera-to-human monitor ratio.

Also, anomalous events occur less frequently as compared to normal ones. As a result, developing sophisticated computer vision algorithms for automatic anomaly identification will save time and money. The term "abnormal" literally means "differing from the norm," notably in a way that is unwanted or unfavorable and goes against the established

rule. Early detection of such events and activities can be extremely beneficial in life-threatening situations. Traffic accidents, crimes, and peculiar actions such as obscuring the cameras are examples of unusual happenings in video surveillance.

There has been a lot of progress in the field of Abnormal Event Detection (AED) based on deep learning techniques, which have a lot of feature extraction capability due to the cascaded and weighted kernel structures of neural networks. These approaches are typically based on autoencoder-inspired reconstruction methods that force the training step to minimize reconstruction errors. These approaches assume that abnormal events are more error-prone than normal events. However, as the well-generalized model can reasonably provide fewer errors than anticipated for anomalous event samples, lowering the reconstruction error can be seen as a disadvantage in detecting an anomaly of events. To address this problem, several studies have begun to use adversarial learning, which involves model training utilizing classification.

In this report, we propose an AED method based on Generative Adversarial Network (GAN) where we construct a One-Class Classifier(OCC) based on reconstruction to classify a dynamic image as abnormal or normal. The suggested method employs adversarial training for past and future events to predict future dynamic images using the present dynamic images. This also improves the robustness of the AED's predictive model without additional information. Using the One-Class-Classifier principle, we train our model using only, normal samples, and as a result, the output will be less accurate when the model is given an anomalous sample as input. We apply this concept to create an anomaly score for a dynamic image and then classify it using that score. The UCSD-Peds dataset and the CUHK Avenue dataset are used to train and evaluate our model.

In this section, we looked at the inspiration and the proposed method for the project, the next section will focus on the aim of the project and how we plan to achieve it.

1.2 Objective

The objective of our B.Tech. project is to build a One-Class Classifier that uses a Generative Adversarial Network (GAN) to classify a dynamic image frame as normal or abnormal. The different sub-objectives to achieve the objective are described below:

a) *Preprocessing videos from dataset to be used as an input for designed models*: This involves generating frames from all the videos of the considered dataset.

b) *Generate Dynamic Images of the extracted frames*: Dynamic images are a summarization of consecutive video frames. To reduce the training over unnecessary frames we pre-process the original video frames and generate a dynamic for an image cube of eight consecutive frames.

c) *Develop an Architecture for the Generator*: We first model an Encoder-Decoder architecture which we use as Generator for our GAN model. The generator takes an image(with noise added) of (160,160,3), encodes it to a latent space, and then decodes it, giving an image of the same size as the input.

d) *Include discriminator(s) to build the GAN Model*: A GAN model has two submodels, Generator and Discriminator. After finalizing the Generator Architecture, discriminator(s) are added to complete our GAN model.

e) *Tune model hyper-parameters*: We change parameters like the number of epochs, learning rate, and batch size to get the best possible results.

f) *Demonstrate the efficiency of our model on different Datasets*: Once the training is done, the model is tested on the UCSD-Peds dataset and CUHK Avenue dataset.

In this chapter, we discussed how deep learning based machine learning models can be useful for anomaly detection in video surveillance and what we aim to accomplish as part of our project. In the next chapter, we will review multiple studies done in the field of anomaly detection as well as the fundamentals of One-Class Classification and Generative Adversarial Networks.

Chapter 2

Literature Review

In this chapter, we discuss some of the other works done in the field of Anomaly Detection. We go through One-Class Classification and discuss its significance. After this, we will understand the basics of Generative Adversarial Networks and its extensions. In the following section, we are going to look at One-Class Classification and some of the related works.

2.1 One Class Classification (OCC)

In computer vision, there are many problems such as outlier detection, and anomaly detection consisting of imbalanced datasets. Generally, the anomalous samples are very infrequent as compared to normal ones. Hence, binary classification is not the best fit for such problems as both classes don't have equal representation. These challenges fall under the umbrella of One-Class Classification in which the goal is to train the model in absence of negative samples.

One of the notable methods for One-Class Classification is the Social Force Model(SVM) proposed by Mehran et al. [1] which uses interaction force between moving objects for the detection and localization of abnormal events in videos. This approach focuses on anomaly detection in crowded videos. To further optimize this approach, they used PSO (Particle Swarm Optimization). One drawback of this approach is that the performance is highly dependent on the hyperparameter setting and it is practically impossible to find optimal parameters that can cover all the real-world variations.

Deep Learning algorithms are also used in several AED(Anomaly Event Detection) methodologies. These approaches exploit the feature extraction capacity with the cascaded and weighted kernel structures of neural networks. Feng et al. [2] have presented one such deep learning based approach for AED. It proposes an unsupervised feature extraction method based using a deep representation-based algorithm. Wo et al. [3] have developed one class neural network DeepOC for anomaly detection. This model simultaneously learns a one-class classifier and compact features using CNN. They have evaluated this model on various datasets like the UCSD dataset, Avenue dataset, etc. The common thing in these deep learning based models is that the target is to minimize the reconstruction error in the training step. They also assumed that abnormal events will produce larger errors than normal events. The results obtained by these approaches have been better than many conventional ones. But still, it is really difficult to construct a generalized model that takes in all the normal scenarios and is strictly discriminative for abnormal events.

To address the above issue, several studies started applying the adversarial learning approaches. Kin et al. [4] proposed AEP(Adversarial Event Prediction) in which abnormal events are detected based on event prediction. They have not used any extra data such as optical flux. It initially derives an event prediction model for the normal events. During testing, it compares the predicted samples with the test samples. As the model has only been trained on normal events, the prediction results will be inaccurate for abnormal events. Having discussed different works on One-Class Classification, let us focus on some of the details of One-Class Classification.

In the Deep End-to-End One-Class Classifier by Sabokrou et al. [5], the model tries to learn the boundary of two classes by learning the representation of only one. The goal is to determine whether the query object belongs to the class on which the model has been trained. Any sample other than the normal is considered an outlier and is expected to be outside of the decision boundary estimated by the model. Efficient data representation is extremely essential for the success of a One-Class Classifier. Statistical modeling and self-representation learning are the two most common approaches for solving OCC problems. In statistical modeling, samples of the target class are mapped to feature space

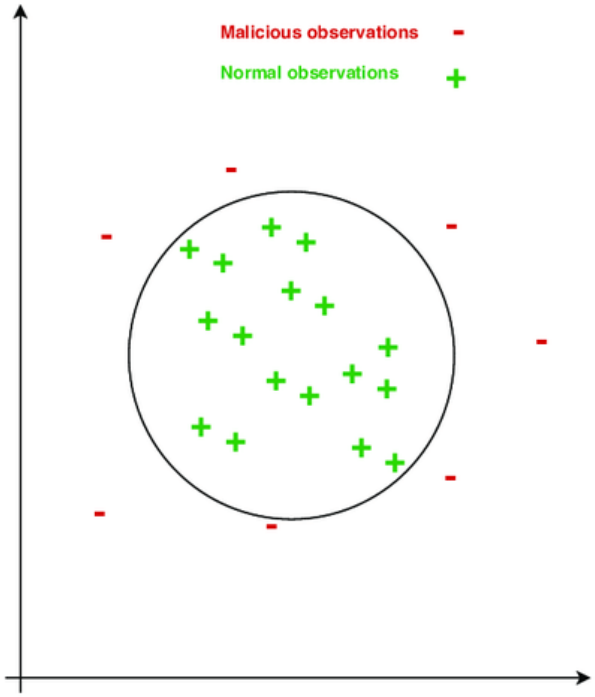


Figure 2.1: One Class Classification

with decreased dimensionality and a maximum likelihood probability distribution is fitted on such represented samples. Samples that do not fit this distribution are defined as outliers. In self-representation learning, the samples are reconstructed on the model trained on only the target class, and based on the error, the sample is labeled as normal or anomalous.

OCC consists of two components: $\mathcal{R}$(Refiner) and $\mathcal{D}$(Detector). The purpose of $\mathcal{R}$ is to reconstruct the input samples and to ensure that generated samples and real samples are indistinguishable. On the contrary, the purpose of $\mathcal{D}$ is to discriminate between real samples and generated samples. These two neural networks compete with each other, hence increasing the efficiency of each other. As the model is being trained only on samples of one class, therefore, when an outlier sample is given to model the reconstructed image by $\mathcal{R}$ will not be as accurate as a normal sample, thus producing a larger reconstruction error(L_R) where

$$L_R = ||X - X'||^2 \tag{2.1}$$

where X is the real input and X' is the reconstructed input.

Let $\mathcal{X}$ be a test sample given to the model. Then, for a certain

predefined threshold t, if $\mathcal{D}(\mathcal{X}) > t$, then $\mathcal{X}$ is classified as anomaly sample else otherwise.

OCC has been found to have numerous advantages in many Deep Learning and Computer Vision problems without which these problems would have been really difficult to solve. In the next section, we will discuss different GANs and their advantages and difficulties faced in them.

2.2 Generative Adversarial Networks

In this section, we will discuss the fundamentals of Generative Adversarial Networks and the problems faced by traditional GANs. We also look at some of the modifications in these GANs which help us in overcoming problems faced in traditional GANs. In the following sub-section, we are going to look at some of the fundamentals of traditional GANs.

2.2.1 Preliminary Concepts

Generative adversarial networks are a type of generative modeling that employs deep learning methods and, generative modeling is an unsupervised learning job, that finds and learns patterns in the input data samples so that the model might be used to generate new samples that appear to be drawn from the original dataset. According to Goodfellow et al. [6], GANs can be used to estimate generative models through an adversarial process in which two models are trained at the same time: a generative model G which captures the data distribution, and a discriminative model D which approximates the likelihood that a sample came from the training data instead of the generative model G. The purpose of G's training is to make D more likely to make a mistake. This is equivalent to a two-player min-max game.

