Henrique Ferreira Cardoso
NEUROMORPHIC VISION-BASED TRAFFIC ANOMALY DETECTION
UNIVERSIDADE D COIMBRA
1 2 9 0

1 2  9 0

UNIVERSIDADE D
COIMBRA

Henrique Ferreira Cardoso

NEUROMORPHIC VISION-BASED TRAFFIC ANOMALY DETECTION

Dissertation within the scope of an Integrated Masters in Eletrical and Computer Engineering supervised by Professor Jorge Batista presented to the Department of Eletrical and Computer Engineering of the Faculty of Sciences and Technology of the University of Coimbra

February 2023

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Neuromorphic Vision-Based Traffic Anomaly Detection

Henrique Ferreira Cardoso

February 2023

Neuromorphic Vision-Based Traffic Anomaly Detection

Supervisor:

Professor Dr. Jorge Manuel Moreira de Campos Pereira Batista

Jury:

Prof. Dr. Hélder de Jesus Araújo

Prof. Dr. João Pedro de Almeida Barreto

Prof. Dr. Jorge Manuel Moreira de Campos Pereira Batista

Dissertation submitted in partial fulfilment for the degree of Master of Science in Electrical and Computer Engineering.

February 2023

Agradecimentos

A conclusão desta dissertação e fase da minha vida só foi possível devido às pessoas à minha volta. Como tal, gostaria de agradecer a algumas destas pessoas.

Em primeiro lugar, um grande obrigado ao meu orientador Professor Jorge Batista pela sua ajuda, dedicação e orientação ao realizar esta etapa. Um obrigado também pela confiança no meu trabalho e nas minhas capacidades. Agradeço também à equipa e colegas presentes no laboratório, que apresentaram disponibilidade para acolher e também ajudar em momentos de necessidade.

De seguida gostaria de agradecer especialmente aos meus pais, à minha irmã e à minha tia, que constantemente fizeram de tudo para me proporcionar as melhores oportunidades para alcançar o sucesso. Um eterno obrigado aos meus avós, que sempre me motivaram a seguir em frente. Sempre desejaram conseguir ver-me concluir esta fase, no entanto tal já não é possível.

Obrigado pelos meus amigos que sempre mostraram a sua presença em todas as vivências e memórias, especialmente nos momentos de fraqueza ou fragilidade. Sempre me deram força para seguir em frente e almejar um futuro com sucesso.

Obrigado a todos colegas e amigos de curso, que sempre me ajudaram a conquistar os melhores resultados ao longo da licenciatura e mestrado, sem os quais não seria possível chegar a esta meta.

Obrigado aos colegas da minha equipa de voleibol, que com o seu acompanhamento e dedicação, sempre tornaram possível manter o equilíbrio necessário entre o intelectual e o físico.

Por fim, um obrigado a todas as pessoas que se cruzaram comigo no decorrer da minha vida e que, consequentemente, ajudaram a moldar a pessoa que sou hoje.

Um sentido obrigado a todos!

Abstract

Anomaly detection is a crucial aspect of ensuring public safety in modern society. With the increasing use of automated systems and intelligent technologies, the ability to detect anomalous events in real-time has become more critical than ever. The proposed models in this work demonstrate the potential of using Generative Adversarial Networks (GANs) for Abnormal Event Detection (AED).

The data used in this work is obtained through the use of Event Cameras (ECs), which are a significant advantage over traditional cameras, as ECs operate by measuring changes in brightness in each pixel independently, allowing them to detect motion with a very high temporal resolution, which makes them well-suited for tasks such as real-time monitoring and other applications where high temporal resolution is important. The dynamic range of ECs enables them to capture images with higher efficiency in challenging conditions, such as low light or high contrast environments, surpassing the performance of conventional cameras.

This dissertation explores the use of GANs for AED in pedestrian contexts using event-based data in order to demonstrate the potential of these techniques, attempting to help improve the safety and efficiency of modern surveillance systems. Specifically, this work validates two GAN models, a conditional GAN (cGAN) and the Pix2Pix model, both making use of PatchGAN as a discriminator, to generate realistic and representative images from event-based data. The generated images are then used to detect whether an event is normal or abnormal, with the aim of seeing these events in real-world scenes.

Overall, this work benefits from the growing body of research on using GANs for computer vision tasks and demonstrates their potential to be used in real-world applications such as surveillance monitoring. The results of this work provide validation for existing research in this area adapted for AED tasks.

Keywords: Anomaly Detection, Event Cameras, Generative Adversarial Networks, Monitoring.

Resumo

A deteção de anomalias é um aspeto crucial da segurança pública na sociedade moderna. Com o aumento do uso de sistemas automatizados e tecnologias inteligentes, a capacidade de detetar eventos anómalos em tempo real tornou-se mais crítica do que nunca. Os modelos propostos neste trabalho mostram o potencial do uso de Generative Adversarial Networks (GANs) para a Deteção de Eventos Anómalos (AED).

Os dados utilizados neste trabalho são obtidos através do uso de Câmaras de Eventos (ECs), sendo uma vantagem significativa relativamente às câmaras tradicionais, visto que estas operam ao medir mudanças de brilho em cada pixel de forma independente, o que lhes permite detetar o movimento com uma resolução temporal muito elevada, tornando-as adequadas para tarefas como monitorização em tempo real e outras aplicações onde a alta resolução temporal é importante. A gama dinâmica das ECs permite-lhes capturar imagens com maior eficiência em condições aadversas, tais como ambientes com pouca luz ou com alto contraste, superando o desempenho das câmaras convencionais.

Esta dissertação usa GANs para AED em contextos pedestres usando dados de eventos, para mostrar o potencial dessas técnicas, ajudando a melhorar a segurança e eficiência dos sistemas de vigilância modernos. Especificamente, são validados dois modelos de GANs, uma conditional GAN (cGAN) e o modelo Pix2Pix, ambos fazendo uso de uma PatchGAN enquanto discriminador, para gerar imagens realistas e representativas a partir de dados de eventos. As imagens geradas são usadas para detetar se um evento é normal ou anómalo.

No geral, este trabalho beneficia da crescente pesquisa sobre o uso de GANs para tarefas de Visão por Computador e demonstra o seu potencial para serem usados em aplicações reais, tais como monitorização de vigilância. Os resultados deste trabalho fornecem validação para a pesquisa existente nesta área, adaptada para tarefas de AED.

Palavras-Chave: Deteção de Anomalias, Câmaras de Eventos, Generative Adversarial Networks, Monitorização.

Contents

List of Acronyms

AED	Abnormal Event Detection
AUC	Area Under Curve
CC-GAN	Context-Cueing Generative Adversarial Network
CNN	Convolutional Neural Network
cGAN	Conditional Generative Adversarial Network
DVS	Digital Video Stabilization
EC	Event Camera
EER	Equal Error Rate
EMST	Event-based Multiscale Spatial-Temporal
FN	False Negatives
FNR	False Negatives Rate
FP	False Positives Rate
FPR	False Positives Rate
Forward KL	Forward Kullback-Leibler
GAN	Generative Adversarial Network
GFTT	Good Features to Track
IOU	Intersection Over Union
MSE	Mean Squared Error
MOG	Mixture of Gaussians

MS	Memory Surface
OF	Optical Flow
ROI	Region of Interest
SAE	Surface of Active Events
SGD	Stochastic Gradient Descent
SR	Sparse Representation
SSC	Submanifold Sparse Convolutions
TP	True Positives
TPR	True Positives Rate
TN	True Negatives
TNR	True Negatives Rate
TS	Time Surface
YOLO	You Only Look Once

List of Figures

List of Tables

1 Introduction

This first chapter presents the context, main objectives and main contributions on which the developed work is based on. It is also provided an overview of the thesis structure.

1.1 Context and Motivation

In a time of rapid technological evolution, there is an associated need for Computer Vision to also evolve and advance its methods and algorithms. A widely explored area for quite some time has been the capacity to detect anomalies in various situations using conventional cameras, designated as AED. For example, if a street is being analysed in which pedestrians are supposed to walk in one direction, if one person walks in the opposite direction, such an event can be defined as an anomaly. If this same street is still under watch and pedestrians are expected to walk, if a person is jumping or running is performing an anomalous activity. This can be applied to the presented pedestrian situations to the same degree that it can be used in multiple situations, such as surveillance cameras or traffic contexts. Therefore, AED can grow to be an important part of safety in society. Since it has been introduced as a challenge, researchers developed either hand-crafted patterns and frameworks or Deep Learning (DL) algorithms, always improving their capacity to better detect anomalous events.

Lately, a different type of camera technology has surfaced and it has attracted a lot of interest due to its advantages. These are called ECs, bio-inspired novel sensors that asynchronously record changes in illumination in the form of events [1]. This allows the processing algorithms to only process the changes in a scene instead of working on the entire scene every time, providing higher efficiency since it has a smaller computational burden. A better description of the technology behind these cameras will be provided in Section 3.1.

A big concern of AED using conventional cameras is the privacy of individuals, due to the cameras being able to capture the appearance of recorded people. Another great impact

of ECs in the future of Computer Vision is the fact that, since it detects movement, they can mainly capture edges, presenting a way to perform the detection of anomalous events without the need to discuss privacy transgressions.

The above-mentioned characteristics, along with the fact that there are very few published articles and available online datasets concerning the use of ECs for AED, introduce not only a challenge but also a motivation for this dissertation to explore the use of these cameras with the implementation of a fitting DL algorithm to perform AED.

In the academic year 2020/2021, the author of the dissertation [2] thoroughly studied some AED methods and was the first to use ECs in our department. These methods will be presented in Section 2.2.

1.2 Main Objectives and Achievements

The use of DL algorithms has become a valuable asset in performing AED, due to its results. Although event-based AED still is poorly explored, it has also started to progress with the use of GANs, already existing a good performing method available in [3]. Further explanation of GANs will be provided in section 3.3.

In a concise manner, the main objectives of this dissertation are the implementation and training of a GAN model that makes use of supplied data in order to learn how to generate normal situations similar to real data. With this model being able to only generate normal events it is then possible to distinguish anomalous events when these are provided to the model since it will not be able to generate these events. The data supplied to train this model comes from a DL network that encodes the temporal information of event-based frames.

The fact that GANs are unsupervised models makes them an asset for AED since the range of possible anomalous events is unknown. This means that a simple classifier cannot be used to predict the existence of an anomaly. By generating new normal data in a GAN, an error can be detected between the generated and real data, indicating an anomaly.

Figure 1.1 demonstrates an example of an anomaly. In this case, a walkway for pedestrians is shown in which pedestrians walking is considered a normal event and the presence of other entities, such as cars, is considered an anomalous event.

Figure 1.1: Example of a normal (left) and anomalous (right) event.

In order to fulfil the desired objectives, the algorithms developed and presented in this dissertation are:

- DL Memory Surface (MS)
- Conditional Generative Adversarial Network (cGAN)

These algorithms are based on the implementation detailed in [3] with an adaptation based on [4].

A more detailed explanation of these algorithms is provided in Section 4.

1.3 Thesis Overview

This dissertation's composition goes as follows:

- **Chapter 2:** The state of the art regarding AED using both conventional cameras and ECs.
- **Chapter 3:** The required knowledge to fully understand the proposed methodologies.
- **Chapter 4:** The materials and methods required to develop the proposed work.
- **Chapter 5:** The implemented work and each step toward the final results.
- **Chapter 6:** The analysis of the results achieved during and after the developed work.
- **Chapter 7:** The final conclusions, limitations and future work to be done.

2 State of the Art

This chapter introduces previous work related to AED using conventional frame and event-based cameras. Some of the techniques presented will be based on hand-crafted frameworks but the most important ones utilise DL algorithms. It is to be noted that methods regarding conventional frame cameras are vastly more explored, being presented more strategies regarding these cameras.

2.1 AED using Conventional Cameras

Although the main objectives of this dissertation involve the use of event-based data provided by ECs, it is reasonable to first analyse the methods developed for AED with traditional cameras.

Anomaly detection is an unsupervised pattern recognition task in which the use of deep generative models has a become valuable asset. These methods, along with their main focus and performance, are discussed in [5]. Some of the most relevant methods will be presented below.

It is worth noticing that the approaches that will be further explained mainly focus on background and foreground modelling, providing spatial context. All these methods also use Deep Learning algorithms for the detection of objects in each frame.

2.1.1 AI City Challenge

With the goal of making cities and their roads safer, the Nvidia AI City Challenge aims to improve the efficiency of operations in city environments by challenging its competitors to develop methods that solve certain problems and are part of intelligent city transportation management systems. Since 2018, one of the problems present in this challenge has been the detection of abnormal events.

5th AI City Challenge 2021

The framework that is going to be presented has ranked first place in the 5th AI City Challenge Challenge 2021 [6] by using a hand-crafted framework presented and discussed in [7].

The authors of [7] proposed an approach that consists of pre-processing, a dynamic tracking module and post-processing. The pipeline for this implementation is visible in Figure 2.1.

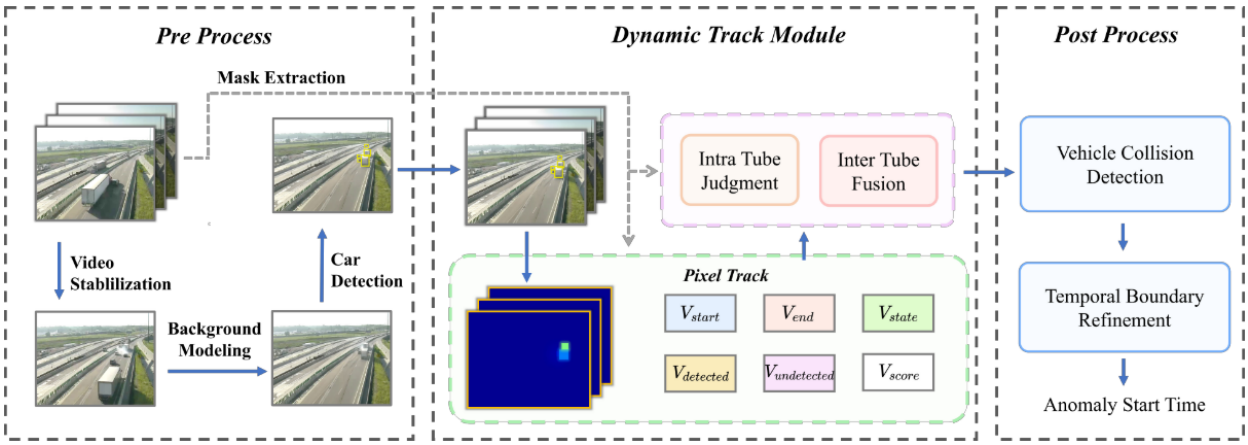


Figure 2.1: Representation of the hand-crafted framework pipeline, consisting of pre-processing, dynamic tracking module and post-processing. (Taken from [7])

The pre-processing step incorporates video stabilization, background modelling, vehicle detection and mask generation.

Since there can be adverse conditions on the camera that may affect the stability of the video, the authors of [7] start by using Digital Video Stabilization (DVS) to correct camera motion oscillations. This is accomplished by estimating the camera movements and proceeding to correct and smooth these movements. They also use a combination of feature point matching using Good Features to Track (GFTT) and estimate the sparse optical flow to generate frame-to-frame transformations.

In order to perform the background modelling they need to distinguish foreground from background, which requires an adaptive background representation, due to the possibility of flawed video sources. This drives them to use background modelling based on the Mixture of Gaussians (MOG). They concluded that background modelling makes stopped vehicles clearer, whilst forward modelling can be used as an auxiliary to get a more precise start time of the anomaly.

Vehicle detection is performed with the use of two two-stage detectors. Their main

detector for object detection was Faster R-CNN [8], whilst the Cascade R-CNN [9] was used as a secondary way to improve the performance of the detection.

Mask generation is used with the goal of preventing stationary vehicles on the side of the road and parking lots from being identified as anomalous events.

The dynamic tracking module has the object of filtering out suspicious events. This is obtained by using the Intersection Over Union (IOU) algorithm to compare the position of the current object with the position that resulted from the detection on the next frame. Resorting to six matrices that record pixel regions and the timestamps of the detections, they then filter true anomalous events.

The post-processing step is responsible for vehicle collision detection and temporal boundary refinement. The anomaly start time of a vehicle crash is at the moment of the crash and not the moment when the car comes to a complete stop. In order to obtain the moment of the crash, the authors of this method discovered that they could obtain the estimated timestamp of when the cars completely stopped and trace the vehicle along the temporal axis to obtain the time of the collision.

This algorithm proved to be effective when detecting traffic anomalous events like car crashes and stationary vehicles, which is only able due to the use of conventional cameras since ECs only detect movement.

2.1.2 Context Cueing Generative Adversarial Network

This approach was proposed in [10] and is based on the use of a Context Cueing Generative Adversarial Network (CC-GAN) to take advantage of the spatio-temporal context and generate regions of interest (ROIs), in order to discriminate irregular ones, which correspond to anomalous events.

The fundamentals of GANs and how they work will be further explained in Section 3.3, by which this section will only provide a superficial explanation of how CC-GANs were developed and used for AED.

The implementation proposed by this approach, demonstrated in Figure 2.2, presents a CC-GAN, a variation of standard GANs, constituted of two generative networks and two discriminative networks, taking into account that the generators are responsible for reconstructing scenes by synthesizing ROIs based on the context clues, whilst one discriminator has the goal to distinguish distorted regions not matching given moving trends, and the other discriminator seeks to discriminate scenes containing incongruous events that conflict

with spatially adjacent surroundings [10].

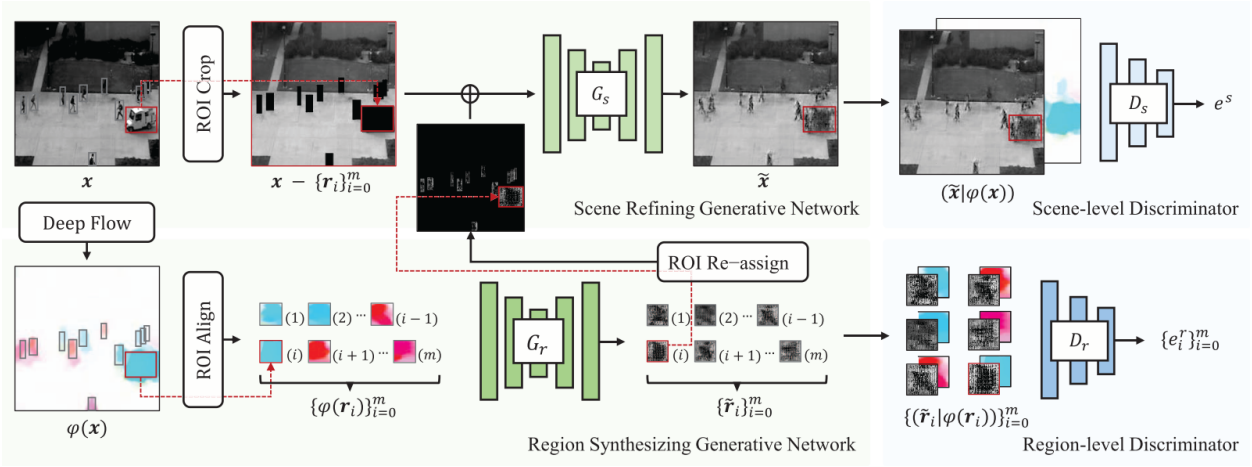


Figure 2.2: Structure of the CC-GAN approach. It shows the two generative networks, the Scene Refining Generative Network and Region Synthesizing Generative Network, and the two discriminative networks, the Scene-level discriminator and Region-level discriminator, along with the pipeline of image manipulation implemented. (Taken from [10])

Generative Networks

The generative networks aim to reconstruct scenes given enough data for them to find the patterns from which they proceed with this reconstruction. In order to do so, the training data sets may only contain normal data, which leads the generators to learn normal patterns to restore normal scenes in the best way possible. As a consequence, during the test phase, when given an abnormal example, the generative networks restore a distorted scene that can be further identified by the discriminators as an anomalous event.

