# Unsupervised deep learning for spatio-temporal representations of videos : application to video surveillance

Devashish Lohani

N° d'ordre NNT : 2023LYO20021

# THÈSE de DOCTORAT DE L'UNIVERSITÉ LUMIÈRE LYON 2

**École Doctorale : ED 512**

**Informatique et Mathématiques**

Discipline : Informatique

Soutenue publiquement le 3 avril 2023, par :

Devashish LOHANI

---

# Unsupervised deep learning for spatio-temporal representations of videos:

*Application to video surveillance.*

---

Devant le jury composé de :

François BRÉMOND, Directeur de recherche, INRIA, Président

Thierry CHATEAU, Professeur des Universités, Université Clermont-Ferrand 2, Rapporteur

Nicolas THOME, Professeur des Universités, Conservatoire National Arts et Métiers, Rapporteur

Jenny BENOIS-PINEAU, Professeure des Universités, Université Bordeaux 1, Examinatrice

Lionel ROBINAULT, Docteur, FOXSTREAM, Examinateur

Carlos Fernando CRISPIM JUNIOR, Maître de conférences, Université Lumière Lyon 2, Examinateur

Laure TOUGNE RODET, Professeure des Universités, Université Lumière Lyon 2, Directrice de thèse

# Contrat de diffusion

Université Lumière Lyon 2
**Ecole Doctorale : ED 512 Informatique et Mathématiques**
*LIRIS*

THÈSE
pour obtenir le grade de
DOCTEUR EN INFORMATIQUE

# Unsupervised deep learning for spatio-temporal representations of videos: application to video surveillance

Devashish LOHANI

Informatique

Sous la direction de Laure TOUGNE RODET
Co-encadrée par Carlos CRISPIM-JUNIOR

Soutenue publiquement le 3 Avril 2023

Composition du jury :
Nicolas THOME, Professeur des Universités, CNAM Paris, *Rapporteur*
Thierry CHATEAU, Professeur des Universités, Université Clermont Ferrand II, *Rapporteur*
François BRéMOND, Directeur de recherche, Inria Sophia Antipolis, *Examinateur*
Jenny BENOIS-PINEAU, Professeur des Universités, Université de Bordeaux, *Examinatrice*
Laure TOUGNE RODET, Professeur des Universités, Université Lyon 2, *Directrice de thèse*
Carlos CRISPIM-JUNIOR, Maître de conférences, Université Lyon 2, *Co-encadrant de thèse*
Lionel ROBINAULT, Docteur (HDR), Foxstream, *Tuteur entreprise*

# Acknowledgments

First of all, I would like to thank my thesis supervisors: Laure TOUGNE RODET and Carlos CRISPIM-JUNIOR. They not only guided me with their scientific expertise, but also supported me morally during difficult times. I am specially grateful to them for their valuable feedbacks on our papers and this manuscript. I would also like to thank my tutor of the enterprise Lionel ROBINAULT and members of the R&I team: Quentin BARTHÉLEMY and Sarah BERTRAND. They followed my work from the start of the thesis and we had very insightful scientific discussions, which helped me immensely for my research work. Without the above mentioned people, I cannot imagine where my thesis work would be. I am sincerely very grateful to all five of you.

I would also like to thank all the jury members for accepting to consider the scientific contributions presented in this thesis. My gratitude also goes for all the members of LIRIS and FOXSTREAM, with whom I had great discussions during coffee breaks. A big thanks to all my family and friends, who always supported me and helped me stayed sane and motivated.

Finally, I would like to thank ANRT, LIRIS and FOXSTREAM, for financing the work of this CIFRE thesis.

# Abstract

In the last two decades, we have witnessed a massive increase of surveillance cameras in our surroundings. One of the most important uses of these cameras is to detect suspectful or abnormal behaviors, *e.g.,* a moving truck in a pedestrian zone or an intruder entering a prohibited site. These abnormal events occur very rarely and thus it is an extremely tedious and difficult task for professionals to attentively monitor the video constantly for finding these events. Therefore, an automatic video analysis system is essential. Traditional systems suffer to generalize across different types of anomalies, often rely on handcrafted rules and cannot adapt to abnormal events that they have never seen before. In the past few years, we have seen a tremendous progress in deep learning based video surveillance systems. These systems learn representative features from the data itself, generalize across different scenes and anomalies. That is why, in this thesis, we explore deep learning based methods. Majority of these methods in automatic video analysis are supervised, *i.e.,* they require a large volume of labeled data. But since abnormal events depend on context and are rare, it is very difficult to have labeled anomalous data beforehand, and even if there is some annotated data for abnormal events, it will always be a small portion compared to normal data. Furthermore, one cannot annotate every possible event that might occur in future. So, we require approaches that can work without labeled data. Since these events occur in videos, they can have both spatial and temporal dimensions. Therefore, the approach must be able to learn pertinent spatio-temporal representations to differentiate abnormal and normal events.