For learning the generator's distribution p_g over data distribution x, we here define an input noise variable as $p_z(z)$, then represent a mapping to data space as $G(z; \theta_g)$, where G is a function that is differentiable and represented by a multi-layer perceptron network with parameters θ_g. A second multi-layer perceptron $D(x; \theta_d)$ is also defined that outputs a single scalar. $D(x)$ estimates the probability that x came from data distribution rather than p_g. D is trained in such a way that it should

optimize the likelihood of correctly labeling both training and generated samples from G and simultaneously G is trained to minimize the value of $\log(1-D(G(z)))$. Here we can say that in a way, D and G are playing the below two-player minimax game with a value function represented by $V(G, D)$:

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z))] \tag{2.2}$$

Below is the algorithm for the learning of generator G and discriminator D:

Algorithm 1 Ian Goodfellow-GAN, k represents no of steps to apply to discriminator, which is taken as a hyperparameter with a vlue of $k = 1$, m is the batch size

for no. of iterations **do**
  for k steps **do**
    Choose $\{z^{(i)}\}_{i=1}^m \sim p_g(z)$ noise samples.
    Choose $\{x^{(i)}\}_{i=1}^m \sim p_{data}$ real data samples.
    Ascend the stochastic gradient of the discriminator to update it:
    $\nabla_{\theta_d} V(G, D) \leftarrow \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(x^i) + \log(1 - D(G(z^i)]$.
  end for
  Choose $\{z^{(i)}\}_{i=1}^m \sim p_g(z)$ noise samples.
  Descend the stochastic gradient of the generator to update it:
  $\nabla_{\theta_g} V(G, D) \leftarrow \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m [\log(1 - D(G(z^i)]$.
end for
Any standard gradient-based learning rule can be used to update the weights.

The creators of LAPGAN(Denton et al.) [7] were motivated to build an alternative way to iteratively upgrade low-resolution produced pictures that can be modeled more consistently after several failed attempts to scale up GANs using CNNs to model images. Attempting to scale GANs using CNN architectures typically employed in the supervised literature also proved challenging. Modifying the below-suggested changes mentioned in the paper by Radford et al. [8] in CNN architectures allowed to obtain stable training and higher resolution for deeper generative models:

1. The All convolutional net (Springenberg et al.) [9] uses strided convolutions to replace spatial pooling layers.
2. To eliminate FCNN layers on top of convolutional features and build a middle ground of directly linking the highest convolutional features to the generator and discriminator's input and output, respectively.

3. To use batch normalization (Ioffe et al.) [10] which stabilizes the process of learning. This helps the networks to converge faster and helps in gradient flow.
4. The generator uses Rectified Linear Unit (Nair et al.) [11] activation except at the output layer, where Tanh activation is employed. This restricted activation aids the model's learning speed.
5. For all the layers in the discriminator Leaky ReLU is used to work effectively, particularly for higher resolution modeling.

The model for DCGAN utilized in the LSUN study is shown below, which takes a 100x1 noise vector, indicated by Z, and translates it into the G(Z) output, which is 64x64x3.

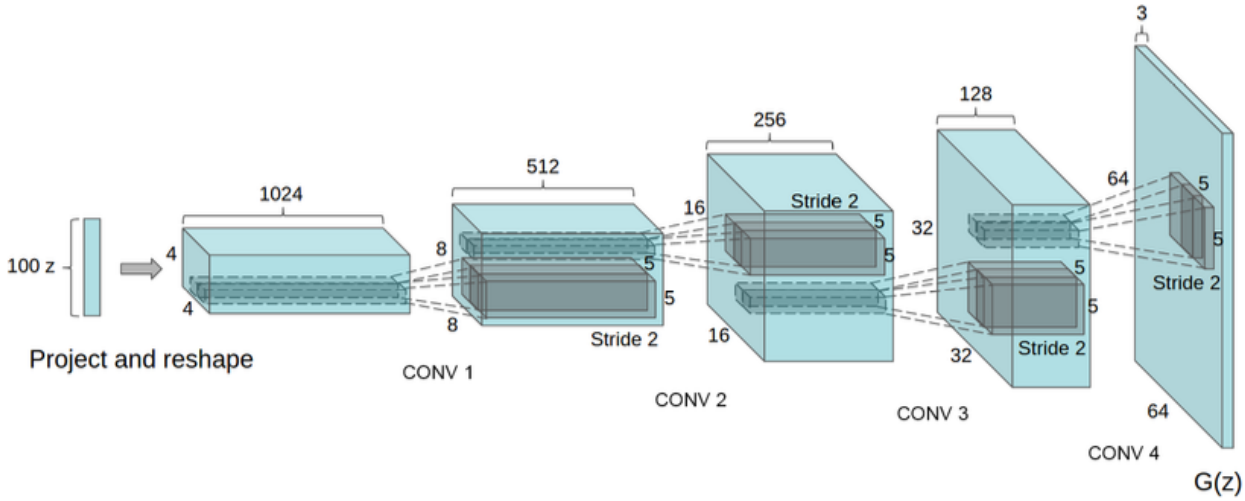


Figure 2.2: Generator using DCGAN architecture employed in LSUN paper [8].

Even though GANs have been applied successfully in many domains and tasks, working with them in practice is challenging because of their unstable optimization procedure, the potential for mode collapse, and difficulty in evaluation. To address these challenges there has been a significant amount of work put into developing new architectures and regularization schemes, one of them is Conditional GANs which will be discussed in the next sub-section.

2.2.2 Conditional GANs (CGANs)

One of the most common problems encountered in the traditional GANs is that the model takes a long time to converge and in many cases, keeps oscillating and never converges. Furthermore, there is no control over the kind of data generated. Since 2014, there have been several modifications to the GAN architecture. One such improvement is CGANs. Conditional GANs are an extension of normal GANs and help us in addressing the above-mentioned problems.

Introduced in 2014 by Mirza et al. [12], they are generative adversarial networks whose Generator and Discriminator are trained by using some additional information. This could be the class of the current image or set of specific characteristics we expect from the output. Adding the information about properties controls the output and directs the Generator to produce the expected output. In Conditional GAN, the job of the discriminator is not only to differentiate between real data and predicted data but also to check if predicted data matches the information provided. The advantage of using Conditional GAN is that convergence will be faster and the output generated will be relevant to the information provided rather than being completely random, Also, the discriminator will be able to distinguish real and generated data better.

Both regular GAN and CGAN have identical loss functions. The min-max loss function can cause the GAN to get stuck in the early stages of GAN training when the discriminator's job is very easy. Also, the issue of vanishing gradients is prominent, where the discriminator is perfect thus making the loss to be zero which results in no gradient to update the weights during the learning of the model. Another downside of CGAN is that they are not completely unsupervised and require labels to function. In the next sub-section, we discuss another improved version of GAN- Wasserstein GAN.

2.2.3 Wasserstein GANs (WGANs)

Problem of Mode Collapse: Ideally, a well-trained generator can generate a wide variety of outputs. Mode Collapse refers to the situation in which regardless of input, the Generator can only produce single output or a small selection of outputs. This could happen due to

various training issues such as the generator discovering the data that is able to fool the discriminator and thus, continue generating that data.

Problem of Vanishing Gradient: Vanishing gradient refers to the situation in which a deep multi-layer feed-forward network is not able to propagate useful gradient information from the output end back to the input end of the model. As a result, the model may not be trained properly and get converged early to a poor solution. The reason for this problem is that as the gradient flows backward, it keeps on getting smaller and smaller. Sometimes, it gets so small that initial layers (near the input end) learn very slowly or may not learn at all. Therefore, weights will not be updated and hence overall training of the model will be stopped.

Both normal GAN and CGAN exhibit the above-mentioned problems. To overcome these issues, Wasserstein GANs were introduced in 2017 in [13]. WGAN provides better stability while training the model. It also provides a better approximation of the distribution of data observed in a given training dataset. Unlike the normal GAN in which we use a discriminator to predict the images as real or fake, WGAN uses the critic that measures the realness or fakeness of the image. The idea behind this change is that the purpose of training the generator should be to minimize the difference between data distributions of training data and generated data.

In Wasserstein GAN, we use Wasserstein loss which is based on Earth-Mover's distance instead of using Binary Cross-Entropy(BCE) loss. The Earth-Mover's distance is a measure of distance between two probability distributions over some region D. The EM(Earth-Mover's) distance is continuous and differentiable which means that the critic can be trained to optimality. If the discriminator does not get stuck on local minima, then it will reject the output that the generator stabilizes upon and hence generator would have to try something new. Therefore, the problem of mode collapse is solved using Wasserstein GAN. As the EM distance is differentiable, therefore the problem of vanishing gradients is also solved. As the critic should satisfy the 1-Lipschitz constraint, we have used weight clipping. The algorithm of WGAN is mentioned below:

Algorithm 2 Wasserstein GAN

Require: α, learning rate; m, batch size; n_{critic}, critic iterations per generator iteration; c, clipping parameter

Require: θ_o: initial values of generator's parameters, w_o: initial values of critic parameters.

for θ has not converged **do**

for t=0, ... n_{critic} **do**

Sample $\{x^{(i)}\}_{i=1}^{m} \sim P_r$ real data batch.

Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ prior samples batch.

$g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$

$w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$

$w \leftarrow \text{clip}(w, -c, c)$

end for

Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ prior samples batch.

$g_\theta \leftarrow \nabla_\theta \left[-\frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$

$\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end for

Although, it has been observed that training is slower in WGAN as compared to normal GAN but still the former is preferred due to its numerous advantages like improved stability, getting rid of problems like mode collapse, and vanishing gradient. In our project, we have made use of both DCGAN and Wasserstein GAN as mentioned in further chapters.

In this chapter, we discussed various concepts like One-Class Classification, GAN, Conditional GAN, Wasserstein GAN, etc. We also reviewed multiple studies done in the field of anomaly detection. In the next chapter, we are going to discuss the design of our proposed GAN-based OCC models. We discuss the architecture of our models in detail and also the theoretical basis of the adversarial training of these models.

Chapter 3

Design Of GAN Architecture

In this chapter, firstly we will discuss different architectures of two sub-models of GAN, the generator and its corresponding discriminator(s). Following that, we will look at how both of the sub-models are trained adversarially as well as how we build a GAN-based OCC to detect anomalous dynamic image frames.