For the generators to produce the desired scenes, Mask-RCNN [11] was used for object detection, providing the bounding boxes for the desired ROIs. With the ROIs obtained, ROI Crop [11] was used with the intent to mask out the pixels inside of the bounding box, resulting in a spatial layout without the objects detected.

The Region Synthesizing Generative Network, G_r, utilises temporal context to estimate the cropped ROIs. Using a two-channel optical flow, ROI Align [11] is employed to crop the ROIs and resize them to match the ROIs cropped from the original scene. These obtained ROIs are fed to G_r as input, resulting in the apparent estimation of the cropped ROIs. ROI Re-Assign is then used to fuse the estimated ROIs provided from G_r with the spatial layout from which the original ROIs were cropped out. The fused canvas can then be assigned to the Scene Refining Generative Network, G_s, as input, resulting in a refined estimated scene,

which takes into consideration the spatial context.

The pipeline of these proceedings can be analysed in Figure 2.2.

Discriminative Networks

The discriminators have the objective of distinguishing whether a scene or region is real or anomalous, given spatial and temporal clues. The Scene-level Discriminator, D_s takes the whole scene into consideration, learning to detect if the inputs fit the data distribution of real scenes. The real/fake scenes used as input for D_s are concatenated with the optical flow obtained from previous scenes. On the other hand, the Region-level Discriminator, D_r, is used to focus on specific ROIs and determine whether they match their temporal context. In order to do so, it concatenates the real/fake ROIs with their corresponding optical flow and uses them as input, resulting in a binary classification.

This implementation is also demonstrated in Figure 2.2.

2.1.3 Decision Trees

Another approach to take into consideration is suggested in [12] and it is based on the use of decision trees to identify anomalous by utilising information from detections on background and foreground images provided by traffic cameras. These detections are obtained with the assistance of a Deep Learning algorithm. Figure 2.3 depicts a representation of the flow chart this method proposes, which starts by using an automated video sorting system on the data, followed by a simultaneous process of detection on foreground images and estimation of background features and potential anomalous events, ending in a decision tree algorithm that detects and separates false anomalies. After identification, the start and end times can be calculated by overlaying the foreground images with anomaly detection.

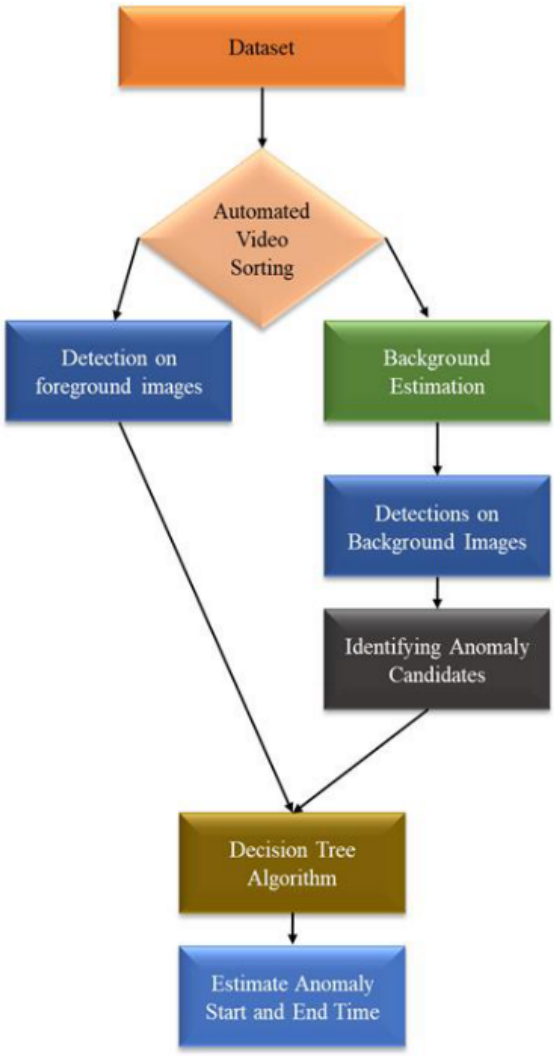


Figure 2.3: Flow chart of the suggested implementation. (Taken from [12])

The vehicle detection model was built using YOLOv5 [13]. YOLOv5 (You Only Look Once) is the 5th version of a convolutional neural network (CNN) designed for real-time object detection, that works by dividing an input image into a grid of cells and predicting the presence and location of objects within each cell. By the time the authors of [12] published the proposed approach, YOLOv5 was the state-of-the-art object detection algorithm.

For video sorting, the authors of this method acknowledged that different traffic conditions may represent different performances in the detection of anomalous events. Therefore, videos were sorted considering the type of road (freeways or intersections), weather conditions (the existence of snow) and the time of day (day or night).

In its turn, anomaly detection was composed of three processes: background estimation, road mask extraction and a decision tree. Since the background estimator learned to obtain the background by masking out all the vehicles in normal conditions if a car vehicle is detected

in a background image, that may be an anomaly. Thus, the candidates for anomalous events were acquired by passing each estimated background image through YOLOv5.

Lastly, the video foreground and background detections are used as input into the decision tree algorithm, illustrated in Figure 2.4. The background detection score and its area are compared to a selected threshold and if greater, an IOU is calculated between the anomaly candidate and the foreground detections. The frequency of overlapping foreground and background detections is used to decide whether it is an anomaly or not [12].

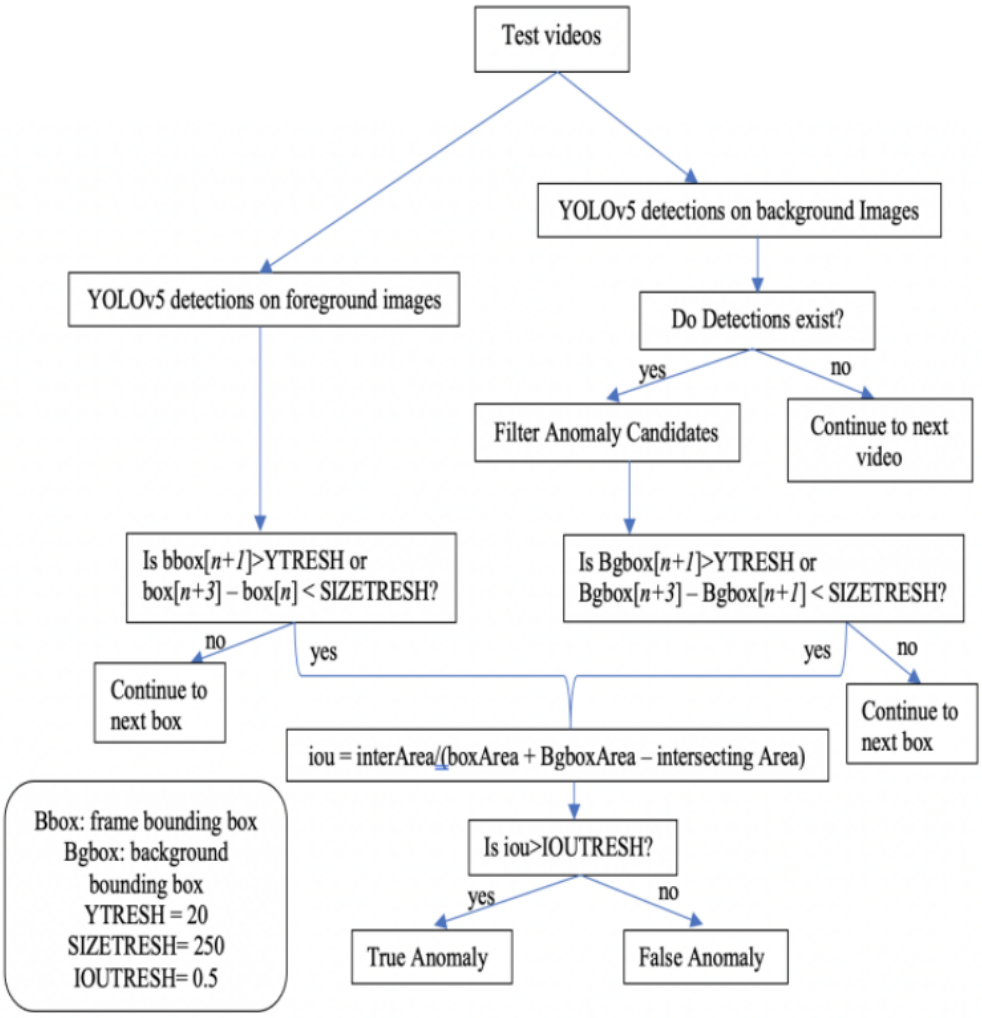


Figure 2.4: Decision tree algorithm for anomaly detection. (Taken from [12])

This methodology proved accurate at AED for anomalies near the camera while presenting some issues detecting distant anomalies.

2.2 AED Using Event-based Data

As the main objective of this dissertation is to implement, validate and discuss an approach to AED that makes use of event-based data, some already existing methods must be presented. Event-based AED, also denominated as neuromorphic vision AED, is still poorly explored, as the first published article [3] about the topic only emerged in November 2019, being subsequently updated in February 2020. This article proposed a GAN-based approach to perform AED. Another study [14] was later published in January 2021 and suggested a different path that makes use of the Optical Flow (OF). Each of these strategies will be further explained.

In the academic year of 2020/2021, Alessio Silva, in his dissertation presented in [2], comprehensively studied and validated the OF-based approach [14] for anomaly detection, amongst other work developed. This motivates this dissertation to study, and validate the GAN-based approach [3], in addition to applying it to traffic context.

It should be noted that although these methods refer to the use of event-based data, none of them makes real use of the sparsity of this data, since a technique is always applied to condense the data into frames, that is, despite the data being from ECs, the algorithms adapt them to be closer to conventional algorithms. In the algorithm on which this dissertation is based, these frames are referred to as Time Surfaces, as it will be explained in Section 4.1.

2.2.1 Optical Flow approach

This technique, designated as the NeuroAED system, was presented in [14] and puts forward a method that relies on the estimation of optical flow, capable of capturing fast-moving entities in a scene. The authors of this method separate the training and testing phases, both consisting of three main stages: optical flow extraction, activated event cuboid selection, and event-based multiscale spatial-temporal (EMST) descriptor generation. These stages are demonstrated in Figure 2.5.

The OF information is first extracted from the training sample and activated event cuboids are selected based on the OF and event density. For each activated event cuboid the EMST descriptor is extracted and fed into sparse representation (SR) models for them to learn standard situations patterns. The trained models are then used to identify descriptors of abnormal patterns extracted from the testing samples [14].

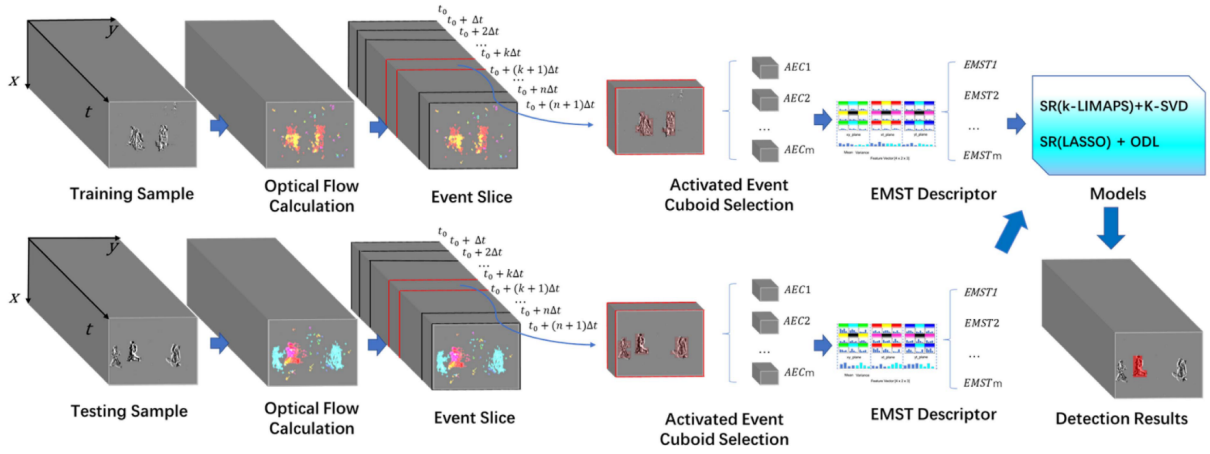


Figure 2.5: Framework of the NeuroAED system. (Taken from [14])

NeuroAED dataset

Easy access to publicly published datasets obtained using conventional frame-based cameras has allowed computer vision algorithms to rapidly develop since it is possible for these algorithms to commit direct comparisons in performance.

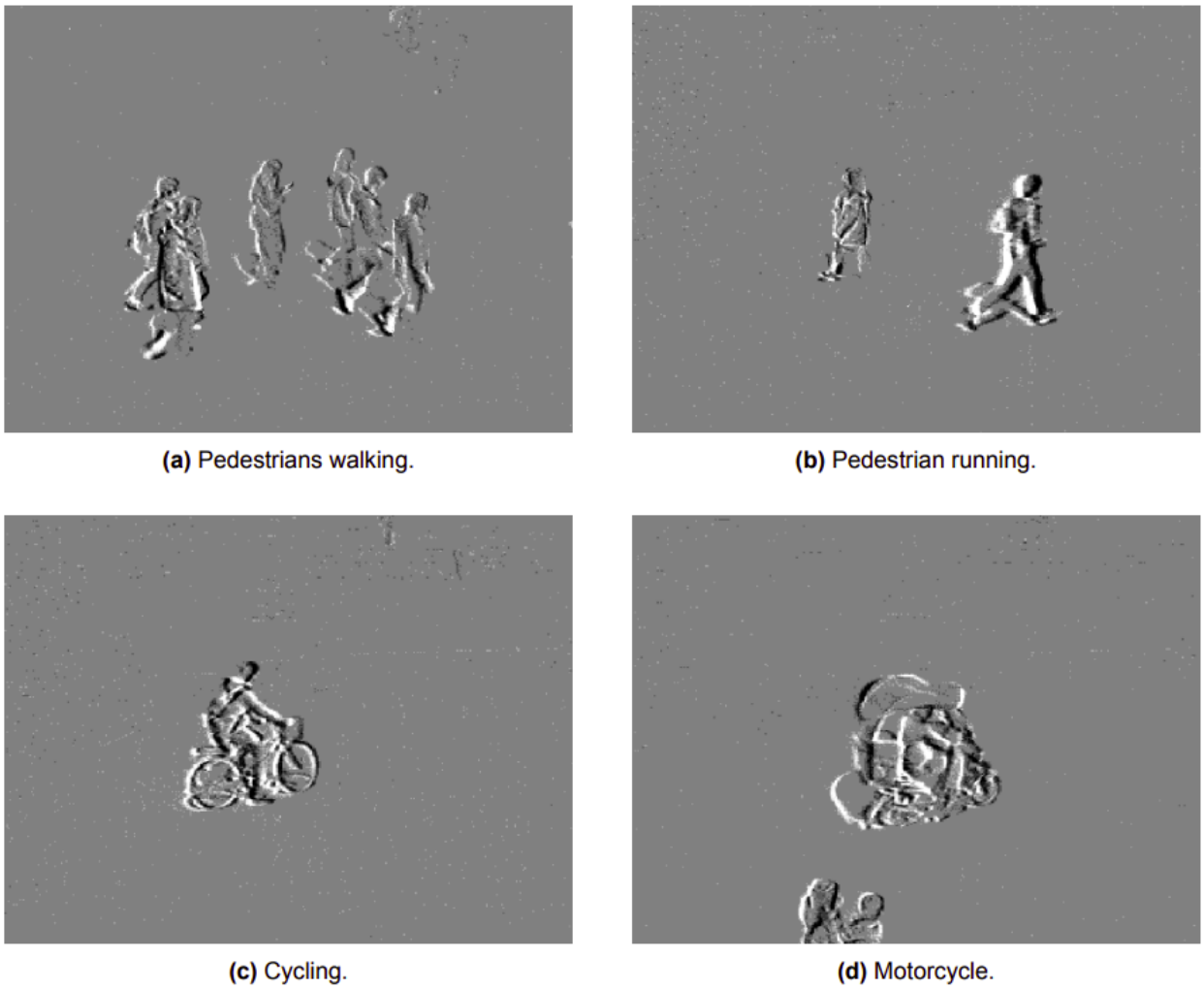
Considering the important role datasets play in the development of abnormal event detection systems and the lack of a neuromorphic vision-based abnormal event dataset, the authors of [14] built the first neuromorphic vision-based dataset dedicated to AED, designated as NeuroAED dataset.

The NeuroAED dataset is composed of 152 samples of four different indoor and outdoor scenarios, being split into four sub-datasets: Walking, Campus, Square and Stair dataset. Each of these sub-datasets presents training and testing samples. While the training samples are only comprised of normal events, the testing samples contain normal and abnormal events. In each slice of the NeuroAED dataset, there is a corresponding ground-truth annotation in the form of a binary flag that indicates whether normal or abnormal events are occurring.

In each sub-dataset, different actions are considered abnormal, for which Table 2.1 presents the normal and abnormal events for each one. Figure 2.6 illustrates the normal and abnormal events referred to in Table 2.1.

Scenario	Normal Events	Abnormal Events
Walking dataset	Walking	Running Bicycle Motorcycle
Campus dataset	Walking	Bicycle Motorcycle
Square dataset	Walking	Scattering
Stair dataset	Walking downstairs	Running Wrong direction

Table 2.1: Description of the events that are considered normal and abnormal in each sub-dataset. (Taken from [14])



(a) Pedestrians walking.

(b) Pedestrian running.

(c) Cycling.

(d) Motorcycle.

Figure 2.6: Normal (a) and abnormal ((b),(c),(d)) events from NeuroAED's Walking sub-dataset. (Taken from [2])

OF-based approach results

For the performance evaluation of the OF-based methodology, two commonly used measurements were adopted: Slice-Level and Pixel-Level.

For the Slice-Level measurement, if one or more cuboids were detected as abnormal cuboids in a testing event slice, it was labelled as an abnormal slice. If the ground truth of this slice was abnormal, it was a True Positive (TP). Otherwise, it was a False Positive (FP).

On the other hand, for the Pixel-level measurement, a detected abnormal slice was TP if more than 40% truly abnormal pixels were detected. A normal slice was FP as long as one pixel was detected as abnormal. Pixel-level measurement accentuates the correct detection of abnormal objects.

The metrics used for each of the above-mentioned measurements were the Area Under Curve (AUC) and Equal Error Rate (EER) based on the ROC curves. Table 2.2 displays the metrics values for each measurement and each dataset.

Scenario	**Slice-Level**		**Pixel-Level**	
	AUC	**EER**	**AUC**	**EER**
Walking dataset	95.8	12.5	87.9	18.7
Campus dataset	85.7	25.5	65.7	38.5
Square dataset	99.7	3.5	-	-
Stair dataset	92.0	15.3	74.9	32.2

Table 2.2: AUC and EER for each dataset using both Slice-Level and Pixel-Level measurements. (Taken from [14])

2.2.2 GAN-based approach

In [3] a GAN-based methodology, named EvAn, is presented for AED. In this method the authors created a DL method that retains the sparsity of the event data, also encoding the temporal information available. They also suggested the use of a GAN variation, which receives the temporal information concatenated with the input, conditioning the output. This GAN variation is designated as Conditional GAN (cGAN).

This procedure was chosen for this dissertation. Hence, it will be thoroughly clarified in Section 4.1.

3 Background Knowledge

This chapter has the objective of providing some required knowledge relating to ECs, the processing of event-based data and GANs in order to fully understand the proposed methodology for this dissertation. An alteration of a simple GAN model is also presented, along with a framework that makes use of it.

3.1 Event Cameras

ECs, firstly known as DVS, are presented in [1] as bio-inspired sensors that differ from conventional frame-based cameras and provide an event-based vision, also known as neuromorphic vision. Simply put, conventional cameras capture the light intensity in a scene, obtaining full images at a fixed rate, while on the other hand, ECs asynchronously capture brightness changes in each pixel independently and output a stream of events. The output of these cameras depends on the amount of motion or brightness changes in the scene, which means that the faster the motion, the more events are produced.

Every time an event is detected in each pixel, the camera outputs its coordinates x_i and y_i, the time, t_i at which the event occurred, and the polarity, p_i of the change in brightness. The polarity is represented by a 1-bit that assumes the value 1 ("ON") if the brightness increases or the value 0 ("OFF") if the brightness decreases.

In comparison with standard cameras, ECs present some characteristics that can be valued as advantages [1], some of them being:

- High temporal resolution and latency, which allows for events to be detected with microsecond resolution, due to the camera's capacity to work each pixel in an individual manner and consequently nullifying the motion blur conventional cameras usually have;
- Low power consumption, since it is only used to process pixels with brightness changes;
- Very High Dynamic Range ($>$ 120 dB), in contrast with the 60 dB of high-quality

conventional cameras, which grants ECs the adequacy to adapt to either very dark or very bright situations;

Figure 3.1 depicts a comparison between the captured output of conventional frame cameras and event-based cameras when recording a moving or stopped dot. In each sub-figure, it is possible to analyse some main differences between these two types of cameras. In sub-figure 3.1a, due to the high temporal resolution presented by ECs, it can capture the dot's movement in greater detail, while the frame-based camera can only capture the dot at a certain frame rate. Since ECs capture motion, as previously stated, when the dot is stopped it cannot capture any movement, therefore it presents no events. On the other hand, the frame-based camera is able to capture the entirety of the image, as expected and demonstrated in sub-figure 3.1b. Lastly, if the dot is moving at a fast pace, traditional frame cameras may experience the occurrence of motion blur, while ECs are not susceptible to this effect, due to their high temporal resolution. This effect is portrayed in sub-figure 3.1c.

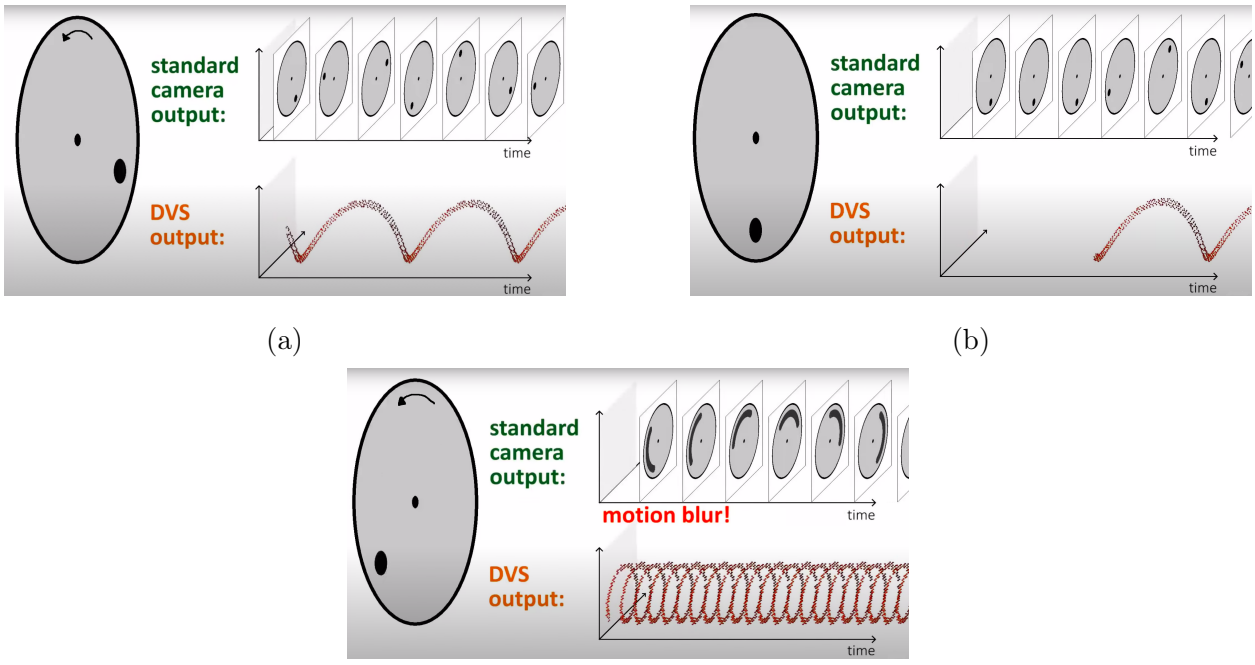


Figure 3.1: Comparison between the output from traditional frame cameras and DVS when capturing a moving dot (a), a stopped dot (b), or a dot moving at a fast pace (c). (Taken from `https://youtu.be/cffwH41ReF4`)

Figure 3.2 shows the difference in output between the two types of cameras when applied to a real context. In this case, the cameras represent the perspective from the inside of a car into the road.

(a) (b)

Figure 3.2: Comparison between the output from traditional frame cameras (a) and DVS (b) when in a real context. (Taken from `https://youtu.be/MjX3z-6n3iA`)

These cameras are able to work in low light conditions, meaning they can be used in different situations in which lighting levels cannot be controlled. Since conventional cameras could not effectively work in more precarious situations, ECs open up a lot of new possibilities for Computer Vision to grow and develop.

Some applications can be referred for ECs [15], such as:

- Liquid monitoring;
- Human tracking;
- Metal process monitoring;
- 3D measurement;
- Vibration monitoring.

The author of [1] also refers that these cameras can be used in real-time interaction systems, in which the above-mentioned characteristics can be revolutionary.

3.2 Event-based Data Processing

As previously stated in Section 2.2, the algorithms presented in this dissertation do not make real use of the sparsity of event-based data. Therefore, some techniques are used to encode the sparse event data into frames, such as:

- **Binning**, which is a simple and widely used technique that divides the data into a fixed number of time bins, and a count of the number of events that occurred within

each bin is recorded. The resulting count vector can be used to represent the data in a compact form and can be used for various analysis tasks, such as trend analysis and anomaly detection.

- **Interpolation**, which is a technique used to estimate values at points where the data is sparse. In this case, interpolation can be used to estimate the number of events that occurred at times between the observed events. This allows for a more complete representation of the data and can be useful for tasks such as prediction and classification.

- **Encoding using temporal windows**, which divides the event data into overlapping or non-overlapping temporal windows, and features are extracted for each window. The features can include, for example, the count of events, the mean and standard deviation of the event timestamps, or other statistical measures. The resulting feature vectors can then be used to represent the data in a compact form and can be used for various analysis tasks, such as classification and clustering.

The main representation of event-based data into frames during this dissertation is the use of Time Surfaces (TSs), which will be described in Section 4.1.1.

3.3 Generative Adversarial Networks

In simple terms, a GAN is an unsupervised deep-learning-based approach for a generative model in which a model learns and finds patterns and certain features in input data, consequently being able to generate new output data in such a way that it can't be determined whether it is from the original dataset or generated by the model.

For a GAN model to properly work two sub-models are required: a generator and a discriminator, trained to compete with each other in a two-player minimax game. Both will be further explained in Section 3.3.3.

GANs were introduced by Ian Goodfellow in June 2014 in [16]. The idea behind GANs comes from the concept of a two-player minimax game, in which two players compete with each other to minimize and maximize a certain loss function. In this case, the loss function is the difference between the synthesised data generated by the GAN and the real-world data.

In order to fully understand the working of a GAN model, the following sections provide the required concepts.

3.3.1 GANs as Unsupervised Learning

The main difference between supervised and unsupervised learning is the necessity for human interaction during the training of a model.

Supervised learning is a machine learning method that uses labelled inputs and outputs in order to learn how to classify or predict data for new and unseen examples drawn from the same distribution, becoming more accurate over time. Two examples of contexts for supervised learning are classification and regression.

Unsupervised learning makes use of machine learning algorithms to learn patterns and distinctive features in unlabelled input datasets. Therefore, these models are not provided with labelled training examples. These models are the ones learning the regularities without the need for human intervention, hence being unsupervised. Unsupervised models are usually applied in clustering and dimensionality reduction.

It must be noted that unsupervised learning still needs labelled data or human interaction in such a way that infers and validates the output data.

A GAN model operates in an unsupervised learning manner since its objective is to learn the common features of the input data and therefore be able to generate new data based on the features it has learned.

3.3.2 Generative Modelling

In order to fully understand what a generative model is, its counterpart, discriminative modelling, must also be comprehended.

Discriminative modelling aims to predict the probability of whether a certain input x belongs to a label y, in such a way that discriminative modelling can be classified as supervised learning, following the above-explained notions.

On the other hand, generative models aim to describe how a dataset is generated, in terms of a probabilistic model, being capable to generate new data from the learned distribution.

From a mathematical point of view, given a set of input data x and a set of labels y, discriminative models estimate the probability of y given x, $(P(y|x))$, while generative models estimate the probability of observing x, $(P(x))$.

3.3.3 Generator and Discriminator Models

Both the generator and the discriminator are usually Convolutional Neural Networks (CNN) and are the essential models on which a GAN is based. The generator's objective is to learn the patterns from the input data and generate new data that could fit the original dataset, whilst the discriminator aims to distinguish whether the data it receives is real, from the original dataset, or fake, generated by the generator. The best description of the relationship between these two models is a game scenario, in which both try to learn and compete, the generator learning to create more realistic samples and the discriminator improving its ability to differentiate real from fake samples [16].

Both models are trained together, one being necessary for the other to improve. The generator starts by generating data from a random noise vector, which is provided to the discriminator, along with examples from the original dataset. The discriminator proceeds to classify the data as real or fake and is then updated to get better at classifying it if needed, and the generator is updated on how well the generated data tricked the discriminator.

Figure 3.3 shows an example of an architecture for a GAN model that follows the training described above. It is possible to observe that the error, relative to the discriminator's ability to distinguish real and fake data, is utilised in both the generator and the discriminator models through backpropagation. In the case of the generator, it has the objective of maximizing the error, whilst the discriminator tends to minimize it.

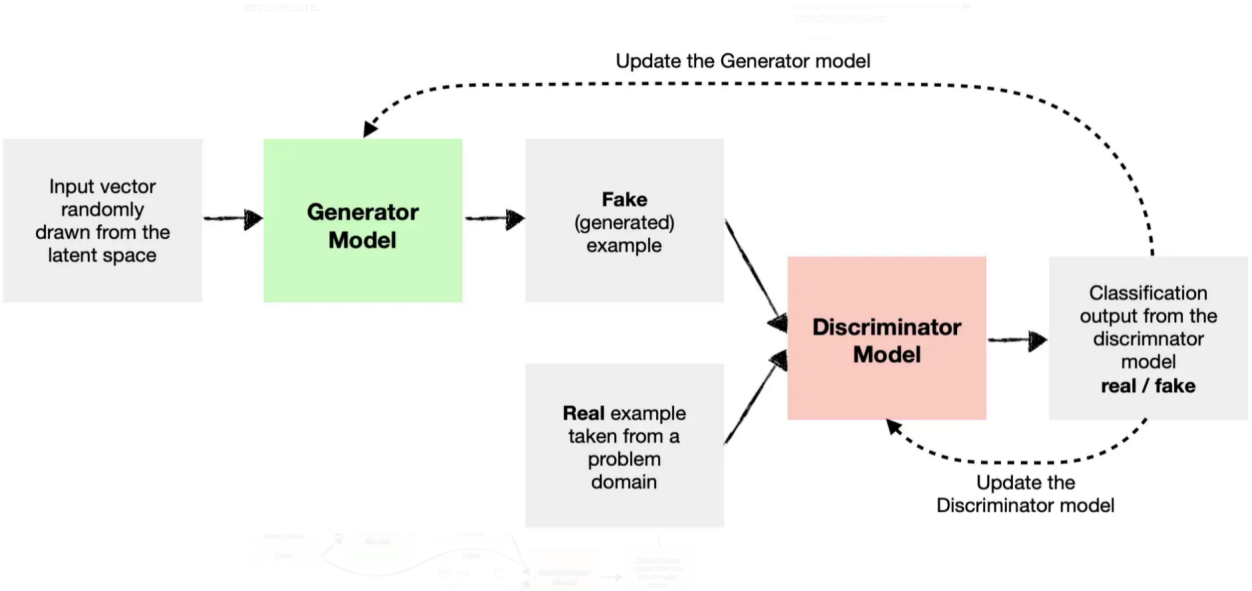


Figure 3.3: Abstract example of a GAN architecture. (Taken from `https://towardsdatascience.com/`)

From a mathematical analysis provided by [16], to learn the generator's distribution p_g over data x, previous input noise variables $p_z(z)$ are defined and mapped to data space as $G(z;\theta_g)$, in which G is a differentiable function defined by a model with parameters θ_g, representing the generator model. Another model, the discriminator model, is defined as $D(x;\theta_d)$, which outputs a single value. $D(x)$ represents the probability that x came from the real data instead of p_g. The model D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. t the same time, G is trained to minimize $log(1-D(G(z)))$. This training is represented by the following two-player minimax game with value function $V(G,D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{3.1}$$

In an ideal situation, the generator creates perfect samples from the real dataset, in which the discriminator can't classify with certainty each sample. This leads to a point of convergence where the discriminator classifies about half (50%) of the times, analogously to a coin flip, which means that the model cannot tell the difference based on any factor and is forced to guess, converging to a 50% performance.

Since the main objective of a GAN is to generate data that can be perceived as real, once the training is complete the discriminator is not needed anymore and therefore discarded.

3.3.4 PatchGAN Discriminator

Different models of neural networks can be used as generators or discriminators of GANs based on the required application for it. A commonly used variation of a simple CNN discriminator is the PatchGAN [17] classifier.

A regular classifier classifies the entire TS as real or fake. A PatchGAN, on the other hand, separates the image into tiles and predicts whether each tile is real or fake. This classifier works by generating an NxN output array, in which N represents the number of tiles into which the TS is divided. The entire TS is declared as real or fake based on the average of classifications for each tile.

An abstract example of PatchGAN can be analysed in Figure 3.4, in which a batch of an image is selected, processed through the convolutional layers and the result of the classification is output into an array.

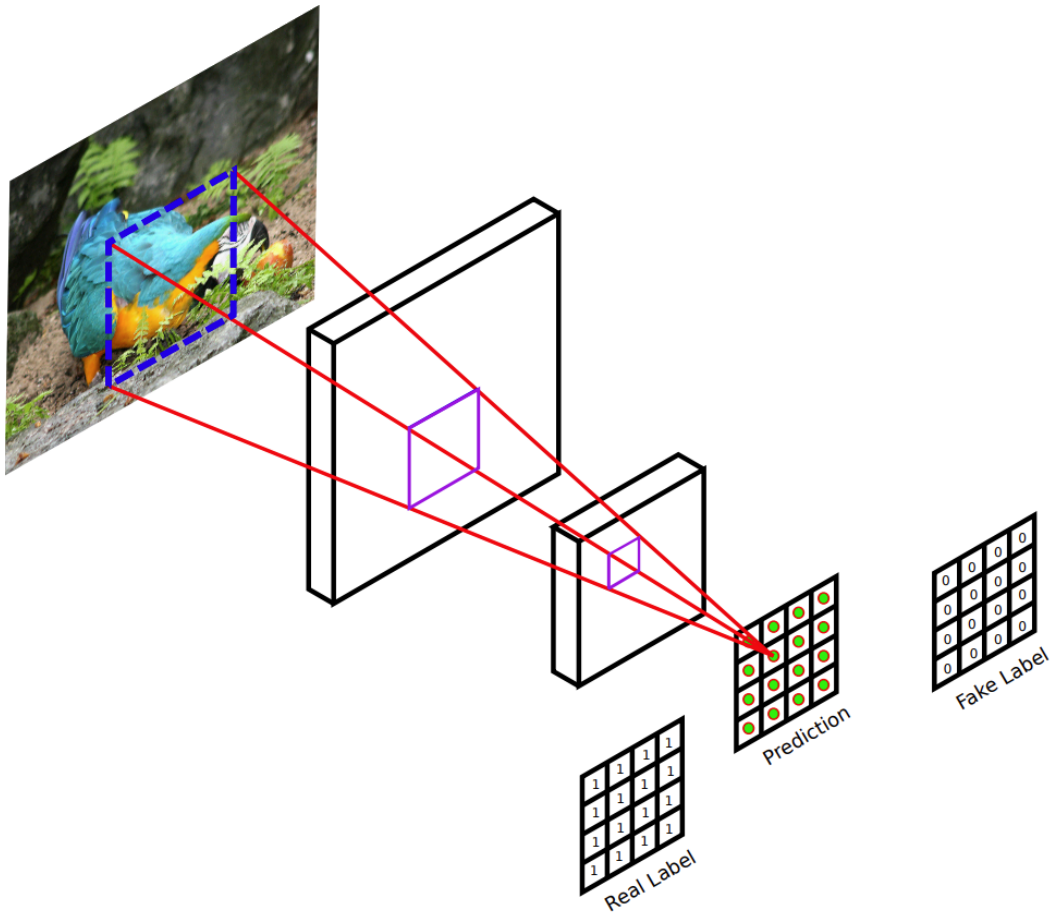


Figure 3.4: Abstract example of a PatchGAN network. (Taken from [17])

The main advantage of a PatchGAN classifier is that it can be applied to tasks with details in sharp, high-frequency variations. By operating on tiles, the classifier learns to focus on the fine details in the TSs, therefore providing more detailed and realistic feedback to the generator network, which leads to more realistic generated TS.

3.3.5 GANs in Real-World Applications

Since GANs appeared, due to their ability to generate data that is similar to a given training dataset without supervision, the number of applications for these networks has been expanding. Although there are many applications for GANs, a few examples of the tasks they are capable of are:

- Image generation: GANs can be used to generate realistic images of objects, people, landscapes, and other subjects, becoming difficult for humans to distinguish the generated data from real photographs.
- Text generation: GANs have been used to generate text that is similar to a given

training dataset, allowing for language translation and text summarising.

- Audio generation: GANs can be used to generate realistic audio samples, including music and speech.

- Anomaly detection: GANs can be used to detect anomalous events in data, such as fraudulent transactions or malfunctioning equipment.

- Data augmentation: GANs can be used to generate additional training data for other machine learning models, allowing them to improve their performance on a given task.

As will be further explained in Section 4.1, the main application for the GAN architecture proposed in this dissertation is based on image generation.