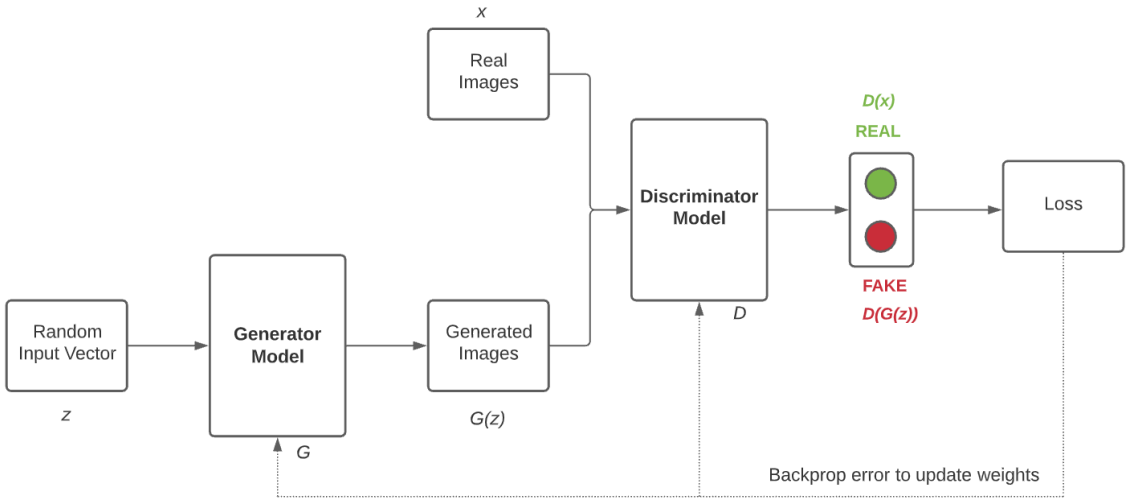


Figure 3.1: Working of GAN

Figure 3.1 shows the basic concept behind the working of GAN, the purpose of the Generator is to generate plausible real samples from a random latent space vector while the discriminator's role is to differentiate between real and generated samples. In the next section, we will discuss constrained as well as unconstrained Generator architecture with varied kernel sizes.

3.1 Generator Architectures

A generator typically takes a normal random vector as an input and outputs a sample in the real domain, but since we are working on OCC based on reconstruction, the input to our Generator is a normal dynamic image and its output is also an image of the same size. Before we begin discussing the Generator architecture, let's take a brief look at CNN.

Convolutional Neural Net or CNN is a class of deep feed-forward artificial neural networks mostly used for the analysis of visual imagery. CNN provides an efficient dense network to detect distinct features from images all by itself, without any human intervention. CNN uses convolution operations with different kernel sizes and the number of filters on images to reduce in a form that is easier to process. There are multiple layers like Convolutional Layer, Pooling Layer, Fully Connected Layer, etc. in a CNN model, each used for a different purpose.

The Generator described by Yu et al. [4] is built with 3-Dimensional CNN and a fully connected Neural Network. The 3D CNNs are used to collect both the spatial and temporal representations of the event samples concurrently. The input to that generator is n consecutive video frames. The Generator is made up of an encoder, followed by a decoder, following the idea of an autoencoder where the first Encoder half, reduces the dimensionalities of an input frame to constrained latent space, and the next Decoder half takes this latent space as input and performs the decoding process (using deConvolution Layers) and restores the old dimensions of the frames.

As we want to process a single dynamic image instead of multiple consecutive frames, we make use of a 2D CNN architecture for our generator. We have worked upon three Generator architectures, each different from the others. In the following subsection, we consider an autoencoder-based generator architecture with a constrained latent space.

3.1.1 Constrained CNN Encoder-Decoder Generator

The architecture for our first generator comes from the paradigm of Auto-Encoders(AEs). In any auto-encoder, the encoder part approximates the input to a latent space Z, and the decoder models the output from this latent space. But in traditional AEs there is no restriction on the manifold Z. Variational Auto-Encoder(VAE) [14] constrains the latent space Z by requiring AE's encoder and decoder to parameterize an approximation to posterior distribution $q(z) \sim \mathcal{N}(z_\mu, z_\sigma)$ in the latent space. A VAE tries to map $q(z)$ to a prior distribution $p(z)$ (usually a normal distribution) by minimizing the Kullback-Leibler(KL) divergence [15] between the two distributions. Thus along with the reconstruction error, we also add this regularisation term to our generator's loss function.

Baur et al. [16] have also used the GAN with VAE to overcome the instability of GAN training and enable faster feed-forward inference, which they successfully exploited for anomaly detection in brain MRI. Sabokrou et al. [5] also use the KL divergence metric to restrict the latent space from the encoder in the generator to a normal distribution to create a deep end-to-end OCC to detect anomalies.

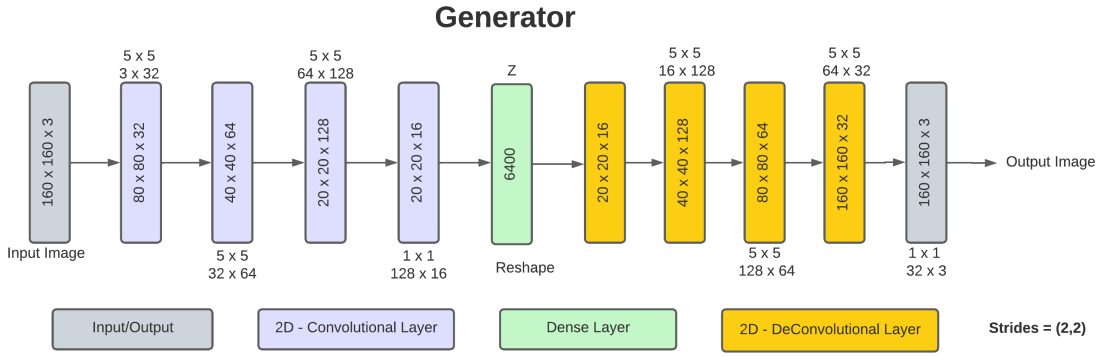


Figure 3.2: Architecture details of constrained Generator $\mathcal{R}$.The kernel dimensions of each layer are shown in the figures above and below the boxes while each layer's input or output is represented by the figures inside each box.

The proposed generator in Figure 3.2 includes multiple convolutional layers as the Encoder, in Purple, followed by a fully connected latent space layer, in Green, and then Deconvolutional layers for the Decoder, in Yellow.

In the first three layers of the encoder of the Generator, the number of filters increases while the input image's width and height reduce by half at each layer, and in the last layer of the encoder, the filters are decreased reducing the number of variables. The output is then flattened and sent as input to the fully connected layer.

The output vector, of size 6400, from the fully connected layer, is reshaped to (20,20,3). After reshaping, in the first layer of the decoder, the number of filters is increased (in response to the decrease of filters in the last layer of the layer). Following that there are three deconvolutional layers where the number of filters decreases to three in the output layer while the image dimensions double at each layer giving the final output from the generator to be the same as that of the input image.

The kernel size and strides at each layer, apart from the last layer of the encoder and decoder, are (5,5) and (2,2) respectively. In the last layers of the encoder and decoder, only the number of filters is changed so the kernel size and strides are set to (1,1). For improving the stability of the network similar to [8], we do not use any pooling layers in this network. After every convolutional and deconvolutional layer, a batch normalization layer is added to regularize the output from the previous layer and minimize the inter-dependency between the layers. After each batch normalization layer, a LeakyReLU activation (with $\alpha = 0.2$) layer is added.

While the LeakyReLU ($y = max(\alpha x, x)$) activation in the last layer gives the output intensities of images in the complete real line, we wanted to experiment with constrained output spaces for the reconstructed image, thus the generator was also modeled with two different activation functions in the last layer of the decoder, namely "Sigmoid Activation" and "Tanh Activation", which confined the individual pixel intensities of the output image to [0,1] and [-1,1] respectively. Thus we get three submodels of the Generator each with different activation functions in the last layer.

The above mentioned are the architectural details for the restricted encoder-decoder Generator based on CNN. An image with added Gaussian noise is fed to the generator as input. The noise is added so that our model becomes robust toward unwanted noise and

corruption. In the following subsection, we discuss an unconstrained encoder-decoder based generator with kernel dimensions different from the generator described above.

3.1.2 CNN Encoder-Decoder Generator

As shown in figure 3.3, the Generator comprises of encoder and decoder. The CNN layers visible in purple makes up the encoder whereas the yellow ones are the decoder layers. The input to the Generator is the image with added Gaussian noise. In the encoder, the number of filters increases with layers, and the size of the input image keeps on reducing by half. In the decoder, the reverse happens. The number of filters keeps decreasing and the size of the input image increases as we progress through various deconvolutional layers. The final output image produced by the generator is of the same dimensions as the input image.

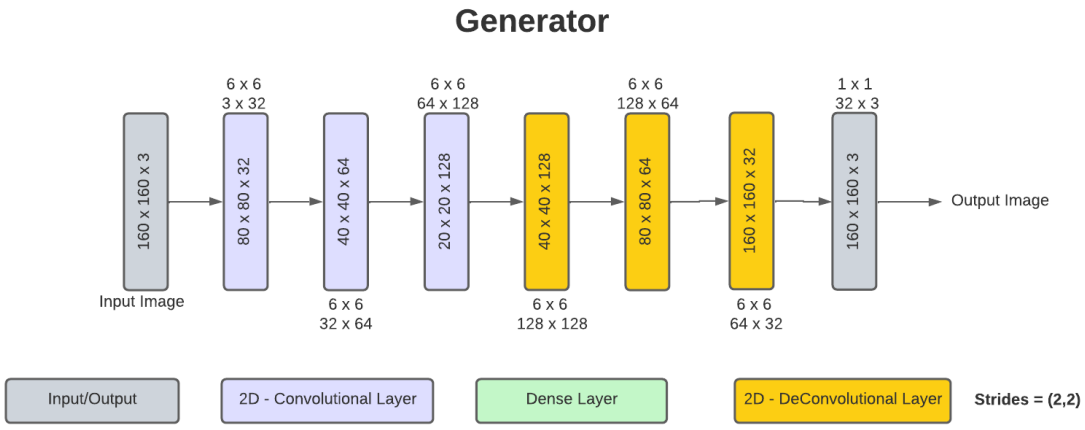


Figure 3.3: Architectural details of the unconstrained encoder-decoder Generator $\mathcal{R}$. The kernel dimensions of each layer are shown in the figures above and below the boxes while each layer's input or output is represented by the figures inside each box.

The kernel size at each layer of the Generator except the last layer is (6,6). We tried a different kernel size like (5,5) in the previous Generator mentioned in 3.1.1. Generally, increasing the kernel size increases the number of parameters which helps the model address more complex problems. However, when we increase the kernel size, it leads to the loss of smaller details or features. Therefore, the idea

behind trying different kernel sizes is to find the optimum value.

After each layer, a batch normalization layer is added to stabilize the learning process. Each batch normalization layer is followed by the LeakyReLU activation layer. The last layer of the decoder has experimented with three different activation functions - "LeakyReLU", "Sigmoid" and "Tanh".

The above details are for the CNN Encoder-Decoder Generator. The major difference between this Generator and the one mentioned in section 3.1.1 is the absence of a dense layer between Encoder and Decoder and the change in kernel size. In the following subsection, we discuss a constrained generator with kernel dimensions different from the generator described above and an additional latent space layer.

3.1.3 Constrained CNN Encoder-Decoder Generator with different kernel size and latent space

The architecture of the Generator $\mathcal{R}$, shown in figure 3.4, is modeled with 2D CNNs which inspired by architecture mentioned in Yu et al. [4], that includes computationally expensive 3D-CNNs that are divided into two halves encoder and decoder. The layers of CNNs in blue makes the encoder that captures the spatial information and the layer in green represents a Fully Connected Neural Network(FCNN) which is used for abstracting the learned representation by CNNs in the encoder. The layers in yellow represent the De-convolutional (DeCNN) layers that make up the decoder. The input image to generator G is added with Gaussian Noise to increase the robustness of the generator. The output of the encoder is fed into latent space (FCNN), which after abstracting the representation is passed to the decoder. As the filters in the encoder increase, the size of the image reduces with each layer whereas the filters in the decoder reduce and the size of the image increases with each layer generating the final output image with the same dimensions as the input image.

This architecture has a kernel size of (4,4) at each layer in the encoder and decoder whereas the previous subsections 3.1.1 and 3.1.2 have a kernel size of (5,5) and (6,6) respectively. Pooling layers are avoided to increase the stability of the network and a normalization layer is used to regularize the output from the previous layer. We also

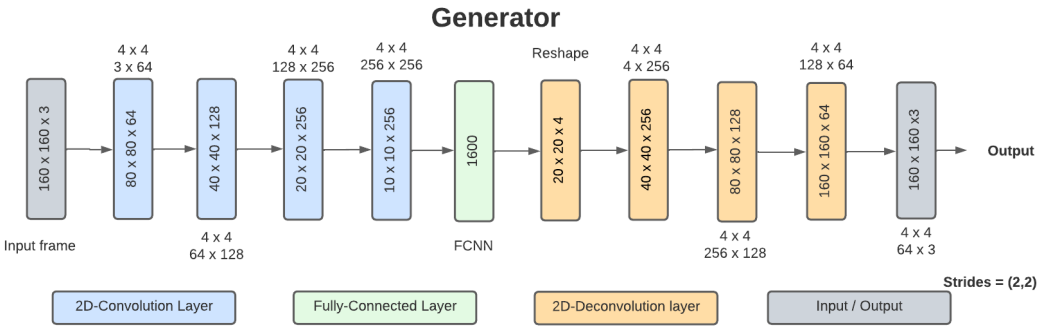


Figure 3.4: Architectural details of restricted CNN Encoder-Decoder Architecture used as Generator $\mathcal{R}$. The kernel dimensions of each layer are shown in the figures above and below the boxes while each layer's input or output is represented by the figures inside each box.

experimented with different activation functions in the final layer of the decoder namely LeakyReLU, Tanh, and Sigmoid.

Until now we have examined three models for the generator of our GAN now, in the next section, we will discuss the second and final component of a GAN, the discriminator. In the following section, we will explore three architectures for the discriminator, one for each of the three generators described in section 3.1.

3.2 Discriminator Architectures

The discriminator in a Deep Convolutional GAN model is a simple classifier that determines whether a given input image is real or generated from the Generator. In our model, we have two discriminators. Past discriminator $\mathcal{D}^{\mathcal{P}}$ and Future discriminator $\mathcal{D}^{\mathcal{F}}$ distinguish whether the dynamic image frame from the generator is the future or past dynamic image frame. The discriminators generate binary values that indicate whether an input frame is future or past and fake or not. For each generator architecture mentioned in Section 3.1, we have a different discriminator architecture. $\mathcal{D}^{\mathcal{F}}$ and $\mathcal{D}^{\mathcal{P}}$ for any single generator are exactly the same in terms of the model architecture, the only difference is their loss function during the training phase.

The architectural details of each of the three discriminators for the corresponding generators are described in the following subsections, first, we look at the discriminator for the constrained generator described in section 3.1.1.

3.2.1 Discriminator for the constrained Encoder-Decoder Generator

First three layers in the discriminator are the same as the initial three layers from the restricted generator in section 3.1.1. Following that, we have one additional convolutional layer succeeded by a fully connected output layer to predict the likelihood in the range [0,1], of the image being real or fake. Figure 3.5 shows the architectural details of both the discriminators.

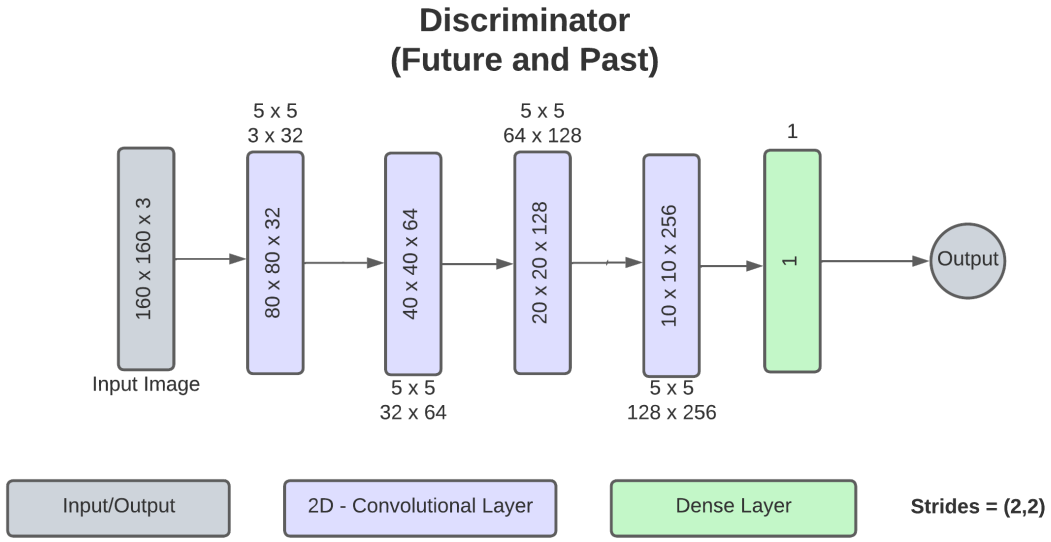


Figure 3.5: Structural details of discriminators $\mathcal{D}^{\mathcal{F}}$ and $\mathcal{D}^{\mathcal{P}}$ for the constrained Generator described in section 3.1.1

The kernel size and strides at each layer, are (5,5) and (2,2) respectively. After every convolutional layer, a batch normalization layer is added to regularize the output from the previous layer and it is followed by a LeakyReLU activation (with alpha = 0.2) layer. In the final output layer, Sigmoid activation is used to give the target likelihood of the image. Having discussed the discriminator(s) for the constrained generator from section 3.1.1, in the next subsection we will look into the discriminator architecture for the unconstrained generator from section 3.1.2

3.2.2 Discriminator for the CNN Encoder-Decoder Generator

As shown in figure 3.6, the first three layers of Discriminator are the same as the first three layers of the encoder section 3.1.2. There is one extra convolutional layer present followed by a fully connected dense layer to predict the likelihood of the image being real or fake.

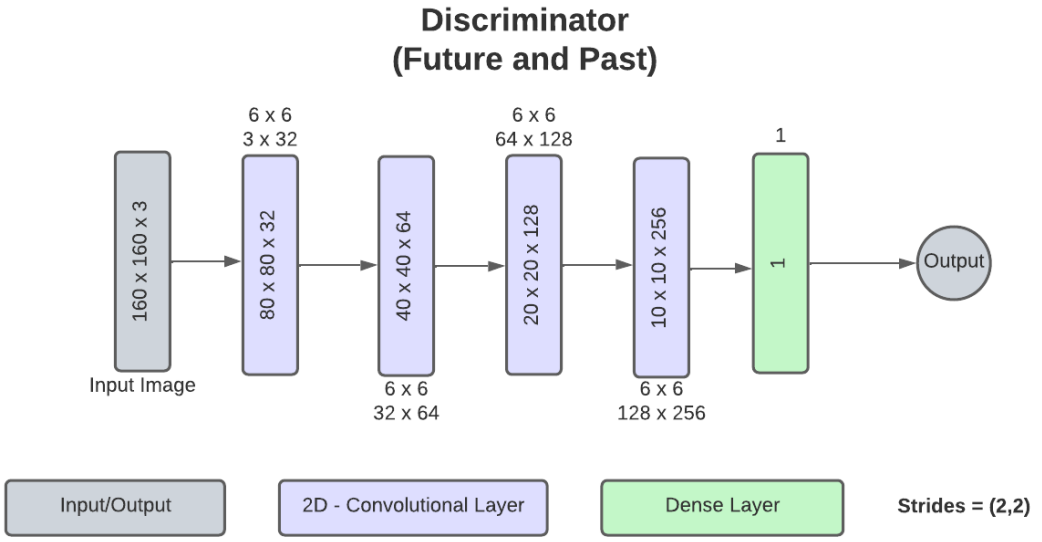


Figure 3.6: Structural details of discriminators $\mathcal{D}^{\mathcal{F}}$ and $\mathcal{D}^{\mathcal{P}}$ for the CNN Generator described in section 3.1.2

The kernel size is the same as CNN Encoder-Decoder Generator i.e. (6,6). A batch normalization layer followed by a LeakyReLU activation layer is present after every convolutional layer. In the dense layer, Sigmoid activation provides the likelihood of the input image being fake or real. Now in the next subsection, we look at discriminator architecture corresponding to the constrained generator from 3.1.3

3.2.3 Discriminator for Constrained CNN Encoder-Decoder Generator with different kernel size and latent space

As shown in the figure 3.7, the first four layers in the discriminator are the same as that of the initial four layers in the encoder of the constrained Generator with different kernel size and latent space mentioned in section 3.1.3 and a fully-connected layer with Sigmoid activation is added at the end to estimate the likelihood of the image being real or fake.