3.4 Conditional Generative Adversarial Networks

As the use of GANs develops, different variations have surfaced, each with its own advantages and disadvantages, depending on the required task. Some of these variations are Conditional GANs (cGANs), Adversarial Autoencoders (AAEs), Wasserstein GANs (WGANs), StyleGANs, Bidirectional GANs (BiGANs) and CycleGANS.

The main focus of this dissertation will be cGANs, as it is the approach proposed in [3]. The introduction of cGANs was published in November 2014 in the article [18], proposing a way of conditioning the GAN model on additional information, making it possible to direct the data generation process.

In order to expand the GAN model into a conditional one, both the generator and the discriminator are conditioned with some extra information y, known as labels. In the case of the generator, this information y is concatenated with the random noise input $p_z(z)$. In the case of the discriminator, it takes both the real data x and the condition y as inputs to a discriminative function. These changes result in an alteration of the Equation 3.1 into:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z|y)))] \tag{3.2}$$

Figure 3.5 shows an example of an architecture for a GAN model that follows the training described above.

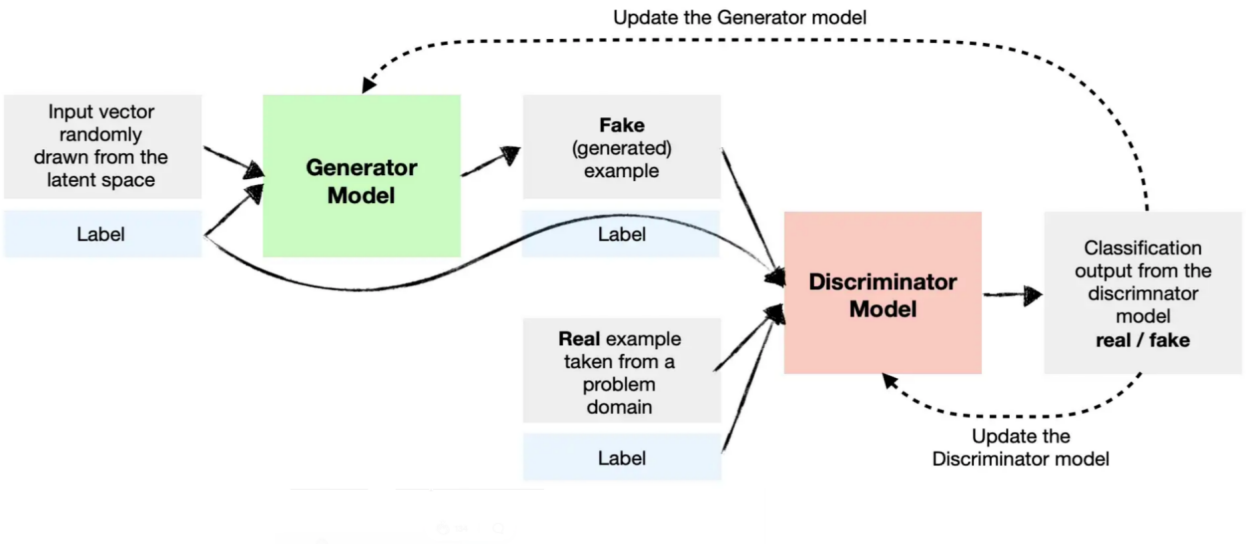


Figure 3.5: Abstract example of a cGAN architecture. (Taken from `https://towardsdatascience.com/`)

3.5 Image-to-Image Translation with cGANs

This section aims to present a model introduced in [4] used for image-to-image translation tasks, named Pix2Pix. Image-to-image translation has the goal to transform one image into another that has a similar structure and content, such as the transformation of a sketch into a coloured image or a daytime image into nighttime.

Figure 3.6 displays some examples of image-to-image translations with the Pix2Pix framework. It can be observed that, as already stated, the structure and content remains the same between the input and output in each example, with the input being either a real image or an abstract representation of what the output should be.

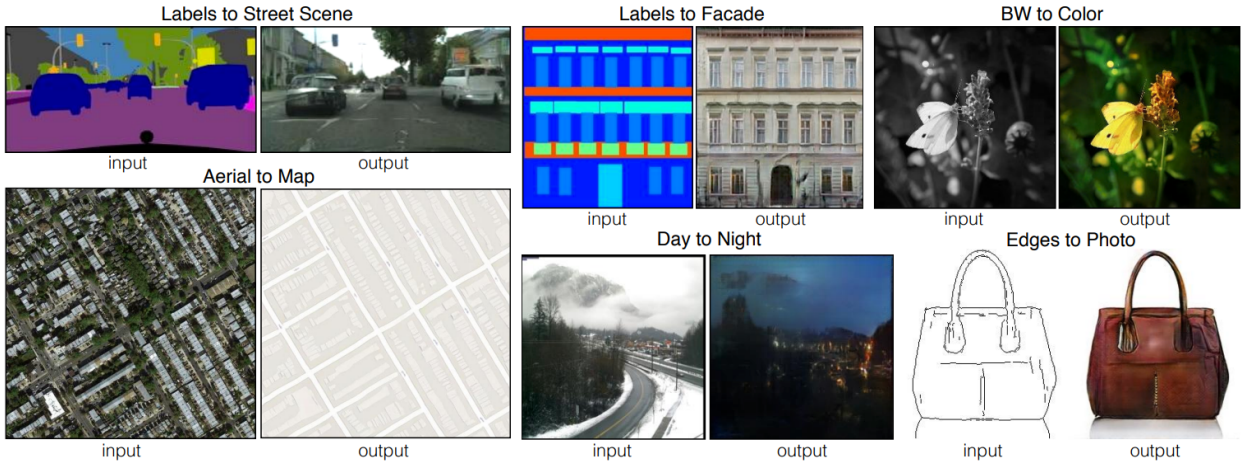


Figure 3.6: Examples of the Pix2Pix model results when applied to different situations in image-to-image translation. (Taken from [4])

The Pix2Pix model employs a cGAN as the network responsible for image generation. Although different models for the generator and discriminator can be used, the most robust ones are the U-Net as the generator and PatchGAN as the discriminator.

The U-Net [19] architecture is a type of encoder-decoder commonly used in image-to-image translation and image segmentation tasks. The U-Net, just like a normal encoder-decoder, consists of an encoder responsible for passing the input through a series of convolution and pooling layers, which reduce the spatial dimension of the image, downsampling it into a lower dimensional representation until it reaches a bottleneck layer. This representation contains high-level semantic information about the input image, such as object shapes, edges and textures. After the encoder, the decoder is in charge of upsampling this representation back into the output image.

The U-Net varies from normal encoder-decoders by adding skip connections from the encoder layers to the corresponding layers in the decoder. These skip connections allow the U-Net to preserve the spatial information and fine details of the input image by permitting the decoder to access both the high-level semantic information from the decoder and the low-level information from the input image. The skip connections are implemented by concatenating the activation functions from the encoder and the corresponding layer in the decoder.

Figure 3.7 demonstrates the structure of a U-Net network.

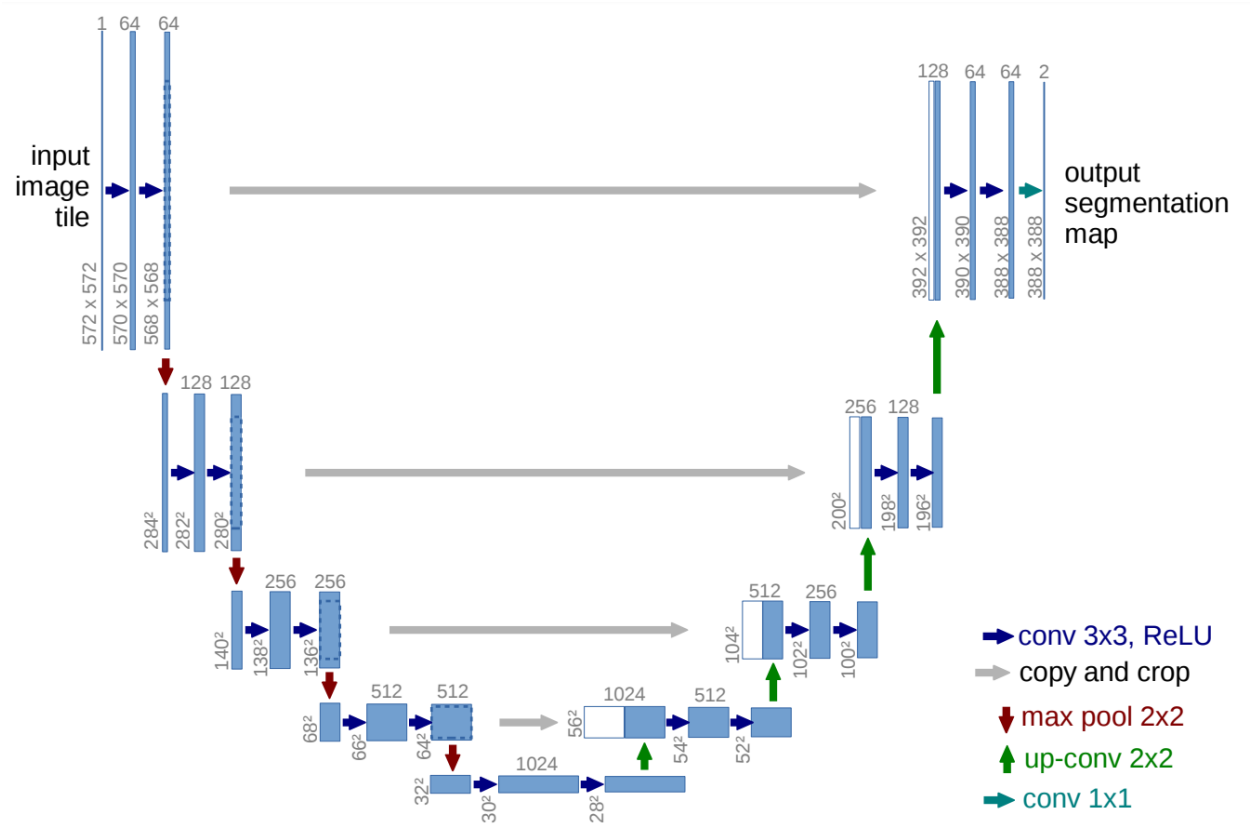


Figure 3.7: U-net architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. (Taken from [19])

The PatchGAN classifier used as the discriminator was already described in Section 3.3.4.

The Pix2Pix model is trained by using a combination of adversarial loss and L1 loss. The adversarial loss prompts the generator to produce visually plausible images which are difficult for the discriminator to distinguish from real images, while the L1 loss boosts the generator to create images that are as close as possible to the target output image.

Some of the already existing applications for this model are: labels into a photo, map into an aerial photo and otherwise, black and white photo into a coloured photo, edges or sketch into a photo, day into night and otherwise, thermal photo into a colour photo.

These applications are depicted in Figure 3.6.

4 Material and Methods

This chapter aims to present the proposed methodologies for the development of this dissertation. A cGAN network is proposed along with a Memory Surface network to encode the temporal information of a series of event-based frames.

4.1 Conditional GAN for Anomaly Detection

As it was previously stated in Section 2.2.2, the proposed methodology for this dissertation was published in [3] as the first baseline for AED using event-based data and suggests anomaly detection as a conditional generative problem, making use of a cGAN network composed of sparse submanifold convolution layers, trained to predict a future TS conditioned on the current DL MS. This DL MS is used to adapt the event-based data as input to the cGAN model, retaining the sparsity of the event data frames and also encoding the temporal information.

The authors also provided an event-based anomaly detection dataset, named as AnoDataset.

The pipeline for the EvAn framework is illustrated in Figure 4.1, in which is possible to see that firstly the DL MS network has event volumes, TS, as input and output. The desired DL MS is generated and extracted from the bottleneck layer and fed into the generator network as a conditioning input. This network results in the prediction of the future TS, which is then fed, along with the DL MS as a conditioning input, into the discriminator. The discriminator network is exposed to the real TS as well, also conditioned with the DL MS, resulting in the classification as real or fake.

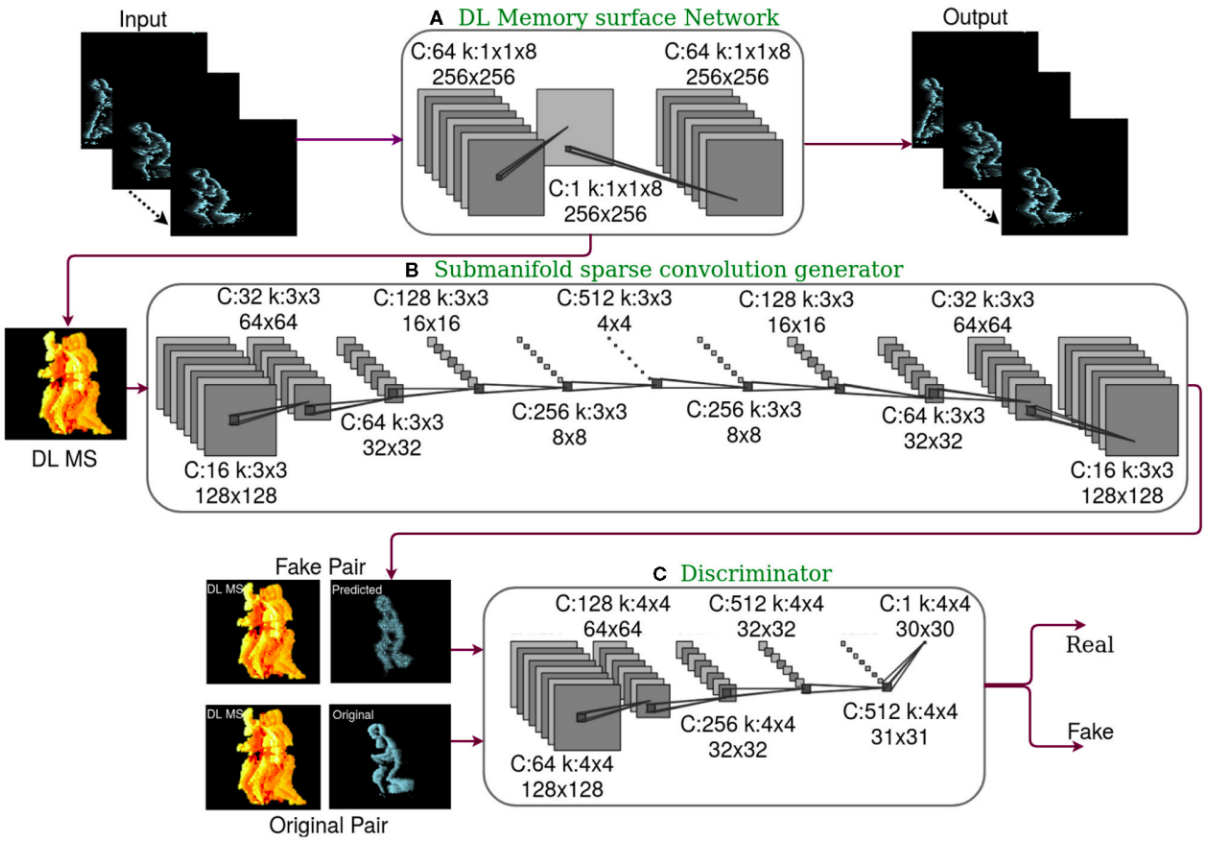


Figure 4.1: Pipeline of the EvAn framework. C refers to the number of channels per layer and k represents the kernel. (Taken from [3])

4.1.1 DL Memory Surface Generation Network

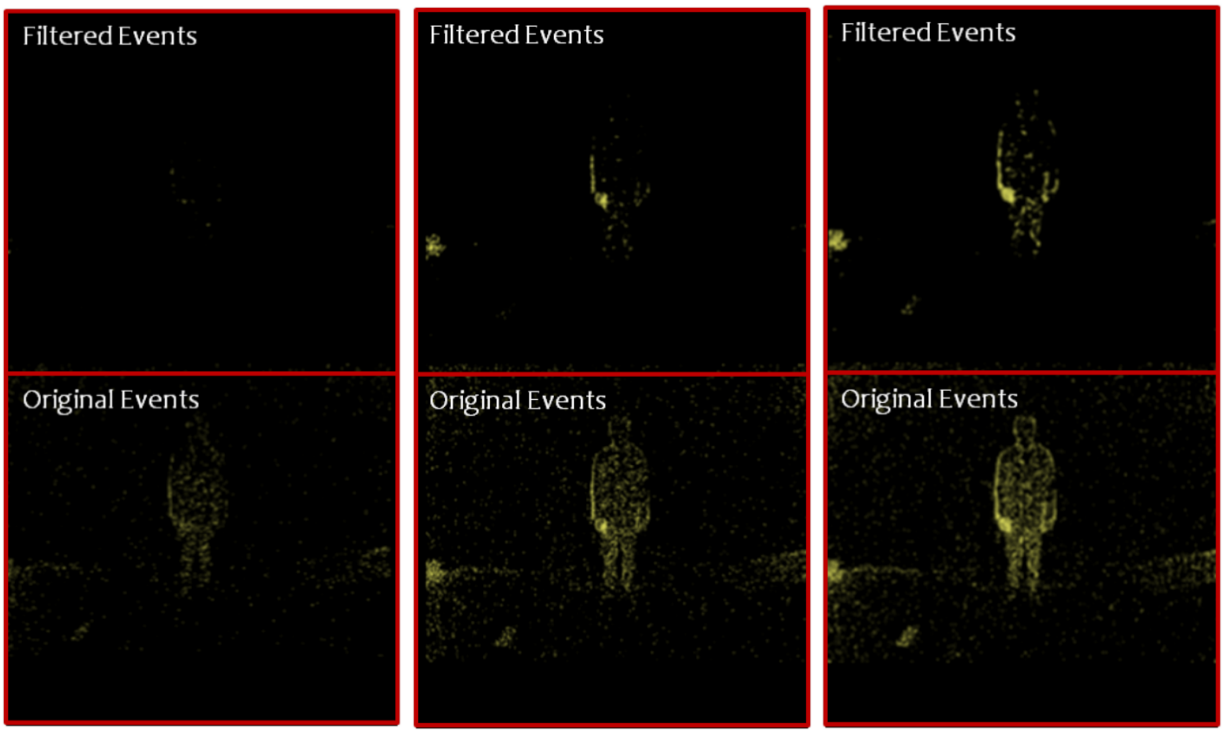
This section explains in greater detail the proposed DL MS network, an unsupervised learning-based network that produces event data memory surfaces, which retains the sparsity of the event data and encodes the temporal information available.

Time Surfaces

TSs [1] are 2D maps where each pixel stores a single time value in which events are converted into an image that represents intensity as a function of the sequence of discrete events that occur at specific points in time at that location, with larger values corresponding to a more recent motion. The TS is constructed by plotting each event as a point in a 2D space with the timestamp on the x-axis and the event feature on the y-axis.

The TSs are formed by accumulating the timestamp of events for a certain duration ΔT. Due to noise effects from the camera, the authors of [3] were required to perform pre-processing on the event data, which led them to note that a smaller ΔT would lead to the retaining of no information. In Figure 4.2 it is shown the discretized event slices for ΔT

of $10ms$, $30ms$ and $50ms$. This guided them to conclude that a ΔT of $50ms$ provided an optimum trade-off between temporal latency and information content.



(a) (b) (c)

Figure 4.2: Discretized events (bottom) and noise filtered events (top) accumulated over ΔT of $10ms$ (a), $30ms$ (b) and $50ms$ (c). (Taken from [20])

In Section 5.1 a validation of the optimal ΔT is studied and analysed.