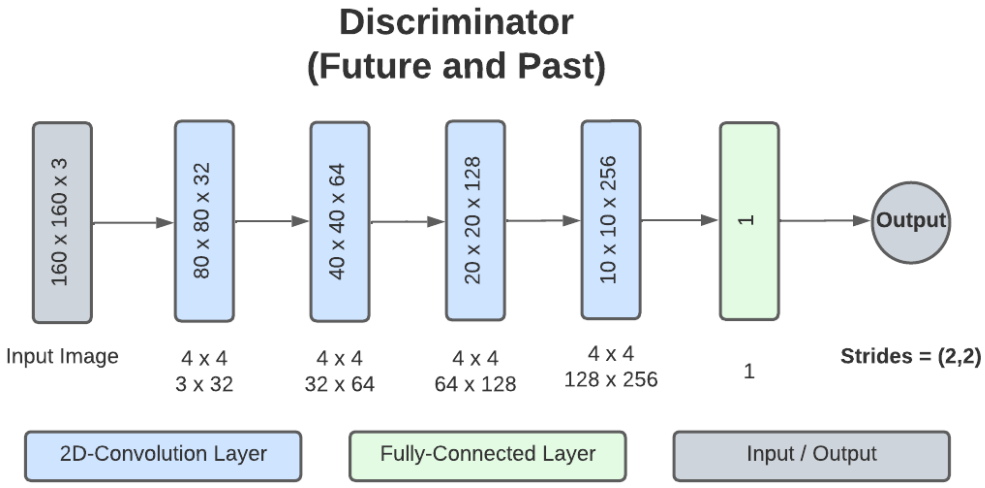


Figure 3.7: Structural details of discriminators $\mathcal{D}^{\mathcal{F}}$ and $\mathcal{D}^{\mathcal{P}}$ for the constrained Generator described in section 3.1.3

The kernel size and strides at each layer are (4,4) and (2,2) respectively. Batch normalization is used after each layer followed by a LeakyReLU activation layer. With this discriminator architecture, two discriminators one for past events ($\mathcal{D}^{\mathcal{P}}$) and the other for future events ($\mathcal{D}^{\mathcal{F}}$) are modeled.

We'll see how the aforementioned generators and discriminator(s) combine to form our proposed GAN in the next section. We will also discuss the loss functions for each of the three models, $\mathcal{R}$, $\mathcal{D}^{\mathcal{F}}$ and $\mathcal{D}^{\mathcal{P}}$, and how they are trained adversarially.

3.3 Proposed Adversarial Training for the GAN

In this section we describe how the above-mentioned Generator and Discriminator architectures constitute the overall GAN architecture. The proposed GAN architecture, as mentioned earlier, is inspired by the 3D architecture by Yu et al. [4] consisting of one Generator $\mathcal{R}$ and two Discriminators, Past Discriminator $\mathcal{D}^{\mathcal{F}}$ and Future Discriminator $\mathcal{D}^{\mathcal{P}}$.

Figure 3.8 shows how the 2D generators, from section 3.1 and discriminator, from section 3.2 interact with each other in an adversarial manner to form our end-to-end one-class classifier based on GAN.

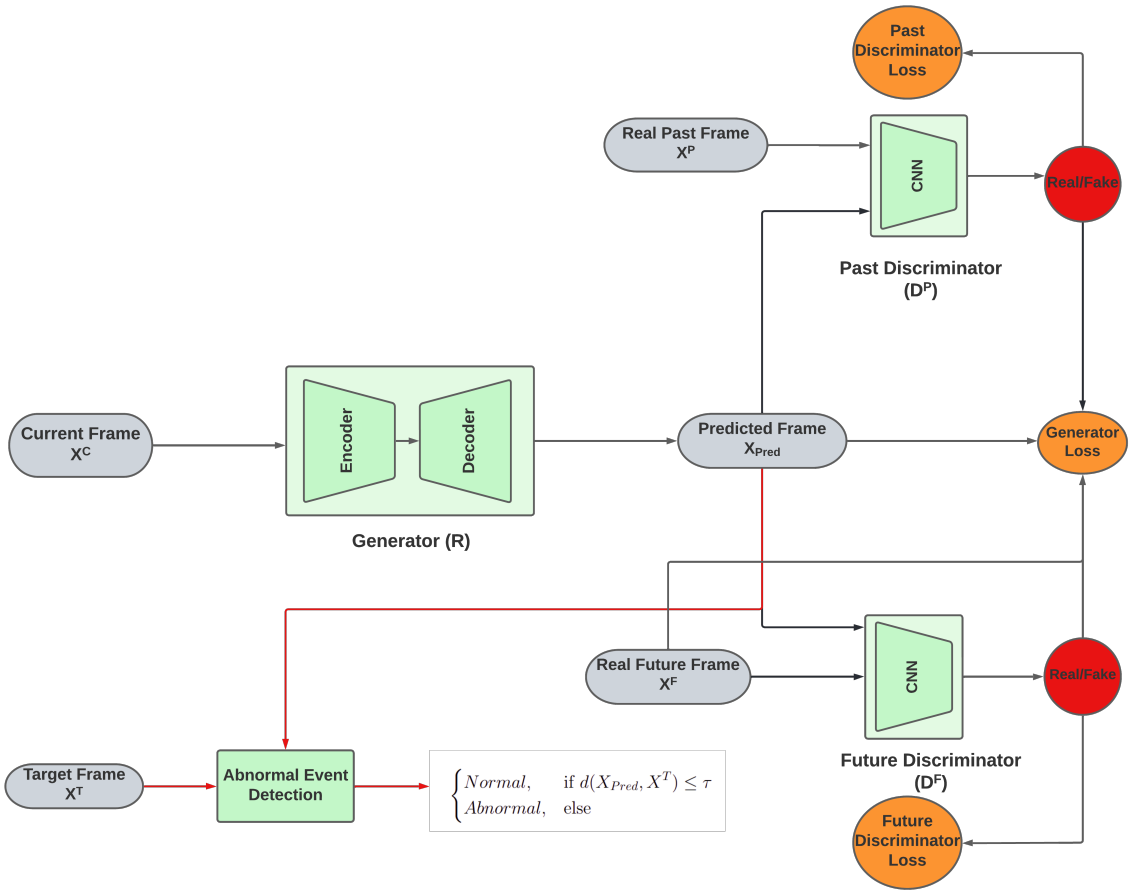


Figure 3.8: Structural details of the OCC based on GAN

The encoder in Generator $\mathcal{R}$, maps the input dynamic image frame $\mathcal{X}^{\mathcal{C}}$ to a latent space and then the decoder generates the prediction image $\bar{\mathcal{X}}$ using those latent features. To improve the robustness of the model Gaussian noise η is added to $\mathcal{X}^{\mathcal{C}}$. Thus $\mathcal{R}$ is represented by

$$\mathcal{R}(\mathcal{X}^{\mathcal{C}} + \eta) = \bar{\mathcal{X}} \tag{3.1}$$

Where η is sampled from $\mathcal{N}(0, \sigma^2 I)$. For the rest of the report, we will use $\mathcal{N}_\sigma$ to denote the noise model. The predicted sample $\bar{\mathcal{X}}$ is given to the two discriminators, $\mathcal{D}^{\mathcal{F}}$ and $\mathcal{D}^{\mathcal{P}}$ which produce scalar values on the Euclidean space representing the input sample's likelihood for past or future and real or fake.

The future discriminator $\mathcal{D}^{\mathcal{F}}$ is defined as:

$$\mathcal{D}^{\mathcal{F}}(\mathcal{X}^{(*)\mathcal{F}}) = v^{\mathcal{D}^{\mathcal{F}}}, \quad v^{\mathcal{D}^{\mathcal{F}}} \epsilon\ \mathcal{R}^1 \tag{3.2}$$

where $v^{\mathcal{D}^{\mathcal{F}}}$ is the output of $\mathcal{D}^{\mathcal{F}}$ and $\mathcal{X}^{(*)\mathcal{F}}$ represents the input to $\mathcal{D}^{\mathcal{F}}$, it can be the real future dynamic image frame $\mathcal{X}^{\mathcal{F}}$ or the predicted future image frame $\bar{\mathcal{X}}$. By evaluating the confidence value for identifying whether a given frame is the ground truth, $v^{\mathcal{D}^{\mathcal{F}}}$ determines whether a particular sample is the predicted outcome or the ground truth.

$\mathcal{D}^{\mathcal{P}}$ is defined as follows:

$$\mathcal{D}^{\mathcal{P}}(\mathcal{X}^{(*)\mathcal{P}}) = v^{\mathcal{D}^{\mathcal{P}}}, \quad v^{\mathcal{D}^{\mathcal{P}}} \epsilon\, \mathcal{R}^1 \tag{3.3}$$

where $v^{\mathcal{D}^{\mathcal{P}}}$ is the output of $\mathcal{D}^{\mathcal{P}}$ and $\mathcal{X}^{(*)\mathcal{P}}$ can be as $\mathcal{X}^{\mathcal{P}}$, the real past dynamic image frame or $\bar{\mathcal{X}}$. The structure of $\mathcal{D}^{\mathcal{P}}$ is the same as that of $\mathcal{D}^{\mathcal{F}}$.

The suggested adversarial learning is based on metric learning, in which it is usual to employ both positive and negative samples when building a model to increase the discriminative nature of learned features. In our model, the future frame $\mathcal{X}^{\mathcal{F}}$ act as the positive sample, and the past event frame $\mathcal{X}^{\mathcal{P}}$ act as the negative sample. By confining the representation learning for previous events and applying the intended adversarial learning for past and future events, the model can produce a more discriminative representation for anticipating the event's future.

The loss function for $\mathcal{D}^{\mathcal{F}}$ for the proposed adversarial learning is defined by:

$$\mathcal{L}_{\mathcal{D}^{\mathcal{F}}} = \mathbb{E}_{\mathcal{X}^{\mathcal{C}} \sim p_{\mathcal{X}^{\mathcal{C}}+\mathcal{N}_\sigma}}[\mathcal{D}^{\mathcal{F}}(\mathcal{R}(\mathcal{X}^{\mathcal{C}}))] - \mathbb{E}_{\mathcal{X}^{\mathcal{F}} \sim p_{\mathcal{X}^{\mathcal{F}}}}[\mathcal{D}^{\mathcal{F}}(\mathcal{X}^{\mathcal{F}})] \tag{3.4}$$

where $\mathcal{X}^{\mathcal{C}}$ is the current dynamic image frame with Gaussian noise for which we want to predict the future, $\mathcal{X}^{\mathcal{F}}$ denotes the target future dynamic image frame corresponding to $\mathcal{X}^{\mathcal{C}}$.

The loss function for the past discriminator with past dynamic image frame as $\mathcal{X}^{\mathcal{P}}$ is defined as follows:

$$\mathcal{L}_{\mathcal{D}^{\mathcal{P}}} = \mathbb{E}_{\mathcal{X}^{\mathcal{C}} \sim p_{\mathcal{X}^{\mathcal{C}}+\mathcal{N}_\sigma}}[1 - \mathcal{D}^{\mathcal{P}}(\mathcal{R}(\mathcal{X}^{\mathcal{C}}))] - \mathbb{E}_{\mathcal{X}^{\mathcal{P}} \sim p_{\mathcal{X}^{\mathcal{P}}}}[1 - \mathcal{D}^{\mathcal{P}}(\mathcal{X}^{\mathcal{P}})] \tag{3.5}$$

The prediction results $\bar{\mathcal{X}}$ are generated by the generator $\mathcal{R}$, and the discriminators check them for legitimacy. $\mathcal{D}^{\mathcal{F}}$ determines whether each

sample it examines pertains to the real future or the generated results, and $\mathcal{D}^{\mathcal{P}}$ determines if each sample it examines belongs to the predicted results or the real past.

The objective of their loss functions appears to be the distinguishing factor between $\mathcal{L}_{\mathcal{D}^{\mathcal{F}}}$ and $\mathcal{L}_{\mathcal{D}^{\mathcal{P}}}$. The purpose of $\mathcal{L}_{\mathcal{D}^{\mathcal{F}}}$ is to maximise $\mathcal{D}(\mathcal{X}^{\mathcal{F}})$. The goal of $\mathcal{L}_{\mathcal{D}^{\mathcal{P}}}$, on the other hand, is to minimize $\mathcal{D}(\mathcal{X}^{\mathcal{P}})$, which is the inverse of $\mathcal{L}_{\mathcal{D}^{\mathcal{F}}}$.

The loss function for adversarial learning of $\mathcal{R}$ is defined as

$$\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{\mathcal{X}^{\mathcal{C}} \sim p_{\mathcal{X}^{\mathcal{C}}} + \mathcal{N}_{\sigma}}[\mathcal{D}^{\mathcal{F}}(\mathcal{R}(\mathcal{X}^{\mathcal{C}}))] + \mathbb{E}_{\mathcal{X}^{\mathcal{C}} \sim p_{\mathcal{X}^{\mathcal{C}}} + \mathcal{N}_{\sigma}}[1 - \mathcal{D}^{\mathcal{P}}(\mathcal{R}(\mathcal{X}^{\mathcal{C}})] \quad (3.6)$$

To optimize $\mathcal{R}$, we integrate the reconstruction error in addition to adversarial learning of future and past frames. The minimizing of the reconstruction error based on the pixel-wise mean square error(MSE) is an efficient way of ensuring that the generator's expected output is close to the target. MSE is calculated as:

$$\mathcal{L}_{re} = \mathbb{E}_{\mathcal{X}^{\mathcal{C}}, \mathcal{X}^{\mathcal{F}}} ||\mathcal{X}^{\mathcal{F}} - \mathcal{R}(\mathcal{X}^{\mathcal{C}})||_2^2 \quad (3.7)$$

In addition, to force the output Z(from the encoder) to follow from Gaussian distribution, the KL Divergence term $KL(z||\mathcal{N}(\mu_1, \sigma_1))$ is also added to the loss function of the generator.

As a result, when optimizing the generator $\mathcal{R}$, the loss function is defined by

$$\mathcal{L}_{\mathcal{R}^*} = \mathcal{L}_{\mathcal{R}} + \mathcal{L}_{re} + KL(z||\mathcal{N}(\mu_1, \sigma_1)) \quad (3.8)$$

Each discriminator and the generator, are trained by maximizing or minimizing the loss function that corresponds to them. The weight update rule for each of them is defined as:

$$\theta_{\mathcal{D}^{\mathcal{F}}} = \theta_{\mathcal{D}^{\mathcal{F}}} + \alpha \frac{d\mathcal{L}_{\mathcal{D}^{\mathcal{F}}}}{d\theta_{\mathcal{D}^{\mathcal{F}}}}, \ \theta_{\mathcal{D}^{\mathcal{P}}} = \theta_{\mathcal{D}^{\mathcal{P}}} + \alpha \frac{d\mathcal{L}_{\mathcal{D}^{\mathcal{P}}}}{d\theta_{\mathcal{D}^{\mathcal{P}}}}$$

$$\theta_{\mathcal{R}} = \theta_{\mathcal{R}} + \alpha \frac{d\mathcal{L}_{\mathcal{R}}}{d\theta_{\mathcal{R}}}$$

where θ^* represents the parameters corresponding to the generator $\mathcal{R}$ and the discriminators $\mathcal{D}^{\mathcal{F}}$, and $\mathcal{D}^{\mathcal{P}}$ and α denotes a learning rate in updating each parameter using the gradient of the three-loss functions.

After looking at several generators and their correpsonding discriminator architectures, we address the theoretical basis of adversarial training of these models in this chapter. Now, in the following chapter, we'll look at development environment settings, datasets used for training and testing the model, and additional training details.

Chapter 4

Experimentation Details

In this chapter, we look at the two datasets we have worked upon, namely, UCSD Peds and CUHK Avenue in depth. Following that, we will discuss the training and testing of our proposed model with both DCGAN and WGAN and with different loss functions. But before that, we will discuss the environmental setup for implementation of the proposed GAN based One-Class Classifier in the next section.

4.1 System Specifications

The Google Colaboratory is used as the platform to perform experiments evaluation. To reduce the training time we have used the google colaboratory's default "NVIDIA Tesla K80" GPU with 12 GB of RAM. The algorithms were implemented in Python version 3.7. We also make use of python exclusive libraries and packages like Tensorflow version 2.8, and Keras version 2.8 to develop the deep learning model. In the next section, we will take a look at the two datasets which we have worked upon for training and testing our model.

4.2 Dataset Description

A key step in developing a good machine learning model for a problem is selecting a good dataset to train and test the model. As we are developing an OCC model, the training videos in each dataset contain only normal events while the testing videos contain both normal and abnormal events which help us to evaluate our model for accuracy. Due to limitations in the system configuration and lack of time for development, we have worked on just two anomaly detection datasets,

UCSD Peds [17] and CUHK Avenue Dataset [18]. The datasets are further described in further subsections.

4.2.1 UCSD Peds Dataset

The UCSD(University of California San Diego) anomaly detection dataset was obtained by mounting a stationary camera at an elevation, looking over the pedestrian walkways. The density of pedestrians varies from very crowded to sparse. The normal events in the videos are when only pedestrians are in the frame while the abnormality in the videos comes from, non-pedestrian entities in the videos and anomalous motion patterns from the pedestrians.

(a) Peds1 - Normal Frame

(b) Peds1 - Abnormal Frame(Cyclist)

(c) Peds2 - Normal Frame

(d) Peds2 - Abnormal Frame(Cart)

Figure 4.1: UCSD Peds Dataset: Example dynamic image frames

Bikers, skaters, small carts, and people walking on a walkway or in the grass that surrounds it are all common abnormalities. There were also a few cases of people in wheelchairs. The recorded video footage

from each scene was clipped into different clips of nearly 200 frames each. The dataset was divided into two parts, Peds1 and Peds2. The Peds1 dataset contains 34 training videos (6800 frames in total) and 36 testing videos (7200 frames in total) while the Peds2 contains 16 training videos (2550 frames in total) and 12 testing videos (2010 frames in total) samples. For each frame, we have a binary flag ground truth annotation telling us if a particular frame is normal or anomalous. Figure 4.1 shows two example frames from UCSD Peds1 (a) a normal frame which does not contain any anomalous event (b) an abnormal frame in which a cyclist riding on the pedestrian walkway which is an anomalous event, and two example frames from UCSD Peds2 (c) a normal frame which do not have any anomalous event (d) an abnormal frame in which a cart coming on the pedestrian walkway is considered an anomalous event. In the following subsection, we will look into another standard anomaly detection dataset, the Avenue Dataset.

4.2.2 CUHK Avenue Dataset

The Avenue Dataset is from The Chinese University of Hong Kong(CUHK). The videos for this dataset are recorded in CUHK campus avenue. The dataset contains 16 training videos(15328 frames in total) and 21 testing video clips(15324 frames in total). The resolution of each video is 360×640. The dataset features a number of difficult challenges, including camera shake, outliers in training films, and the lack of motion in training videos.

(a) Normal Frame

(b) Abnormal Frame(Throwing)

Figure 4.2: Avenue Dataset: Example dynamic image frames

Figure 4.2 shows two example frames from the CUHK Avenue dataset (a)a normal frame which does not contain any anomalous even and (b) an abnormal frame in which a person performing strange action throwing which is considered an anomalous event. Here also we make use of frame-level annotations, a binary flag depicting if a certain frame is anomalous or not. Now as the extracted frames from both these datasets are large in number and the frames from the Avenue dataset differ in size from the UCSD Peds, thus we perform pre-processing over the datasets, which will be discussed in further detail in the next subsection.

4.2.3 Data Pre-Processing

A video is a stack of still images and many of these still images are very close to each other especially when they are consecutive frames. Thus we can say our model trains on redundant frames. To mitigate the use of unnecessary frames we pre-process the frames from the above-mentioned datasets we make use of dynamic images [19]. Dynamic images are a summarization of eight consecutive video frames. These consecutive frames are ranked using some ranking mechanism such as rank-SVM and then passed through convolutional layers and fully connected layers to get a single dynamic image of the 8 frame cube.