Memory Surfaces

MSs make use of the event data sparsity present in the TSs to represent the temporal structure of the events. In the proposed approach, batches of eight TSs are used to provide input to the DL MS network, which, since each TS presents a ΔT of $50ms$, results in a time duration of $400ms$. MSs are extracted from the bottleneck layer of the network. Analysing Figure 4.3 it is possible to examine that the DL MS captures the information regarding the history of event data. The examples used in the figure, bending activity and running activity, come from an action recognition dataset the authors used to evaluate the performance of the DL MS network.

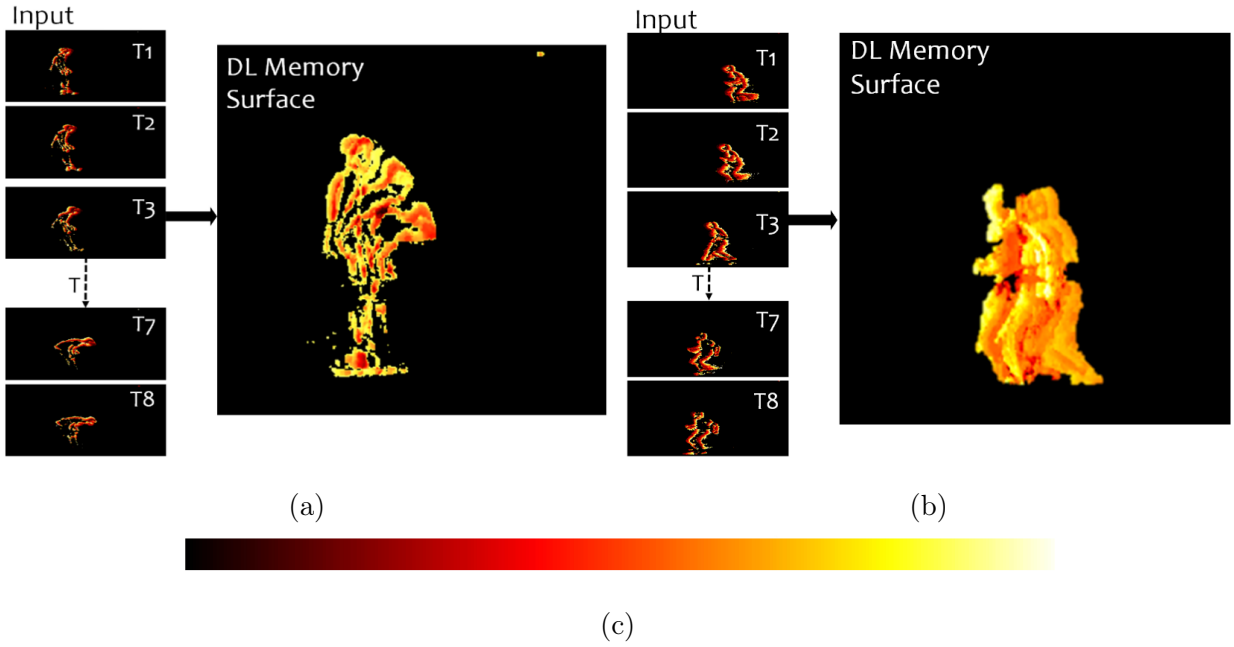


Figure 4.3: Discretized events (left) DL memory surfaces (right) for bending (a) and running (b) activities. A colormap (c) is used for better visualization. (0: Black, 255: Yellow) (Taken from [21])

Some examples of the obtained MSs during the development of this dissertation can be observed in Section 5.2.

Network Architecture

In simple terms, the DL MS network aims to receive as input a discretized volume of events provided by a batch of TS, learn to encode the important features of the volume of events into a single image at the bottleneck layer, and decode the information in that image to re-create the input data, as one can see on the top row of Figure 4.1.

This network presents a fully convolutional encode-decoder architecture. As already mentioned and presented in [3], the discretized volume of event data ($Ev = [ev_0, ev_1, ..., ev_B]$) is produced by stacking events into TS (ev_i), given a time duration T and a set of B discrete time bins $[b_0, b_1, ..., b_B]$, each with ΔT duration.

For the temporal history of the data to be preserved without altering the spatial distribution, the convolution operation is restricted to the time dimension, by which a convolution of 1x1 is performed, known as 1D convolution.

The encoder convolution layer presents 64 channels and the above-mentioned 1D convolution, followed by the bottleneck convolution, from which the DL MS is extracted, and ending in the decoder convolution layer with 64 channels.

Loss Function

The network tries to learn a function $h_{\theta_{MS}}(Ev)$ in a manner that the output values $[\widehat{ev_0}, \widehat{ev_1}, ..., \widehat{ev_B}]$ are similar to the input values $[ev_0, ev_1, ..., ev_B]$, while the bottleneck learns to model the temporal information. The encoder and the decoder are defined by the transformation functions $\phi_{\theta_E} : Ev \rightarrow MS$ and $\psi_{\theta_D} : MS \rightarrow EV$, with θ_E and θ_D representing the parameters of the encoder and the decoder, respectively.

Simply put, this means that the decoder aims to output values as similar as possible to the values used as input for the encoder.

To maximize the ability of the latent variable encoding, the authors use a data term that tries to model the probability distribution $[\mathbb{P}(Ev|MS^*)]$ of getting the event discretized volume, Ev, given the ideal DL memory surface, MS^*, by maximizing the forward Kullback-Leibler (KL) divergence between the ideal distribution $\mathbb{P}(Ev|MS^*)$ and their estimate $\mathbb{P}(Ev|\widehat{Ms})$, resulting in Equation 4.1.

The authors learned that Forward KL divergence results in the best latent variable. Forward KL divergence is a measure that calculates the difference between two probability distributions.

$$KL = \mathbb{E}_{Ev\sim\mathbb{P}(Ev|MS^*)}\, log[\mathbb{P}(Ev|MS^*)] - \mathbb{E}_{Ev\sim\mathbb{P}(Ev|MS^*)}\, log[\mathbb{P}(Ev|\widehat{MS})] \tag{4.1}$$

The output of the decoder can be modelled as a function of latent variable $\widehat{MS}$ and noise $\eta \sim \mathbb{N}(0, 1)$ as $\psi_{\theta_D}(\widehat{MS}) + \eta$.

The only term of the equation that depends on the estimated latent variable is the second one, which means the first term can be discarded. This leads the equation to result in maximizing the log-likelihood of $\mathbb{P}(Ev|\widehat{MS})$, which changes to minimizing $-\|Ev - \psi_{\theta_D}(\widehat{MS})\|^2$.

The authors also introduce an activation regularization term that makes use of a combination of both $L1$ and $L2$ regularization of activations in the bottleneck layer, in order to maintain the sparsity of the event data.

4.1.2 Sparse Convolutional cGAN Network

The sparse convolutional cGAN architecture proposed for the EvAn framework comprises a sparse convolutional generator and a conditional discriminator made up of convolution-BatchNorm-ReLU. The structure cGANs are based on was already explained in greater detail in Section 3.4, as well as their minimax optimization function in Equation 3.2.

Submanifold Sparse Convolution Generator

The generator network makes use of sparse convolutional encoder-decoder architecture. Its objective is to predict the future TS conditioned on the DL MS. The generator is designed to be computationally efficient by using pooling layers to decrease the spatial dimension, increasing the number of channels as it progresses.

However, the decrease in the spatial dimension may lead to a loss of the spatial structure of the sparse data. This motivated the authors to utilise Submanifold Sparse Convolutions (SSC) convolutions. SSC convolutions were presented in [22] as an optimised solution for the use of spatial sparsity present in the event data since these convolutions are computed only on pixels termed as active sites. A pixel is only in an active site if the central site of the used kernel is non-zero.

The blocks used for each layer of the generator are Minkowski layers [23], composed of Convolution/Deconvolution-Pooling-Activation, wherein the activation functions are used only on active sites and the deconvolution operations are defined as the inverse of the SSC convolution operation.

The encoder produces a batch of 512 feature maps, which are then fed to the decoder in order to reconstruct the input through deconvolution and unpooling layers in the reverse order of the encoder, as one can see in the middle row of Figure 4.1.

Discriminator

For the discriminator network, the authors implemented a PatchGAN [17] classifier, instead of a regular one. This classifier was already presented in Section 3.3.4.

The ability this classifier presents of dividing the output into separate tiles was one of the motives for the authors of the EvAn framework to choose it as the discriminator network since it enables the model to gain additional information in each frame, such as the estimation of the area of the frame where an anomalous event is located, as determined in Figure 4.4.

The discriminator network applied to the EvAn framework can be observed in the bottom row of Figure 4.1, from which is possible to deduct a classification array with size 30x30 was chosen for the output of the PatchGAN classifier.

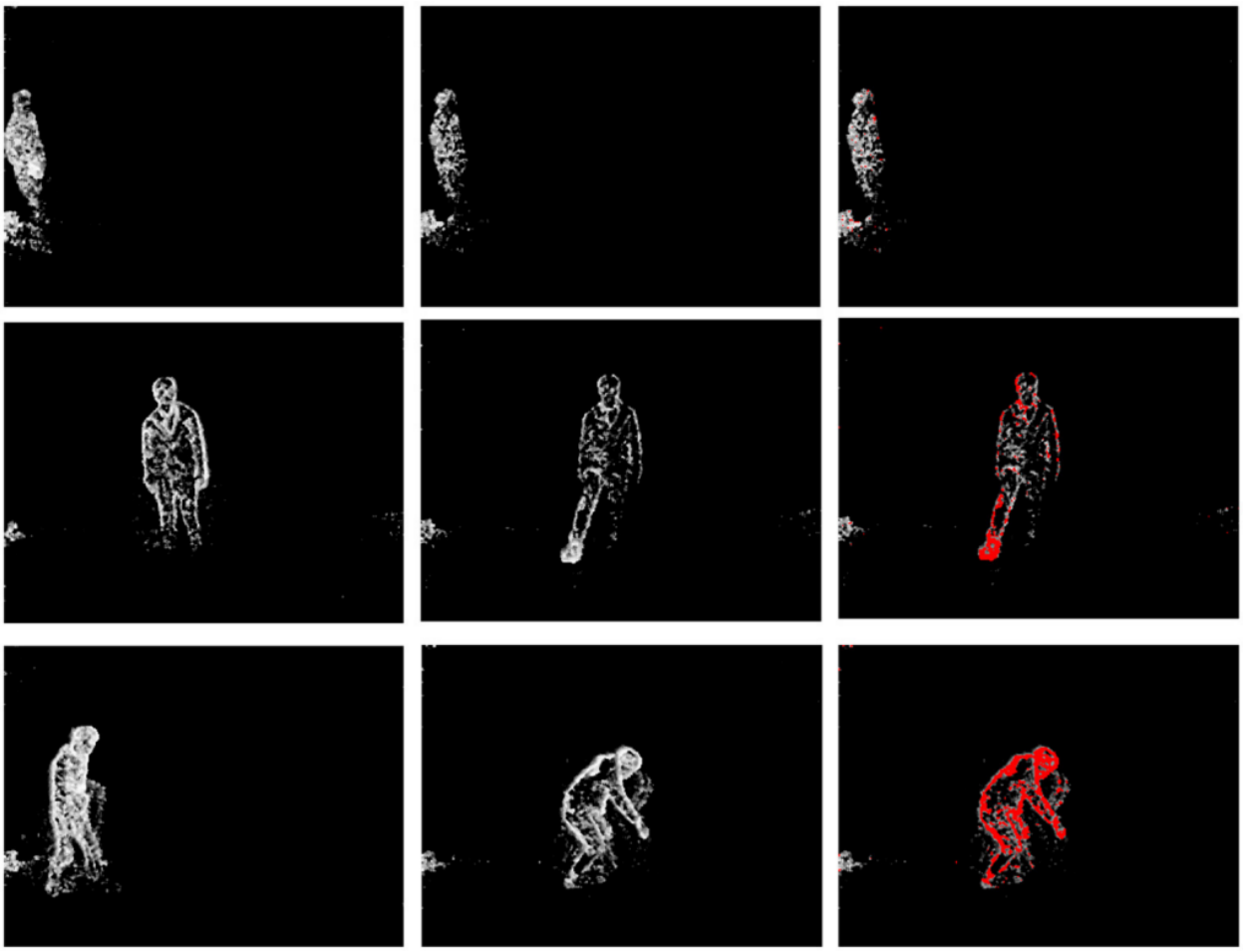


Figure 4.4: TS for three different activities (one for each row). In red are represented the zones in which a certain threshold is exceeded. (Taken from [3])

5 Developed Work

This chapter addresses the developed work in order to fulfil the main objectives proposed for this dissertation. It contains the implementation for the DL memory surface generation network and the Conditional GAN network, as well as a detailed explanation of the required preprocessing for each dataset.

As the pipeline for the selected methodology demands and due to its significance in the training and validation of DL models, the pre-processing of the datasets was the first step. Afterwards, the DL MS network was the first DL model to be implemented since it is required for MSs to be obtained in order to provide input for the cGAN network.

5.1 Datasets Pre-Processing

Datasets perform an important role in the development of computer vision algorithms and deep learning models since they allow models to be trained to recognise and understand images and videos. Typically, datasets include a large number of images, in which diversity is a critical factor for models to learn patterns and generalise to new data. During the development of the proposed methodologies for this dissertation, datasets that allow the models to learn to identify anomalous events are required.

The first step was to obtain and process datasets for the required task. Therefore, similarly to [3], ActDataset and PedDataset were utilised although they were not built with the objective of AED. Thus, some pre-processing was required, which demanded manual separation of the frames containing normal situations from anomalous events.

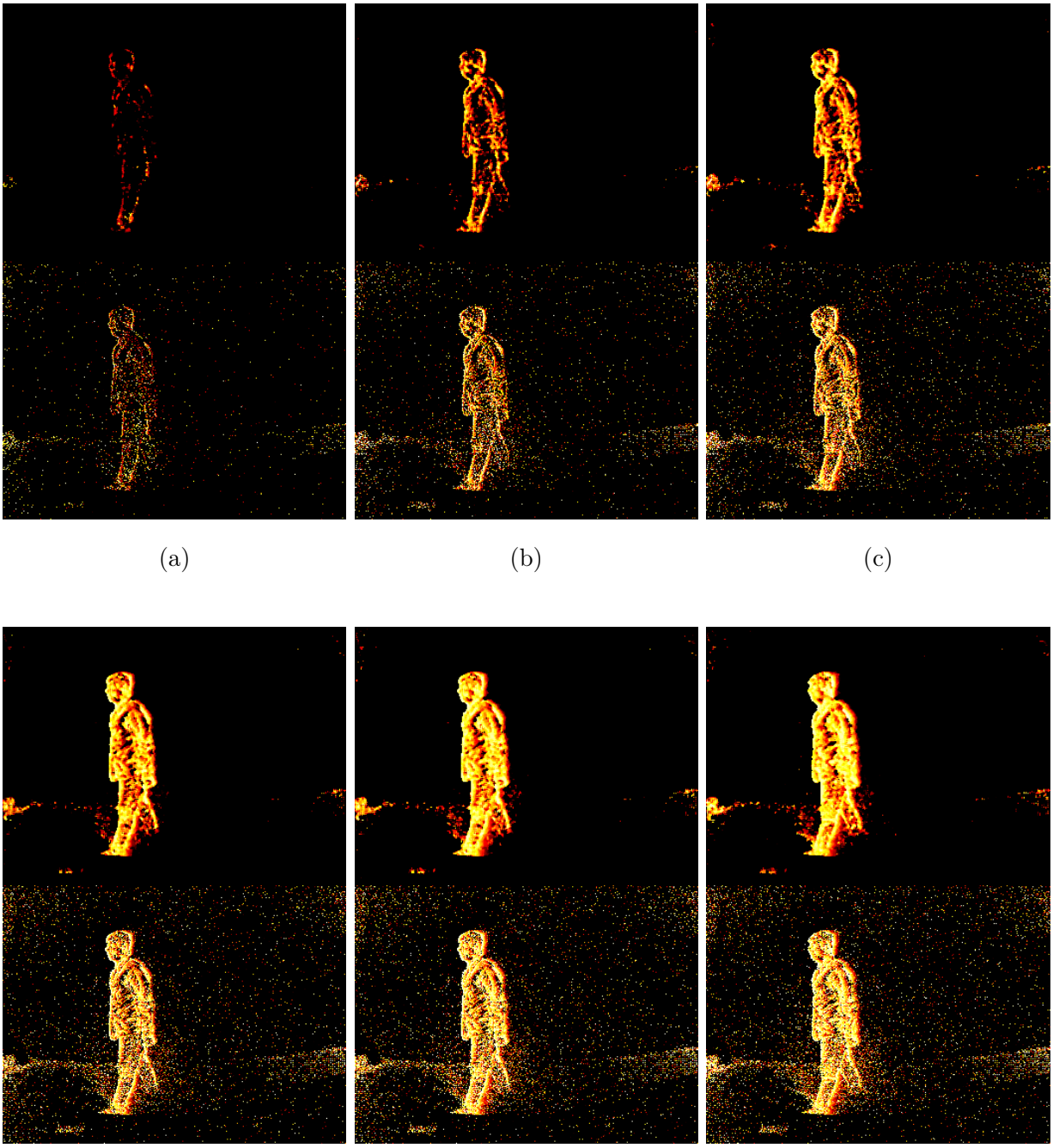
Both datasets were introduced in [24] and were captured using a Davis sensor with a resolution of 346x260, in which ActDataset is a dataset conceived for action recognition that presents 450 recordings with an average of 5 seconds each and PedDataset is a dataset targeted for pedestrian detection and is composed of 12 recordings with an average of 30 seconds each.

The datasets are provided through *.aedat* files, which is a file format to store data provided by ECs. These files encode the event-based data, representing each event by a timestamp and a set of (x, y) coordinates that indicate the location of the event on the EC. Therefore, the chosen technique to decode the data and visualise the events was the Surface of Active Events (SAE), permitting the TSs to be obtained.

As previously stated, TSs are formed by accumulating the timestamp of events for a certain duration ΔT and the authors of [3] concluded that a ΔT of $50ms$ provided an optimum trade-off between temporal latency and information content.

However, a validation for the optimal ΔT was needed using the chosen decoding technique. Figure 5.1 depicts this validation for the same timestamp whilst using different accumulation times ($10ms$, $30ms$, $50ms$, $70ms$, $90ms$, $100ms$) before and after applying a median filter to the event data frame. The display of these images used a colour map in order to easily watch the data and its depth.

It became possible to conclude that for an accumulation time of $10ms$, too much information is lost, becoming an unreliable choice. As for the times of $70ms$, $90ms$ and $100ms$ the retained information is excessive even after the filter was applied to the image, which permitted for these options to be discarded. From the two remaining possibilities, although an accumulation of events of duration $30ms$ presents less noise when filtered, some intensity on the edges and information may be lost. Thus, as stated in [3], the experimental validation proves a ΔT of $50ms$ to be optimal.



(a) (b) (c)

(d) (e) (f)

Figure 5.1: Discretized events (bottom of each sub-figure) and noise filtered events (top of each sub-figure) accumulated over ΔT of $10ms$ (a), $30ms$ (b) and $50ms$ (c), $70ms$ (d), $90ms$ (e), $100ms$ (f).

5.1.1 ActDataset

After analysis of the data available in the ActDataset, it was inferred that for each action, there is a total of three directions in which the action is recorded. That is, there is a recording of each action to the right and to the left, both parallel to the camera plane, and

another recording towards the camera. In turn, different people were recorded performing these actions in each direction. Figure 5.2 displays an example of the *Walking* action for each direction.