So we obtain the dynamic frames for all of our datasets. After dynamic imaging, the size of the Peds1 dataset was reduced to 850 training frames and 900 testing frames while the size of Peds2 reduces to 310 training frames and 245 testing frames and for the Avenue dataset, we get 1906 dynamic image frames for both the testing and training dataset. Once the dynamic images are obtained, they are resized and normalized to 160×160 and [0,1] respectively to maintain the consistency of our model throughout different datasets. After pre-processing of the datasets is completed, we look at the training details of the GAN based OCC in the next section.

4.3 Training of proposed GAN based OCC

This section contains the details of the training of our models based on architectures mentioned in Section 3.1 and section 3.2.

We started by implementing normal DCGAN which comprises a Generator and Discriminator(s). We experimented with different models by making variations in Generator and Discriminator (For example - changing the kernel size, removing the dense layer between encoder and decoder, altering the activation function in the last layer of decoder, etc.).

For the loss functions of Generator and Discriminator, we made use of the Binary Cross-Entropy (BCE) loss function. BCE is the most common loss function for Binary Classification. It compares the predicted probabilities to the actual label and then penalizes the probabilities. The penalty is directly proportional to the difference between the actual label and predicted probabilities. Mathematically, it can be defined as the negative average of the log of corrected predicted probabilities. We use Mean Squared Error (MSE) loss to calculate reconstruction loss.

The optimization algorithm used is the Adam optimizer. It is the optimization algorithm that is an improvement of normal stochastic gradient descent. The major application of this algorithm is in solving problems involving large data or a lot of parameters. We used the in-built function available in Python for Adam Optimizer for the learning rate of 0.00015.

After this, we tried a different loss function called Squared Hinge loss function. It is different from Binary Cross-Entropy (BCE) as along with wrongly classified labels, it also penalizes the correct labels within a defined margin from the decision boundary. With this loss function, we used the same optimizer (Adam Optimizer) keeping the parameters set the same. Note that both the loss functions (BCE and Squared Hinge) have been used on normal DCGAN implementation.

Moving forward, we experimented with WGAN due to certain advantages it has over DCGAN. WGAN solves the problems like vanishing gradient and mode collapse which are generally faced in normal GAN. Further details about this have been already mentioned

in the second chapter. In Wasserstein GAN, for discriminator loss functions we have used reduce mean loss function. For the generator, the loss function is BCE (Binary Cross Entropy). The optimizer used for WGAN implementation is RMSProp(Root Mean Squared Propagation) with a learning rate of 0.00005 for the Generator and Adam for the discriminator. RMSProp helps in faster convergence and accelerates the optimization process.

Having discussed the training details, in the next section we will see the testing procedure of our GAN-based OCC, how an appropriate threshold value is estimated to mark a frame as normal or abnormal, and how we proceed to calculate the classification accuracy.

4.4 Testing of proposed GAN based OCC

The trained generator $\mathcal{R}$ predicts the future frame for the current input frame from the testing dataset which contains both normal and abnormal frames. As the model has only been trained on normal event samples, if the input frame is anomalous, the predicted frame is reconstructed with either a lower likelihood or a higher error.

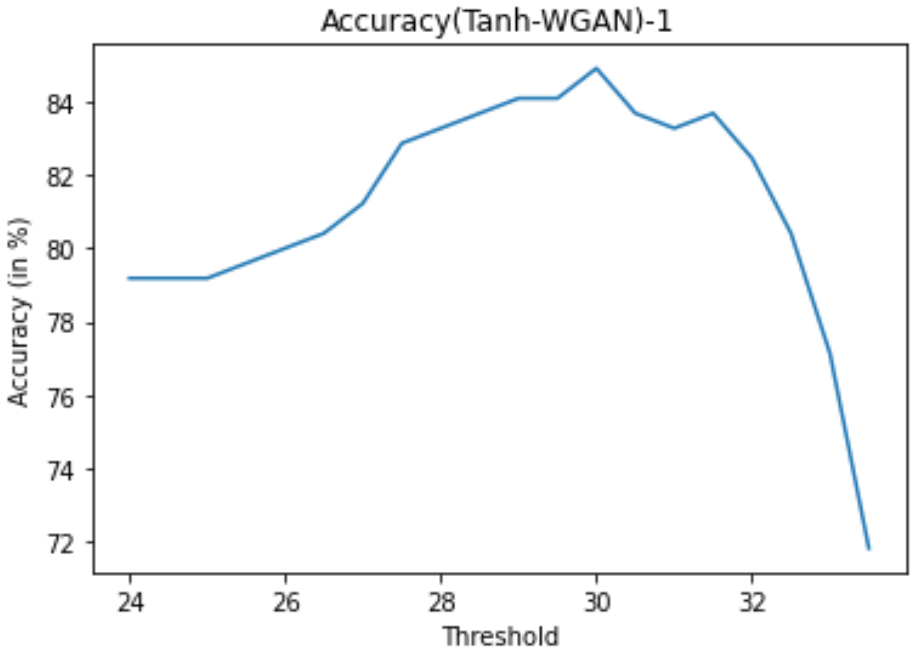
For the input frame, X^C the generator $\mathcal{R}$ predicts the future frame $\bar{X}$, and abnormal events can be detected by comparing the predicted output $\bar{X}$ with their respective target frames X^T in the test step. Now the absolute difference between predicted future frame $\bar{X}$ and target frame X^T is obtained and convolution is performed on the difference matrix (d) to mark anomalous pixels with the help of neighboring pixels. So having each anomalous pixel marked as shown in figure 5.1, the anomalous pixel percentage in the predicted frame $\bar{X}$ is calculated and used to estimate the appropriate threshold τ.

As the model is only optimized for normal event frames, there would be a significant difference in anomaly percentages for an abnormal event frame when compared with a normal event frame. With the help of this larger error, a threshold τ is estimated as below

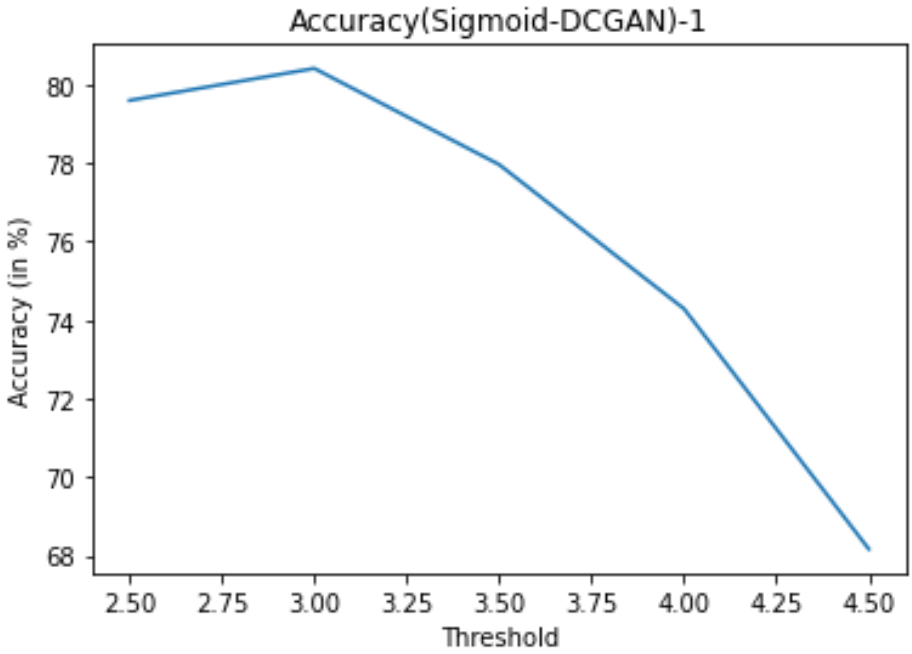
$$OCC(x) = \begin{cases} \text{Normal event} & \text{if } \mathcal{F}_{x^n}(x) \leq \tau \\ \text{Abnormal event} & \text{otherwise} \end{cases}$$

τ represents the threshold, x represents the samples of input, and $\mathcal{F}_{x^n}$

is the model trained only on normal events.



(a) Plot of Accuracy Vs Threshold on UCSD Peds2 dataset with WGAN implementation



(b) Plot of Accuracy Vs Threshold on UCSD Peds2 dataset with DCGAN implementation

Figure 4.3: Accuracy Vs Threshold Percentage plots

For estimating the best threshold τ, accuracy is calculated for various thresholds with small increments and plotted for all the model architectures and two of them can be seen in figure 4.3(a) shows the plot

of accuracy in percentage *vs* threshold with WGAN implementation and Tanh activation used in the final layer of decoder and figure 4.3(b) shows the plot of accuracy in percentage *vs* threshold with DCGAN implementation and Sigmoid activation used in the final decoder layer.

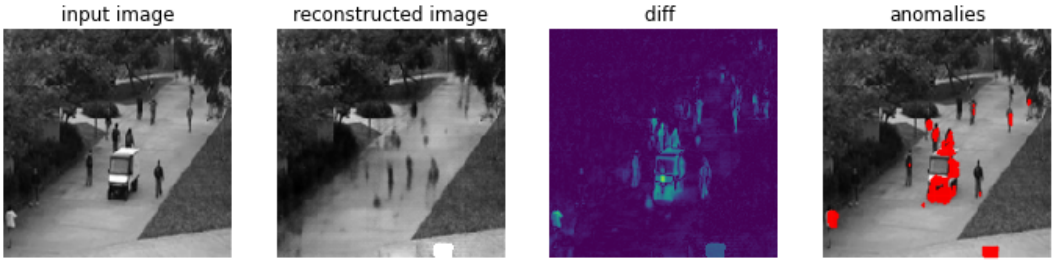


Figure 4.4: UCSD Peds1: Future frame prediction and anomaly detection(truck in the pedestrian walkway) is marked in red-colored pixels

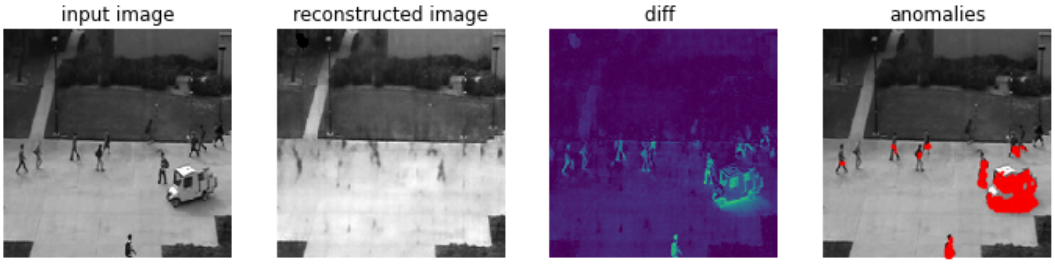


Figure 4.5: UCSD Peds2: Future frame prediction and anomaly detection(cart in the pedestrian walkway) is marked in red-colored pixels

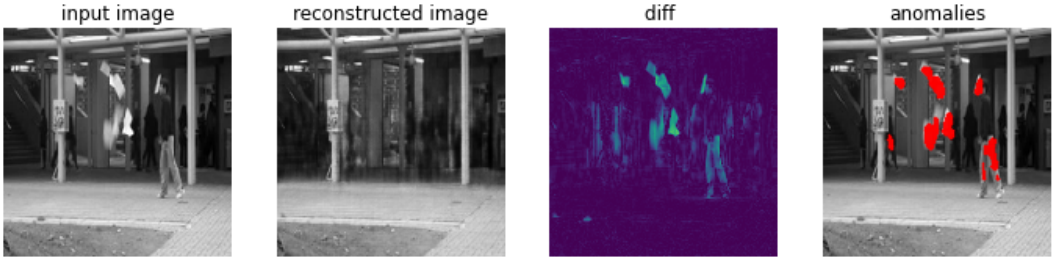


Figure 4.6: CUHK Avenue: Future frame prediction and anomaly detection(strange action by a person) is marked in red-colored pixels

The input image in figure 4.4 is from testing data of the UCSD Peds1 dataset, and we can see a truck coming into the pedestrian walkway, which is an anomalous event and is marked in red pixels by our model. For the input image from testing data of the UCSD Peds2 dataset in figure 4.5 observe that a cart coming into the pedestrian walkway is an anomalous event and our model marks it with red-colored pixels. Similarly for an input image from the CUHK Avenue dataset shown in figure 4.6 a person throwing papers which is a strange action is identified by our model and marked in red-colored pixels.

Having discussed about the training and testing details, in the next chapter, now we will discuss the observed accuracy results on the two datasets and following that we will also take a look at some challenges faced while developing our model and how we overcame them.

Chapter 5

Results and Discussion

We use classification accuracy as an evaluation metric to test our model efficiency. As mentioned in section 4.4, we determine an appropriate threshold percentage, based on the number of anomalous pixels in the frame, to classify any frame as normal or abnormal. As we already have the ground truth labels of each frame, we cross-verify with the predicted label and calculate the accuracy of the model. Following the results, we also discuss some obstacles while developing the models and how we tried to tackle those.

5.1 Results

We state results in the same order as we mentioned the generator architectures in section 3.1, also with each generator we experiment with three different activation functions in the last layer, so we provide the accuracy measures for each of them in a tabular form in the following sections. Before working on the WGAN algorithm, we begin our experiment on the Peds2 dataset with the DCGAN algorithm, the corresponding results are also mentioned in the following subsections.

5.1.1 Results with constrained encoder-decoder Generator and the corresponding Discriminator

The following results for the generator from section 3.1.1 and its corresponding discriminator show that in LeakyReLU activation gives better accuracy in comparison with Sigmoid and Tanh activation functions for DCGAN on the UCSD Peds2 dataset. For the WGAN, LeakyReLU performs better on UCSD Peds, while with the Avenue dataset, both Sigmoid and Tanh activation functions give similar results

better than the LeakyReLU activation function.

Activation Function	DCGAN	WGAN		
	UCSD Peds2	UCSD Peds1	UCSD Peds2	CUHK Avenue
Sigmoid	79.59%	63.0%	80.0%	72.66%
LeakyReLU	80.8%	64.11%	80.4%	72.61%
Tanh	79.59%	62.77%	79.59%	72.66%

Table 5.1: Performance of our OCC model with Generator from section 3.1.1 and Discriminator from section 3.2.1 over different datasets

5.1.2 Results with CNN Encoder-Decoder Generator and the corresponding Discriminator

The following results for the generator from section 3.1.2 and its corresponding discriminator show that LeakyReLU and Tanh activation functions give better accuracy in comparison with Sigmoid activation, for DCGAN on the UCSD Peds2 dataset. For the WGAN, Sigmoid performs better on UCSD Peds1 while Tanh gives superior results with Peds2 and for the Avenue dataset, LeakyReLU gives the best results.

Activation Function	DCGAN	WGAN		
	UCSD Peds2	UCSD Peds1	UCSD Peds2	CUHK Avenue
Sigmoid	79%	62.22%	79.18%	72.51%
LeakyReLU	79.18%	61.78%	79.18%	72.77%
Tanh	79.18%	58.89%	84.89%	72.46%

Table 5.2: Performance of our OCC model with Generator from section 3.1.2 and Discriminator from section 3.2.2 over different datasets

5.1.3 Results with constrained encoder-decoder Generator from section 3.1.3 and the corresponding Discriminator

The following results for the generator from section 3.1.3 show that LeakyReLU and Tanh activation perform similar and better in comparison with Sigmoid activation function for DCGAN. For the WGAN, LeakyReLU performs better on the UCSD Peds, while with the Avenue dataset, Tanh gives the best results. In the following section, we discuss the problems faced during the development of the OCC model and how we overcame them.

Activation Function	DCGAN	WGAN		
	UCSD Peds2	UCSD Peds1	UCSD Peds2	CUHK Avenue
Sigmoid	77.5 %	63.88%	79.59%	72.45%
LeakyReLU	80.4%	64.66%	80.4%	72.44%
Tanh	80.4%	62.66%	79.59%	72.50%

Table 5.3: Performance of our OCC model with Generator from section 3.1.3 and Discriminator from section 3.2.3 over different datasets

5.2 Discussion

The obtained results demonstrate the capability of our model to detect anomalies in dynamic images. During the duration of this project, we came across various challenges. In the beginning, we were training our models on Google colaboratory CPU but the training time of our models was very high. In order to reduce the overall training time, we decided to switch to Google colab GPU which significantly improved our training duration. Another problem we faced was in the choice of loss functions for our models. There were many options like Binary Cross-Entropy(BCE), Squared Hinge, etc. We implemented both and decided to continue with BCE as it was giving us better results in comparison to Squared Hinge. To overcome the problems of Vanishing Gradient and Mode Collapse, we implemented WGAN.

We train our model in an adversarial manner in which Generator and Discriminator compete with each other and keep on getting better and better during the training phase. One important factor in our training was to determine the stopping criterion i.e. to decide when to stop the training of our model. To handle it, we saved the most valuable parameters for both Generator and Discriminator and tried to reach convergence by the hit and trial method. Between Generator and discriminator, we kept the model which is giving lower loss constant, and train the other model to reach minimum loss.

When we pass any abnormal image through our model, the anomaly in the image is highlighted in red pixels as shown with examples in the section 4.4 which helps in better detection of the anomalies. In the next chapter, we will conclude the work we have done in this project and discuss the possible future work.

Chapter 6

Conclusion and Future Work

This work has helped us to understand the process required to build a machine learning model, starting from the analysis phase, understanding the need from the real-world use cases, the in-depth technical knowledge required to develop the machine learning model, preparing the dataset for training the model, understanding and identifying the features that are required for training the model, choosing the right algorithm for preparing the model, and evaluating the machine learning model to analyze its performance and accuracy.

We design and experiment with different GAN architectures to model an OCC based on reconstruction for anomaly detection. We start by modeling an encoder-decoder-based Generator architecture and based on that we model our discriminator(s). After that, we combine the two architectures and make the proposed GAN-based OCC. After detecting the problem of vanishing gradient during training the model via the Goodfellow GAN algorithm(DCGAN) we change our approach and make use of the WGAN. Using different activation functions at the end of Generator also helped us to expand our analysis of the generated dynamic frames. These models have been tested on UCSD Peds and CUHK Avenue datasets and our observations show that the OCC based on the constrained generator from section 3.1.1 and the corresponding discriminator performs better, with an average accuracy of 75.72%, in comparison with the GAN models from the other two generators. Also, in regards to activation function at the last layer of

the decoder, LeakyReLU gives better results as compared to Sigmoid and Tanh activation.

We have worked with 2-Dimensional architecture for our GAN-based OCC model so far, the next step could be to adapt it to 3D GAN architecture. Also, moving to the next phase, another feature can be added to the model where we classify the anomalies into specific classes, after detecting them.

Bibliography

[1] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 935–942, IEEE, 2009.

[2] Y. Feng, Y. Yuan, and X. Lu, "Deep representation for abnormal event detection in crowded scenes," in *Proceedings of the 24th ACM international conference on Multimedia*, pp. 591–595, 2016.

[3] P. Wu, J. Liu, and F. Shen, "A deep one-class neural network for anomalous event detection in complex scenes," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 7, pp. 2609–2622, 2019.

[4] J. Yu, Y. Lee, K. C. Yow, M. Jeon, and W. Pedrycz, "Abnormal event detection and localization via adversarial event prediction," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[5] M. Sabokrou, M. Fathy, G. Zhao, and E. Adeli, "Deep end-to-end one-class classifier," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 675–684, 2020.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[7] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," *Advances in neural information processing systems*, vol. 28, 2015.

[8] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[9] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[12] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[13] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[15] J. M. Joyce, "Kullback-leibler divergence.," *International encyclopedia of statistical science*, vol. 720, p. 722, 2011.

[16] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab, "Deep autoencoding models for unsupervised anomaly segmentation in brain mr images," in *International MICCAI brainlesion workshop*, pp. 161–169, Springer, 2018.

[17] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 1975–1981, IEEE, 2010.

[18] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *Proceedings of the IEEE international conference on computer vision*, pp. 2720–2727, 2013.

[19] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi, "Action recognition with dynamic image networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2799–2813, 2017.