It can be deduced beforehand that the generator will be unable to generate data when trained with recordings made towards the camera, as it is possible to observe in the bottom row of Figure 5.2 that the information in consecutive frames does not present a substantial change.

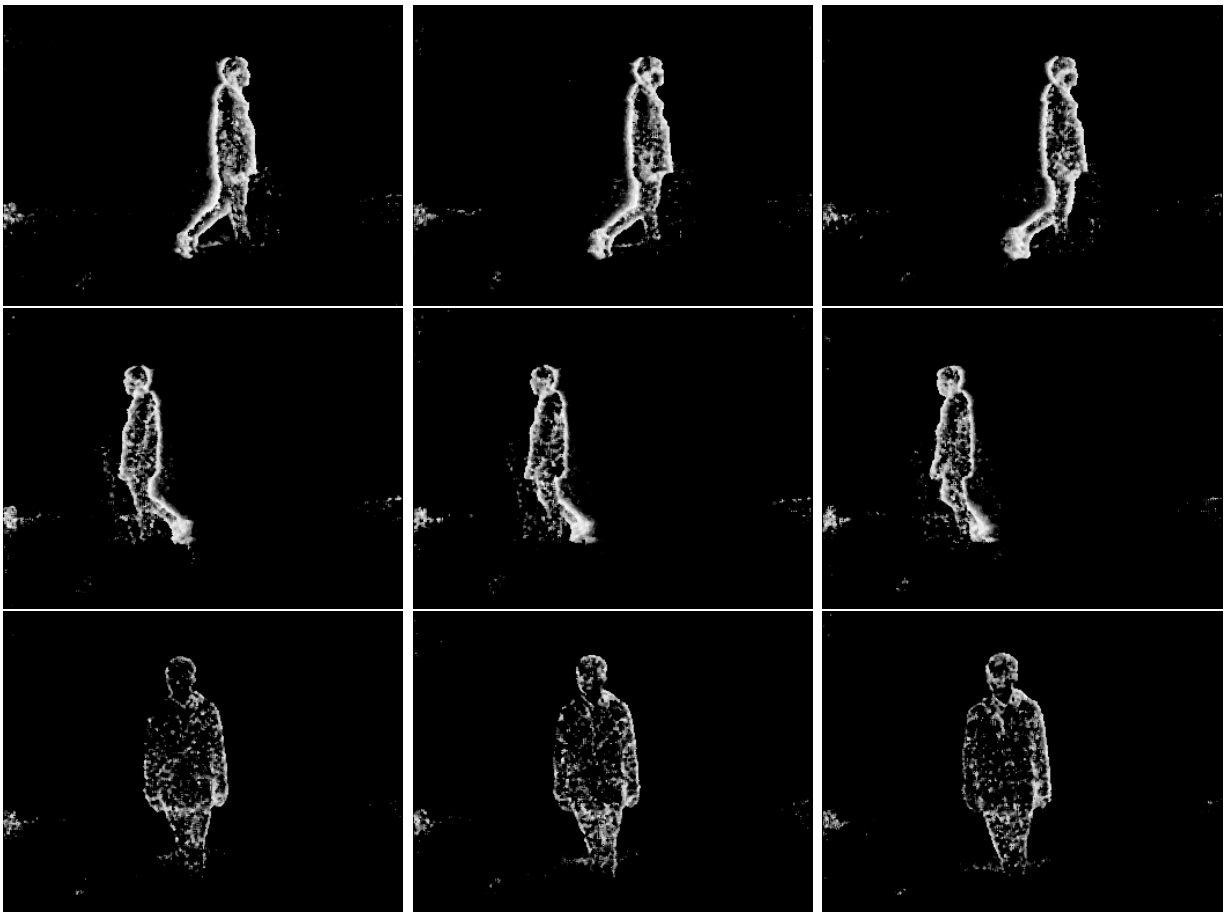


Figure 5.2: Consecutive frames of ActDataset's *Walking* action to the right (top row), to the left (middle row) and towards the camera (bottom row).

For the training and testing of the DL models to be implemented, the *Walking* action of ActDataset was chosen as the normal situation. In each video of this action the recorded pedestrian stands still for some moments in front of the camera before and after the desired action. Since ECs are only able to record movement there are some frame samples which provide no information. Thus, a careful selection of the starting and ending moments of each video was needed, which led to fewer available samples.

After this selection, anomalous situations were required. Besides *Walking*, ActDataset

provides the following set of actions: *Arm Crossing*, *Getting Up*, *Jumping*, *Kicking*, *Picking Up*, *Sitting Down*, *Throwing*, *Turning Around*, *Waving*. Since one of the main goals is to utilise a DL model that encodes the temporal information into an MS in order to perceive the notion of movement, as described in Section 4.1, some of these actions proved to be unreliable due to the lack of sufficient movement. Another factor to take into consideration when choosing the number of anomalous actions was the number of available frame samples in order to be approximately even with the normal events. The actions that were chosen to use from this dataset and the number of event-based frames are depicted in Table 5.1, along with the direction in which each action is recorded.

Situation	**Action**	**Direction**	**No. of Frames**	**Total**
Normal	Walking	Right	514	1397
		Left	508	
		Towards the Camera	375	
Anomalous	Arm Crossing	Left	154	924
	Getting Up/Sitting	Towards the Camera	117	
	Picking Up	Right	471	
	Jumping	Right	182	

Table 5.1: Description of the events that are considered normal and abnormal in ActDataset.

5.1.2 PedDataset

PedDataset is comprised of a series of videos recorded from a high position relative to the pedestrian walking area. Figure 5.3 displays some examples of the obtained event-based frames. In these images, it is possible to see some of the activities present in the dataset, such as walking in the direction away from the camera, which will be referred to as forward, walking closer to the camera, which will be called backwards, walking parallel to the camera, a motorcyclist and a cyclist.

It can be expected that when the pedestrians are far away from the camera the implemented models will not be able to perform efficiently since there are not many changes in the information between frames.

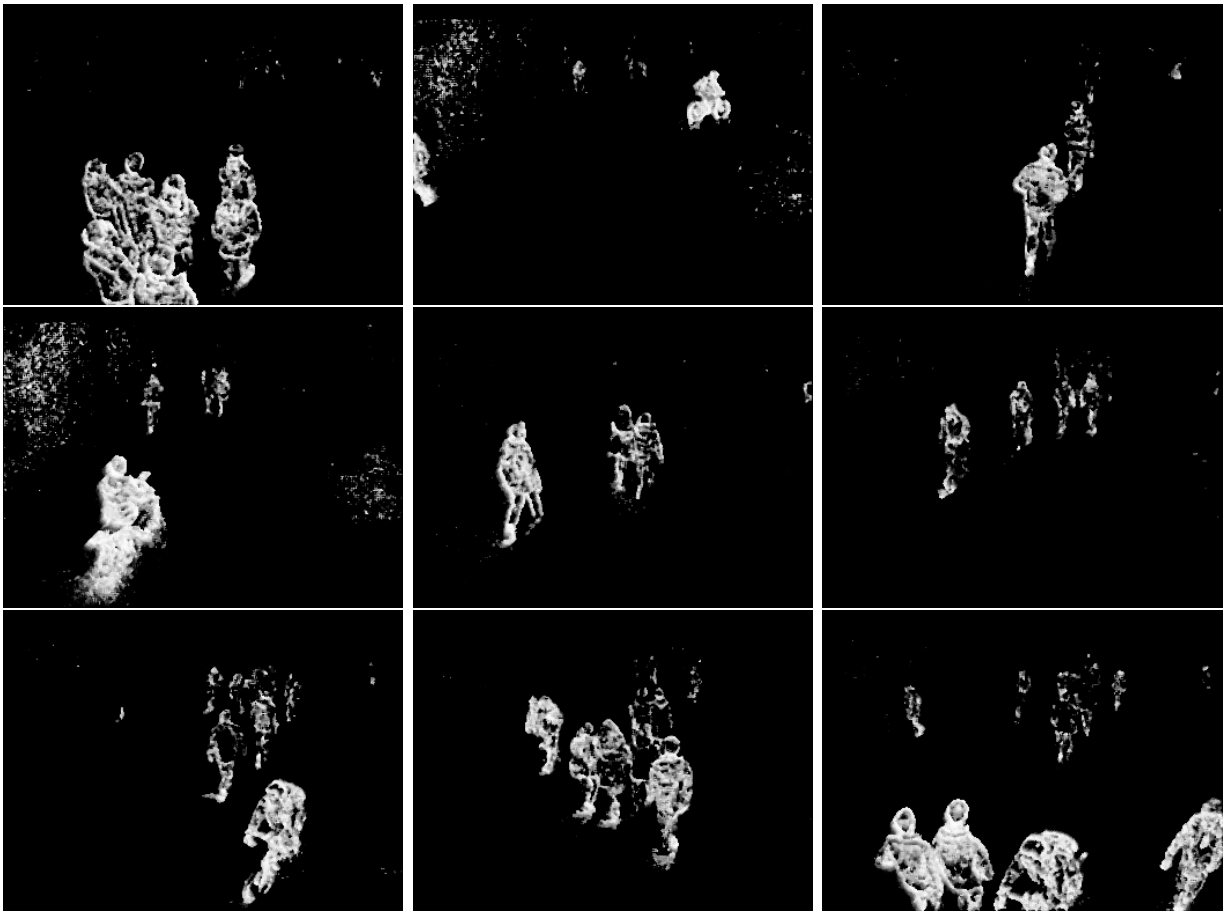


Figure 5.3: Examples of event-based frames present in the PedDatset.

Table 5.2 shows the events considered normal and anomalous in this dataset, as also the number of event-based frames obtained for both situations. Table 5.3 displays the number of frames that were selected for both the training and testing phases.

Situation	**Action**	**Direction**	**Total No. of Frames**
Normal	Walking	Forward	5500
Anomalous	Walking	Backwards/Parallel	8708
	Bicycle	Forward/Backwards	
	Motorcycle	Forward/Backwards	

Table 5.2: Description of the events that are considered normal and abnormal in PedDataset.

Phase	Situation	No. of Frames
Training	Normal	2136 (∼39%)
Test	Normal	3364 (∼61%)
	Abnormal	8708 (100%)

Table 5.3: Description of the events that are considered normal and abnormal in PedDataset.

5.2 DL Memory Surface Network

With the TSs acquired, the next phase was to implement the network for MS generation. Although its architecture is represented in Figure 4.1, some changes were required in order to fully work.

The first step was to implement the architecture described in [3], followed by applying a batch normalization layer and the ReLu activation function after each convolution. The discretized volume of event data used as input is composed of eight TSs ($Ev = [ev_0, ev_1, ..., ev_7]$), resulting in a duration of $400ms$. The authors of [3] refer that the convolution layers perform 1D convolutions to restrict them to the time dimension while maintaining the spatial distribution. Therefore, 3D convolution layers from the PyTorch library were used with a kernel size of $(8, 1, 1)$ considering the input data is organised in $(depth, height, width)$, which means it only convolutes across the depth dimension. However, the main change to the architecture is the need for a *padding* before each convolution, due to the fact that the depth convolution turns the data from 8x256x256 into 1x256x256. Since the desired kernel has a depth of eight, the padding layer was used to transform the data into 15x256x256, which will then be convoluted and result in 8x256x256. With this, the data keeps the same dimensions across the entire network, varying only in the number of channels.

Figure 5.4 demonstrates the result of the padding on a discretized volume of event data, with an example of the kernel (red squares) of size $(8, 1, 1)$ on which the convolution is performed.

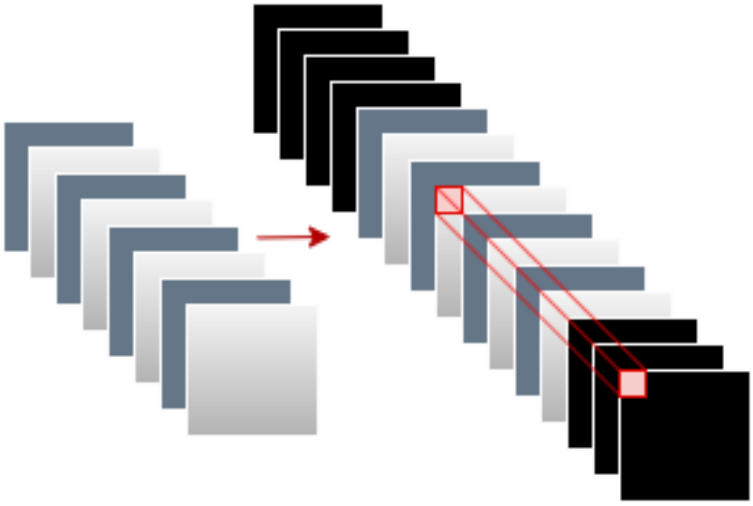
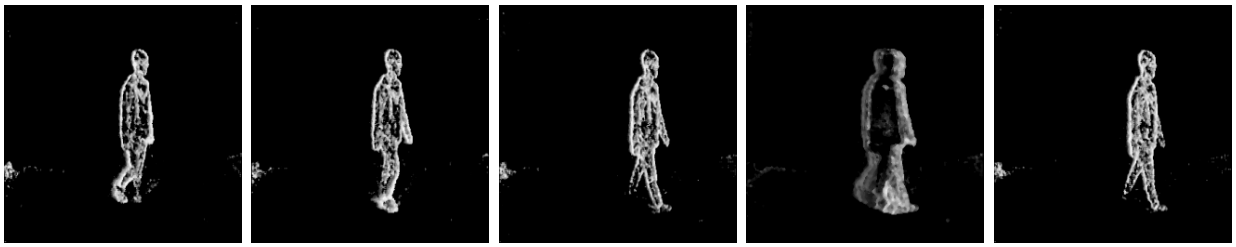


Figure 5.4: Padding applied to a discrete volume of events with a depth of eight time surfaces.

The extraction of the MS is performed at the bottleneck layer, at which the data is one channel of dimension 8x256x256. However, since the objective is to obtain a single frame with the encoded temporal information, a mean of the eight 1x256x256 frames is calculated, resulting in the desired MS.

Figure 5.5 shows the resulting MS of a pedestrian walking at an instant, t, as well as some of the TS from which the MS was created and the TS corresponding to the following instant, t+1. From a qualitative standpoint, it is possible to conclude that the DL MS network was able to condense the temporal information into a single frame, displaying higher intensity on the side corresponding to the direction in which the pedestrian is moving.

It is also an interesting aspect to note that the network was able to smooth out the noise present on the left side of the frames, proving to be an added value by being able to eliminate or smooth out the noise that remained present even after pre-processing.



(a) (b) (c) (d) (e)

Figure 5.5: Time surfaces at instants t-8 (a), t-5 (b), t-1 (c) and t+1(e) and the obtained memory surface that corresponds to instant t (d) for the *Walking* action of ActDataset.

Similarly to Figure 5.5, Figures 5.6 displays the same situations for the obtained MSs with other actions of ActDataset.

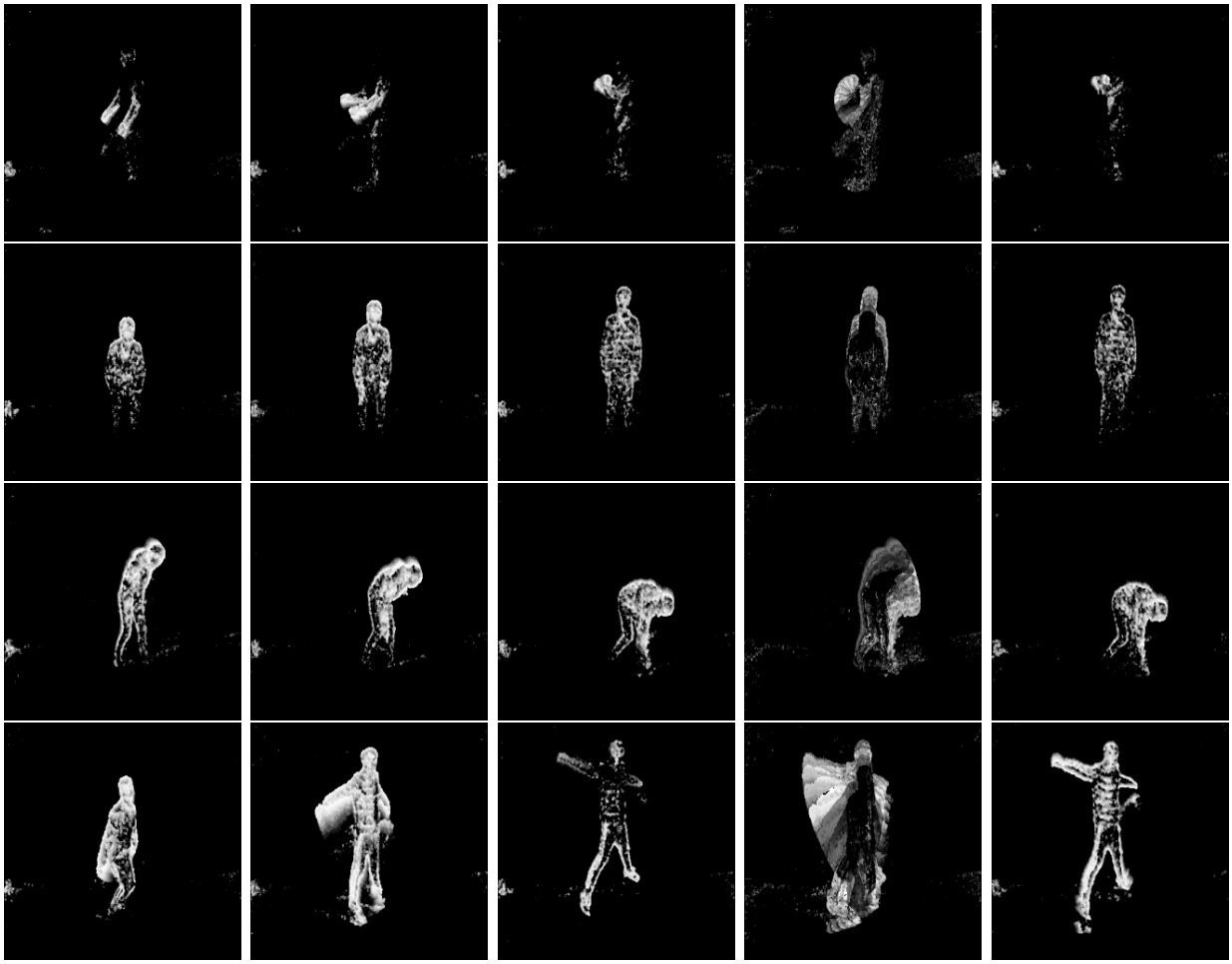


Figure 5.6: Time surfaces at instants t-8, t-5, t-1 (first to third columns, respectively) and t+1 (last column) and the obtained memory surface that corresponds to instant t (fourth column) for the *Arm Crossing*, *Getting Up*, *Picking Up* and *Jumping* actions of ActDataset.

This network was trained using the ActDataset, taking into account that for this training the situation of the events is irrelevant, i.e., it is not important whether the network receives normal or anomalous events, as the goal is to generate memory surfaces.

Different parameters, optimization functions and loss functions were attempted in the training of the network in order to find which ones produced the best results. Table 5.4 demonstrates the parameters that best fit the network. The chosen optimization function was the Stochastic Gradient Descent (SGD) with its momentum also referred to in the table. As for the loss function, in [3] the authors used a variation of the Forward KL divergence which turned similar to the Mean Squared Error (MSE) loss function, therefore the MSE was used. It is important to note that a small subset of ActDataset was used to train this network, which led to a bigger number of training epochs used to achieve the same performance as when working with a larger dataset.

Parameter	Value
No. of Time Surfaces	8
Volume of Events Duration	$400ms$
Learning Rate	$1x10^{-4}$
Epochs	600
SGD Momentum	0.5

Table 5.4: Parameters and structure of the DL memory surface generation network.

Although it was trained using ActDataset the model proved to output good results when used with the PedDataset, which proves the model can adapt and provide a good MS representation from different perspectives. Figure 5.7 shows the resulting MS in the same conditions as Figure 5.5.

(a) (b) (c) (d) (e)

Figure 5.7: Time surfaces at instants t-8 (a), t-5 (b), t-1 (c) and t+1(e) and the obtained memory surface that corresponds to instant t (d) on the PedDataset.

5.3 Conditional GAN Network

With the DL MS generation network fully functioning and providing the MSs, the next step was implementing the conditional GAN. Although an SSC cGAN with a PatchGAN discriminator is proposed in [3], a simple cGAN was implemented first, with the objective of progressively altering it to include the PatchGAN.

Thus, the cGAN structure follows the description provided in Section 4.1.2 and showed in Figure 4.1 (middle and bottom rows), presenting Convolution/Deconvolution-Pooling-Activation blocks in the generator network, in which the pooling layers consist of max pooling layers and the activation function is ReLu, with the exception of the final layer, which utilises the Tanh activation function. On the other hand, the first implementation of the discriminator varies in the sense that it outputs a single value to classify the image as real

or generated, instead of an NxN matrix. To output a single value, a fully-connected layer was added which is then convoluted. The discriminator was also altered in the sense that a kernel of size 3x3 was used instead of the one suggested in Figure 4.1. Figure 5.8 shows the structure of the described implementation of the discriminator network.

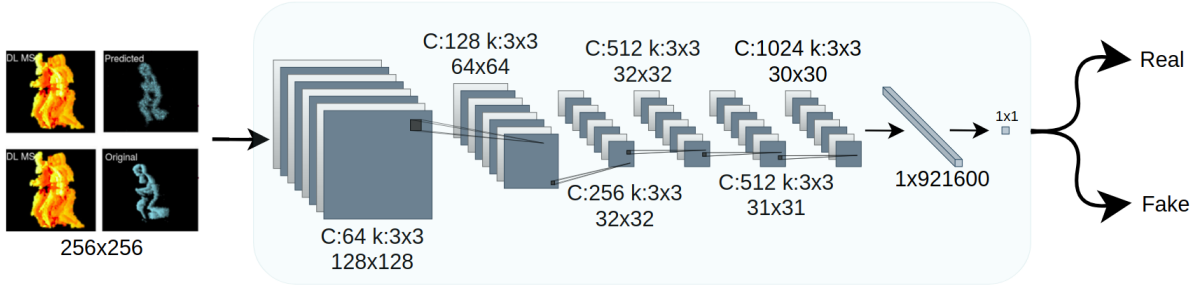


Figure 5.8: First implementation of the discriminator model.

Training a GAN is a challenging task due to a number of factors that make it difficult to find the optimal balance between the generator and discriminator. Some of these challenges are the instability of GANs during the training phase, which produces a disequilibrium between the two networks, or mode collapse, in which the generator becomes too powerful too quickly and only builds limited variations of the same example instead of generating a diverse set of data. A solution for mode collapse is training with a weaker generator or stronger discriminator, however, if they are unbalanced the model also fails to learn properly.

With the objective of preventing these issues, some regularization techniques were added, such as batch normalization layers after each convolution and a dropout layer in the generator. A Cosine Annealing learning rate was also implemented for the training phase, which takes the form of half a cosine curve, starting at a large learning rate and rapidly decreasing to a minimum value. The form this learning rate takes is depicted in Figure 5.9.

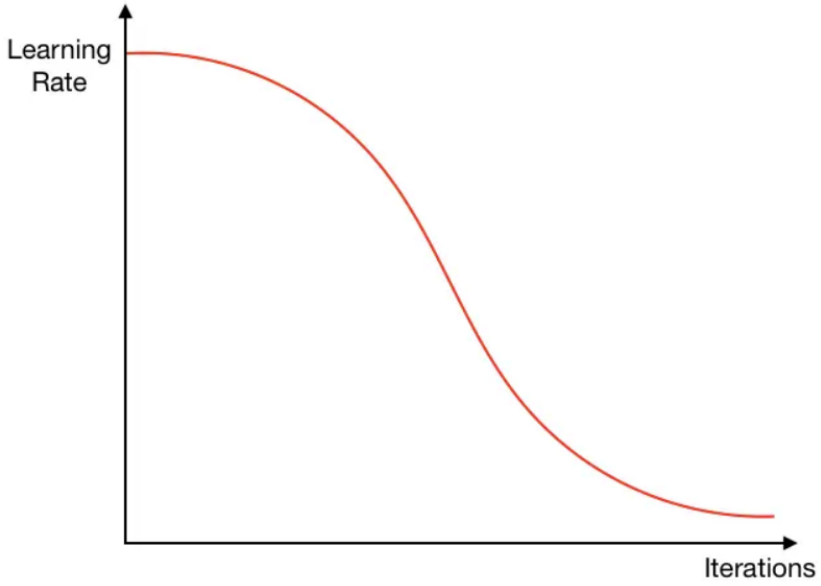


Figure 5.9: Cosine Annealing Learning Rate.

The loss curves for the generator and discriminator while training a GAN can provide some information about the progression of the training but are not reliable indicators of when it has converged to a good result since GANs are non-convex optimization problems, i.e., there can be multiple local minima and the global minimum may not produce the best output sample. Therefore, while these loss curves can be useful to indicate whether the training is converging to a minimum, the generated samples must also be evaluated qualitatively.

This first implementation was initially trained and tested on the ActDataset. This training was applied in three different situations, with these three dividing the cases considered normal as the pedestrian walking to the right, the pedestrian walking to the left and to the right, and the pedestrian walking to the right, to the left and towards the camera. These situations are referred to as trained for one direction, two directions, and three directions, respectively. Another method used to prevent the instability of the cGAN model is to increase the size of the dataset, thus data augmentation was used for two directions through a horizontal flip, which replicates the data and flips it along the height axis, resulting in twice the size of the original dataset.

A qualitative evaluation of the samples generated by this model reveals that the results obtained fall short of expectations, as demonstrated in Figures 5.10, 5.11 and 5.12. As deduced in Section 5.1.1, when trained in three directions, which includes the movement towards the camera, the model has the poorest ability to generate the expected. In the training regarding one direction, is it possible to note that the model would still require more training in order to learn a more detailed frame generation, however, due to hardware

limitations, it is not possible to infer this affirmation. In the two-direction training, it is noticeable that the model is able to generate a blurred silhouette of a person in the correct location, but fails to generate the person with better detail.

A possible explanation for the model to generate a blurred silhouette of a pedestrian may be the need for the convolutions to make use of the sparsity of data instead of performing normal convolutions, meaning the use of SSC convolutions could improve the model's performance due to how these convolutions work, explained in Section **??**. However, the implementation of these convolutions was not taken into consideration.

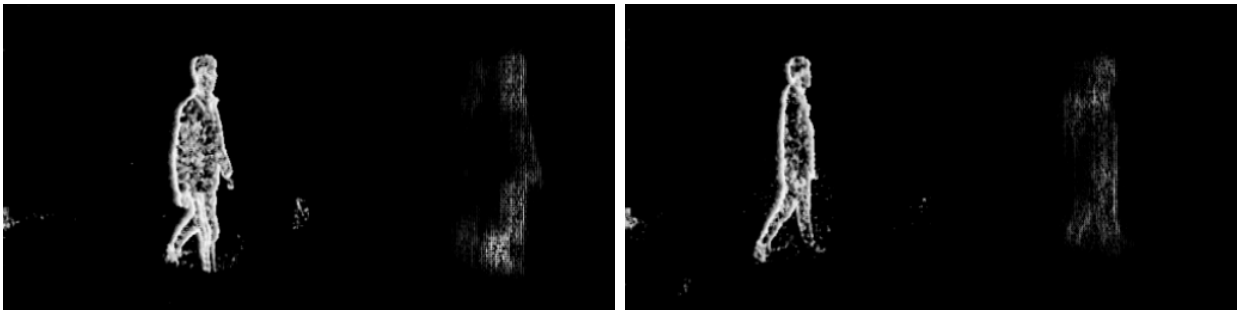


Figure 5.10: Real frames (left of each sub-figure) compared to generated frames (right of each sub-figure) when trained for one direction.

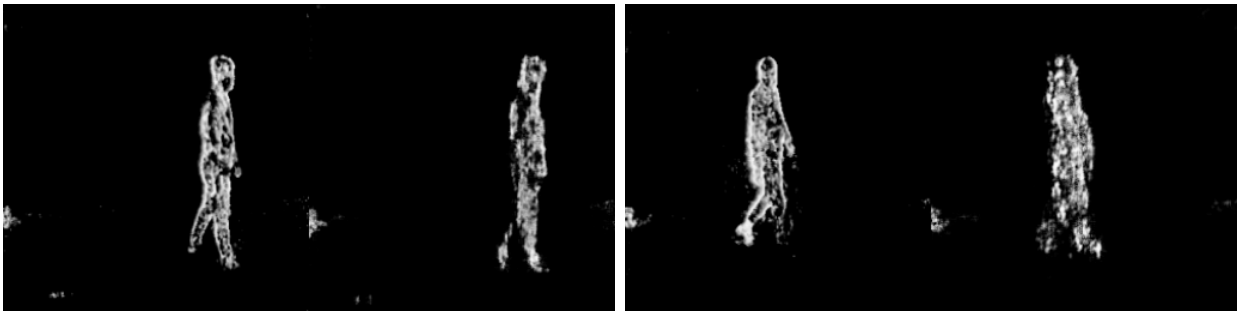


Figure 5.11: Real frames (left of each sub-figure) compared to generated frames (right of each sub-figure) when trained for two directions.

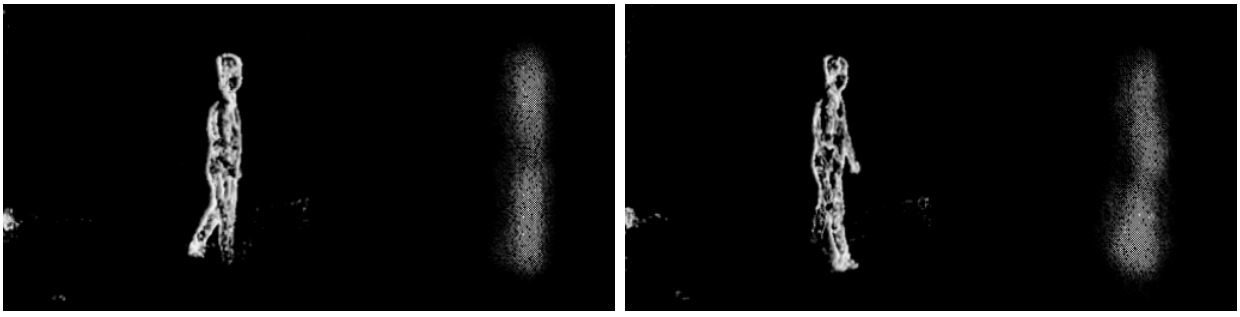


Figure 5.12: Real frames (left of each sub-figure) compared to generated frames (right of each sub-figure) when trained for three directions.

Although the model could improve with the fine-tuning of different parameters or the implementation of SSC convolutions, due to time and hardware limitations a different approach was taken. Since the proposed methodology makes use of a cGAN to produce the desired output, a Pix2Pix model was implemented, replacing the previous instance. This model, already explained in Section 3.5, is built for image-to-image translation, which falls in line with the desired result.

The authors of Pix2Pix [4] publicly provided their own implementation of this model via GitHub repository, which made it possible to adapt and train it to this dissertation's objectives.

Two main training conditions were performed using ActDataset, one regarding the standard parameters provided by the model and the other making use of a Cosine Annealing learning rate. Once again, only normal events were fed to the network during the training phase. The standard parameters for this model, fine-tuned by its creators, are demonstrated in Table 5.5. Unlike the previous implementation, this framework employs the Adam optimizer as the optimization function.

Parameter	Value
Epochs	200
Batch Size	1
Starting Learning Rate	$2x10^{-4}$
Learning Rate Policy	Constant
Generator Model	U-Net
Discriminator Model	PatchGAN

Table 5.5: Parameters and structure of the DL memory surface generation network.

Figures 5.13 and 5.14 depict the new results obtained with Pix2Pix for the right and left directions of the *Walking* of ActDataset, in which a qualitative evaluation reveals a significant improvement in the output, becoming possible to see a more distinct outline of a person in the generated frames.

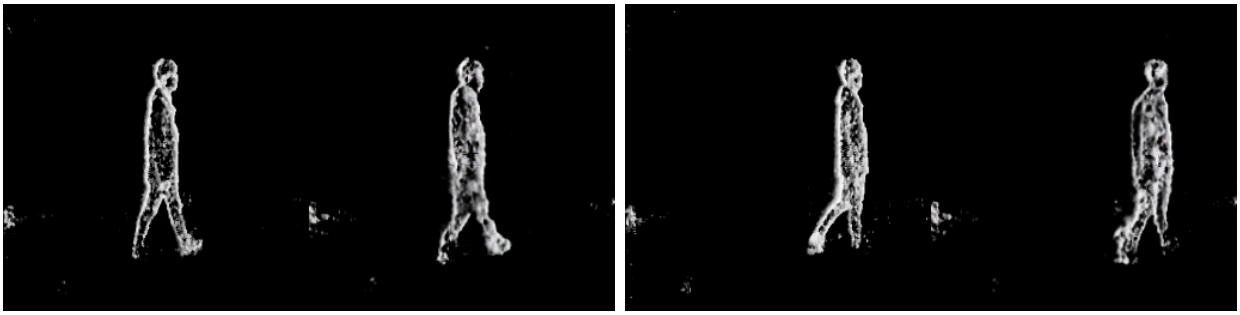


Figure 5.13: Real frames (left of each sub-figure) compared to generated frames (right of each sub-figure) when trained for the right direction of *Walking* on ActDataset.

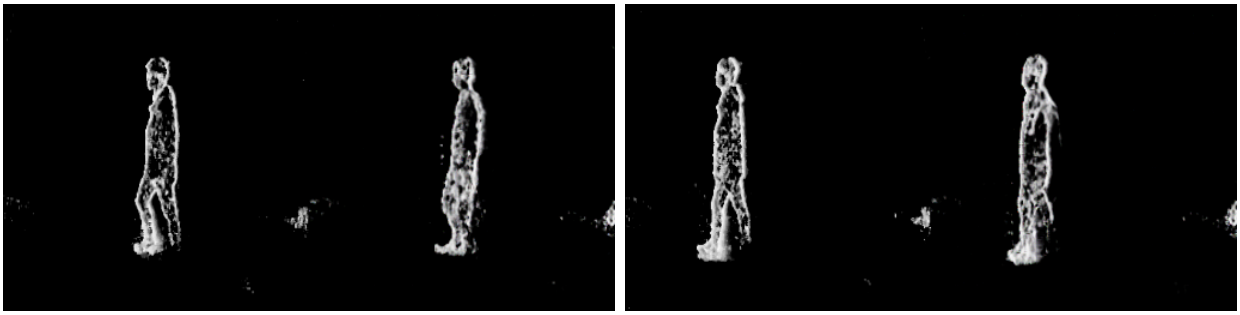


Figure 5.14: Real frames (left of each sub-figure) compared to generated frames (right of each sub-figure) when trained for the left direction of *Walking* on ActDataset.

In Figures 5.15, 5.16, 5.17 and 5.18 it is perceptible that the model it is not able to generate reliable frames when provided with anomalous actions during the testing phase. In most of the provided anomalous MS, since the model does not know how to predict the anomaly frame, it generates a frame that shares more similarities with the MS than with the real frame. For this comparison, the figures show the MS provided as input, the generated frame delivered as output and the real frame. It is especially noticeable in the *Picking Up* action.

Figure 5.15: MS (left) and real frames (middle) compared to generated frames (right) when trained for the *Arm Crossing* action of ActDataset.

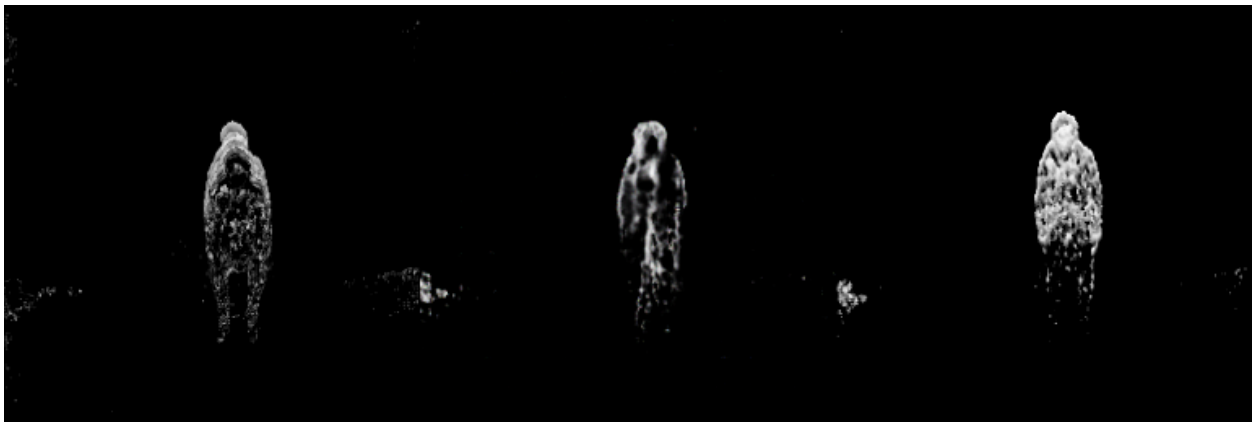


Figure 5.16: MS (left) and real frames (middle) compared to generated frames (right) when trained for the *Getting Up* action of ActDataset.

Figure 5.17: MS (left) and real frames (middle) compared to generated frames (right) when trained for the *Picking Up* action of ActDataset.

Figure 5.18: MS (left) and real frames (middle) compared to generated frames (right) when trained for the *Jumping* action of ActDataset.

Unlike the testing phase, during the training only normal cases were provided to the network so that it could learn to generate normal events. After being trained for the described conditions, the final step was to use the saved models to test the model to detect anomalies with the normal cases and abnormal events being the ones referred to in Section 5.1.1.

The metrics used to evaluate the performance of each implementation will be described in the following section and the results for all the tests with this model will be provided in Section 6.

5.4 Implementation Details and Metrics

All the implementations and tests performed were developed using the Python programming language with the assistance of the Pytorch library for DL models.

The training and inferences of the implemented models were executed on one of these GPUs: Nvidia TITAN X or Nvidia RTX 3090.

The Matlab environment was also used since it provides an easier image and data manipulation, such as applying the median filter to the TSs or process and obtaining the ROC curves displayed in Section 6.

The metrics used to evaluate the developed framework were recall, precision, f1-score and accuracy. All these metrics were calculated based on True-Positive Rates (TPR), False-Positive Rates (FPR), True-Negative Rates (TNR) and False-Negative Rates (FNR), in which positive or negative indicates the presence or absence of anomalous events.

The recall is used to measure the proportion of the positive predictions made by a model out of all true-positive instances and it goes by:

$$Recall = \frac{TPR}{TPR + FNR} \tag{5.1}$$

The precision calculates the proportion of true-positive predictions made by a model out of all positive predictions made by a model. It is estimated as:

$$Precision = \frac{TPR}{TPR + FPR} \tag{5.2}$$

The f1-score is the harmonic mean of recall and precision, providing a balance between the two. It is represented by:

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{5.3}$$

Lastly, accuracy indicates the proportion of correct predictions made by a model out of all predictions made and its formula is:

$$Accuracy = \frac{TPR + TNR}{TPR + FPR + TNR + FNR} \tag{5.4}$$

The authors also used the EER to summarise the performance of the anomaly detection network. EER is the ratio of frames that are misclassified at $FPR = 1 - TPR$.

As already stated, the training was performed with normal activities so that the model learns to also generate TSs of normal situations. However, the inference was made with normal and abnormal scenes, so that it generates a TS that corresponds to a normal situation when given an abnormal DL MS, which allows for the inference to detect a difference between the generated and the real TS.

To estimate the prediction error, the used measure was the normalised mean square error (MSE) between the TS predicted by the generator and the ground truth TS. The MSE is estimated as given in Equation 5.5, described in [3], and is expected to represent a lower reconstruct error with normal situation sequences rather than with anomalous events.

$$MSE = \frac{1}{N_x N_y} \sum_{x=1}^{N_x} \sum_{y=1}^{N_y} \widehat{en_{xy}} - en_{xy} \tag{5.5}$$

In Equation 5.5, $\widehat{en_{xy}}$ and en_{xy} represent the predicted and original normalised timestamps at the location (x, y) in the TS and N_x and N_y are the number of rows and columns, respectively.

A Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classifier model, as is the case of the implemented method. The

ROC curve is a plot of the TPR against the FPR at various threshold settings. These are obtained as previously explained. To obtain the ROC curve, multiple threshold settings are evaluated and plotted to generate a smooth curve.

The AUC of the ROC is a scalar value that describes the performance of the classifier network, providing a single metric for comparison across different models or datasets. AUC is the area under the ROC curve and is equal to the probability that a randomly chosen positive sample will have a higher predicted probability of belonging to the positive class than a randomly chosen negative sample. AUC values range from 0.5 to 1, with a perfect classifier having an AUC of 1 and a random classifier having an AUC of 0.5.

6 Experimental Results

This chapter aims to present the results that came from the developed training and inferences for each implementation. As previously stated, a model's performance can be expressed by the AUC of a ROC curve. The ROC curves and their AUC values for each inference will be analysed. Although the loss curves from both the generator and discriminator of a GAN are not sufficient to evaluate the learning course of a model, as already discussed, these are helpful to infer whether the model has converged to a minimum. Therefore, these curves will also be presented.

As stated in Section 5.3, two training conditions for the Pix2Pix model were used on ActDataset, being the main change between them the learning rate policy, in which the standard one makes use of a linear/constant learning rate whilst the other employs a Cosine Annealing learning rate.

Figure 6.1 displays the loss curves for the discriminator and generator during the training phase with the standard conditions of the model. It is noticeable that the loss of the generator, although it does not decrease as it would be expected in a normal network, it oscillates around the same value (~ 200), which means the model converged to a minimum. In order to infer whether the model would stagnate at this minimum or it would try to converge to another, a second training was fulfilled in which the number of training epochs was increased from 200 to 400. Figure 6.1 displays the loss curves for this training. It is possible to see that around epoch 270 the loss of the generator was indicating that the GAN could be entering a state of GAN mode collapse, i.e., producing limited or repetitive outputs, resulting in a sharp increase in the loss curve. However, the model presents robustness in such a way that it was able to stabilise and converge again.

It is important to notice that the model was able to rapidly converge and learn how to generate normal events. Therefore, three inferences were made to these standard parameters, one at the start of the training, in epoch 10, one in the middle, in epoch 120, and the last one at the end, in epoch 200. An inference was also made at the end of the extended version

of these conditions, in epoch 400.

The ROC curves for these inferences can be seen in Figures 6.2. These curves allowed for the AUC values to be obtained which are demonstrated in Table 6.2.

The second training performed on this model made use of Cosine Annealing and proved to make almost no difference. In Figure 6.1 it is concluded that in a similar manner to the standard version, the generator rapidly converged into approximately the same value, although it is possible to note that near the end of the training the value at which the loss oscillates around gained a slightly decreasing tendency. The performance represented by the AUC value displays an increase, demonstrated in Table 6.2, meaning it was a valuable addition to the model. The ROC curve for this inference is displayed in Figure 6.2.

In addition to the AUC values, obtained for a range of thresholds, Table 6.2 also shows the metrics referred to in Section 5.4 for the optimal threshold. The optimal is chosen based on the F1-Score metric, which represents a balance between recall and precision.

Training this model on PedDataset demonstrated unsatisfactory performance, as shown by its ROC curve in Figure 6.2 and by its AUC value in Table 6.1. In Figure 6.1 the loss curve for this training is depicted and it is possible to conclude that the model was unable to converge into a minimum, gradually increasing the generator's loss. This failure in the performance of this dataset may possibly be due to the inability to encode the information into MSs when pedestrians are away from the camera. Two achievable ways that may help surpass this problem are a deeper selection of the frames available for training and testing, and an alteration of the events considered normal and anomalous, becoming normal events all frame with pedestrians walking, independently of the direction of the movement, and abnormal every frame with non-pedestrians in it, such as motorcycles and bicycles.

Model	Epoch	AUC	Threshold	Recall	Precision	F1-Score	Accuracy
Standard	200	0.5633	0.4074	0.4692	0.9993	0.6386	0.6533

Table 6.1: Metric results for the Pix2Pix model trained on PedDataset.

Model	Epoch	AUC	Threshold	Recall	Precision	F1-Score	Accuracy
Standard	10	0.8471	0.9670	0.3336	0.9990	0.5002	0.6001
Standard	120	0.8098	0.9610	0.4005	0.9993	0.5719	0.6253
Standard	200	0.8247	0.9399	0.3338	0.9981	0.5003	0.6002
Standard	400	0.8023	0.8278	0.3766	0.9968	0.5467	0.6162
Cosine	200	0.8583	0.9740	0.3339	0.9990	0.5002	0.6001

Table 6.2: Metric results for the Pix2Pix model trained on ActDataset.

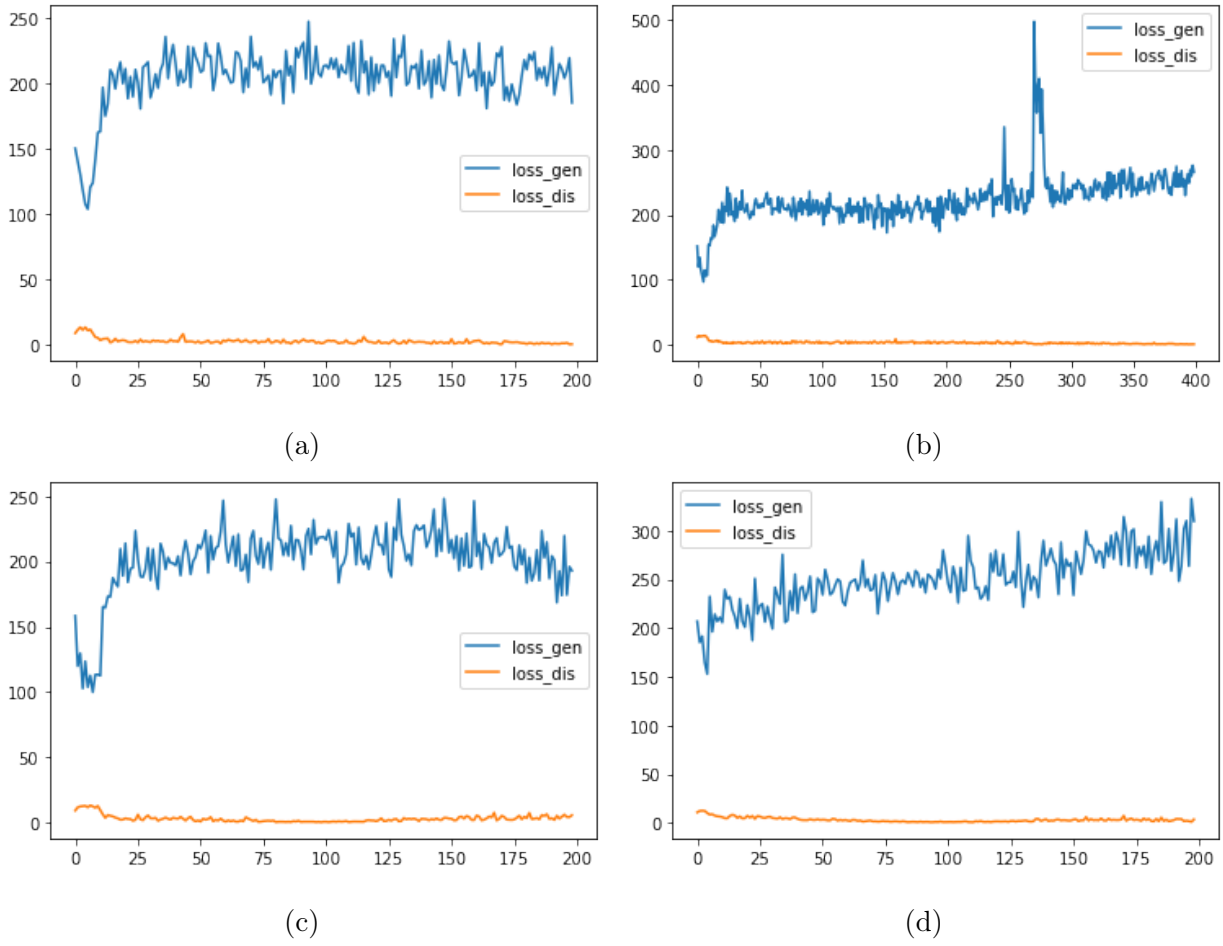
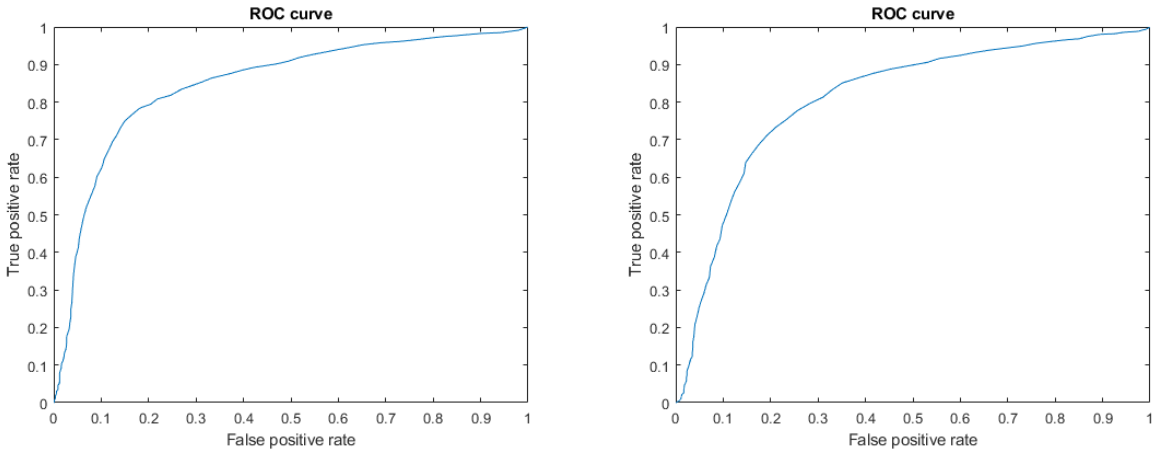
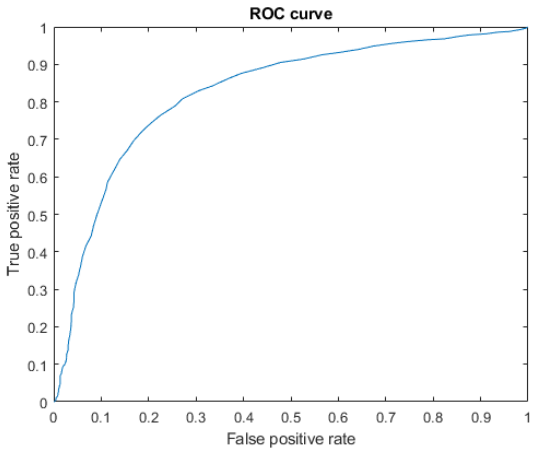


Figure 6.1: Loss curves of the generator (blue) and discriminator (orange) networks when trained with standard parameters (a), extended epochs from 200 to 400 (b) and a Cosine Annealing adaptive learning rate (c) on ActDataset. Loss curves of the generator and discriminator networks when trained with standard parameters on PedDataset.

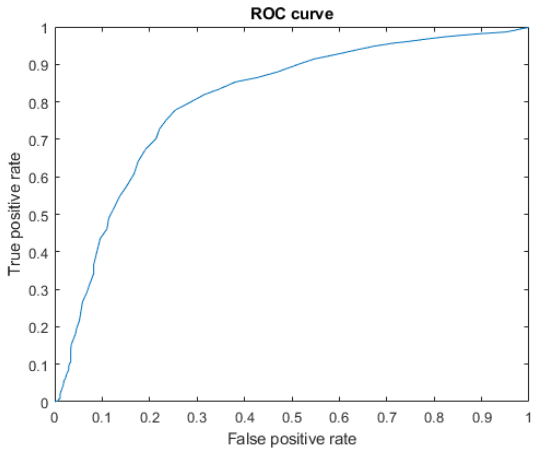


(a)

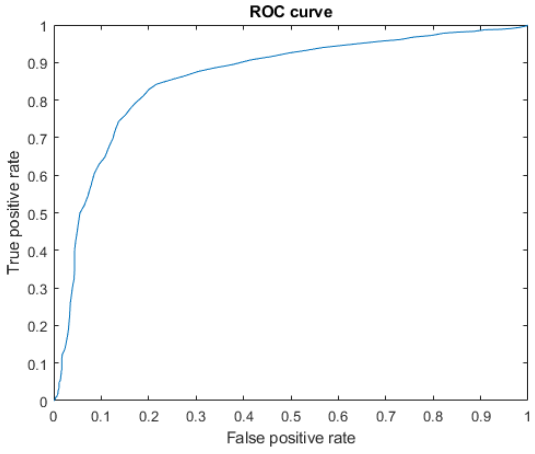
(b)



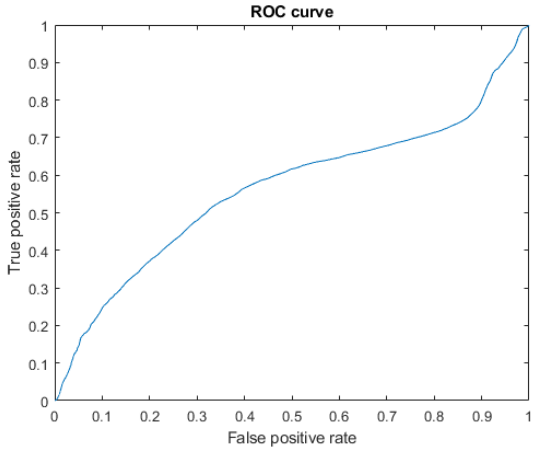
(c)



(d)



(e)



(f)

Figure 6.2: ROC curve for Pix2Pix model trained with standard parameters, being inferred at epoch 10 (a), epoch 120 (b), epoch 200 (c) and epoch 400 (d), and with a Cosine Annealing adaptive learning rate (e) on ActDataset. ROC curve for Pix2Pix model trained with standard parameters, being inferred at epoch 200 (f), on PedDataset.

7 Conclusion and Further Work

This chapter seeks to provide an overview of the suggested goals and the completed work, as well as the challenges encountered in carrying out this work and any future work proposals.

Although one can conclude that validating incorporation of a DL MS network with a Pix2Pix model to perform AED demonstrated effective results when used on one of the available datasets, it falls short of expectation when compared to the base model on which this dissertation was based.

An important aspect to notice is that the validation of the DL MS generation network has proven to be successful, being able to properly encode the temporal characteristic of the sparse event-based data into a single TS frame. This temporal information is valuable because it allows us to take into account the temporal dependencies between different moments in time.

The generative networks implemented were two different cGANs, although only the one with the best qualitative results was inferred, corresponding to the Pix2Pix model. This model productively generated normal situations with the detailed shape of the pedestrian in the TSs when fed with normal MSs and proceeded to generate poorly when fed with anomalous MSs.

Some limitations were felt regarding the available datasets. Since the used datasets were not designed for anomaly detection, even though a careful separation of their data into normal and abnormal events has taken place, it could have contributed to worse results. A deeper and more cautious analysis of the available data may be helpful to improve the performance of the model.

Due to the lack of time and hardware resources, not many instances of training and inference were possible. With the proper resources, a greater fine-tuning of the model could have been performed.

Therefore, one of the proposals for future work proceeding with this dissertation is the

validation of the developed model with different parameters to improve the obtained performance results.

Another important suggestion for future work would be to make real use of the PatchGAN discriminator, allowing it to detect the zone of anomalous events on each abnormal frame, instead of using it as a simple classifier. This detection is possible because of its architectural design, in which the output is divided into separate tiles.

8 Bibliography

[1] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. 4 2019.

[2] Alessio Rivetti Silva, Jorge Manuel Moreira, and Campos Pereira Batista. Neuromorphic event-based activity and anomaly detection.

[3] Lakshmi Annamalai, Anirban Chakraborty, and Chetan Singh Thakur. Evan: Neuromorphic event-based sparse anomaly detection. *Frontiers in Neuroscience*, 15, 7 2021.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

[5] B. Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, 4(2):1–15, 2018.

[6] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E. Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. The 5th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021.

[7] Yuxiang Zhao, Wenhao Wu, Yue He, Yingying Li, Xiao Tan, and Shifeng Chen. Good practices and a strong baseline for traffic anomaly detection.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[9] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection, 2017.

[10] Zhi Zhang, Sheng hua Zhong, and Yan Liu. Video abnormal event detection via context cueing generative adversarial network. pages 1–6. Institute of Electrical and Electronics Engineers (IEEE), 6 2021.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[12] A vision-based system for traffic anomaly detection using deep learning and decision trees.

[13] Glenn Jocher. YOLOv5 Github: `https://github.com/ultralytics/yolov5`, 2020.

[14] Guang Chen, Peigen Liu, Zhengfa Liu, Huajin Tang, Lin Hong, Jinhu Dong, Jorg Conradt, and Alois Knoll. Neuroaed: Towards efficient abnormal event detection in visual surveillance with neuromorphic vision sensor. *IEEE Transactions on Information Forensics and Security*, 16:923–936, 2021.

[15] Event-based vision sensor (evs) technology | image sensor for industrial use | technology | sony semiconductor solutions group | `https://www.sony-semicon.com/en/technology/industry/evs.html`.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63:139–144, 6 2014.

[17] Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks, 2018.

[18] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[20] Lakshmi Annamalai, Anirban Chakraborty, and Chetan Singh Thakur. Supplementary of the paper evan: Neuromorphic event-based sparse anomaly detection.

[21] Supplementary material for evan !

[22] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks, 2017.

[23] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. 4 2019.

[24] Shu Miao, Guang Chen, Xiangyu Ning, Yang Zi, Kejia Ren, Zhenshan Bing, and Alois C Knoll. Neuromorphic benchmark datasets for pedestrian detection, action recognition, and fall detection. *Frontiers in neurorobotics*, 13:38, 2019.