Thus, in this PhD, prepared at LIRIS laboratory and in collaboration with the Foxstream enterprise, we aim to learn spatio-temporal representations from unlabeled videos to detect abnormal events. Precisely, we address the task of video anomaly detection and its sub-task, perimeter intrusion detection. We provided mathematical definitions to these tasks because they were not clearly defined in the literature. The definitions have a direct impact on the evaluation and therefore, we proposed new suitable evaluation schemes. Concerning spatio-temporal representation learning without annotations, we proposed two approaches. In the first approach, we designed a strided 3D convolutional autoencoder network and it was used for the perimeter intrusion detection task. The main idea is to learn normal representation from training data without intrusions (or anomalies) and detect intrusions (or anomalies) as they deviate from learned normality. It worked well in small-length videos but suffered in long videos, which have changes in scene dynamics like weather, lighting, *etc.* To address this problem, we introduced an adaptive thresholding approach using moving z-score. Our extensive experiments showed the viability of our approach in comparison with other existing methods. To further improve the spatio-temporal comprehension of normality, we introduced our second approach. It consisted of a framework that leverages unsupervised and self-supervised learning in an autoencoder. To be precise, we proposed multiple, carefully designed tasks (unsupervised and self-supervised) that are performed in a single autoencoder. This method is also

trained in an end-to-end and joint manner, where training data is without anomalies or intrusions. For detecting anomalies (or intrusions), each of the task provide an anomaly score and the combined score is used for final detection. This approach is generic and was applied to the two tasks. We obtained state-of-the-art results in all major public datasets for both video anomaly detection and perimeter intrusion detection task.

**Keywords:** deep learning, computer vision, unsupervised learning, self-supervised learning, video surveillance, video anomaly detection, perimeter intrusion detection

# Résumé

Au cours des deux dernières décennies, nous avons assisté à une augmentation massive des caméras de surveillance dans notre environnement. L'une des utilisations les plus importantes de ces caméras est de détecter des comportements suspects ou anormaux, par exemple un camion en mouvement dans une zone piétonne ou un intrus pénétrant dans un site interdit. Ces événements anormaux se produisent très rarement et c'est donc une tâche extrêmement fastidieuse pour les professionnels de surveiller manuellement la vidéo en permanence pour trouver ces événements. Par conséquent, une analyse vidéo automatique est essentielle. Les systèmes traditionnels ont du mal à se généraliser à différents types d'anomalies, s'appuient souvent sur règles et ne peuvent pas s'adapter à des événements anormaux qu'ils n'ont jamais vus auparavant. Au cours des dernières années, nous avons constaté d'énormes progrès dans les systèmes de vidéosurveillance basés sur l'apprentissage profond. Ces systèmes apprennent des caractéristiques représentatives à partir des données elles-mêmes et généraliser sur différentes scènes et anomalies. C'est pourquoi, dans cette thèse, nous explorons méthodes basées sur l'apprentissage profond. La majorité des travaux de la littérature en analyse vidéo automatique sont supervisés, c'est-à-dire qu'ils nécessitent un grand volume de données étiquetées pour obtenir des résultats pertinents. Mais comme les événements anormaux dépendent du contexte et sont rares, il est très difficile d'avoir des données anomaliques étiquetées à l'avance, et même s'il existe des données annotées pour les événements anormaux, ce sera toujours une petite partie par rapport aux données normales. De plus, on ne peut pas annoter tous les événements possibles qui pourraient se produire. Nous avons donc besoin d'approches qui peuvent fonctionner directement sur les vidéos, sans nécessiter d'annotations. Puisque ces événements se produisent dans des vidéos, ils ont à la fois des dimensions spatiales et temporelles. Par conséquent, l'approche doit pouvoir apprendre des représentations spatio-temporelles pertinentes pour différencier les événements anormaux et normaux.

Ainsi, dans cette thèse, préparée au sein du laboratoire LIRIS et en collaboration avec l'entrepr-ise Foxstream, nous visons à apprendre des représentations spatio-temporelles à partir de vidéos non étiquetées pour détecter des événements anormaux. Plus précisément, nous abordons la tâche de détection d'anomalie vidéo et sa sous-tâche, la détection d'intrusion périmétrique. Comme ces tâches n'étaient pas clairement définies, nous en avons proposé des définitions mathématiques. Ces définitions ont un impact direct sur l'évaluation et nous avons donc proposé de nouveaux schémas d'évaluation adaptés. Concernant l'apprentissage des représentations spatio-temporelles sans annotations, nous avons mis en place deux approches. Dans la première approche, nous avons conçu un réseau d'auto-encodeur convolutif 3D. L'idée principale est d'apprendre la représentation normale à partir de données d'entraînement sans intrusions (ou anomalies) et détecter les intrusions (ou anomalies) lorsqu'elles s'écartent de la normalité apprise. Il a été utilisé pour la tâche de détection d'intrusion périmétrique. Cela a bien fonctionné dans les vidéos de petite durée, mais moins bien dans les longues vidéos, qui ont des changements dans

la dynamique de la scène comme la météo, l'éclairage, etc. Pour résoudre ce problème, nous avons introduit une approche de seuillage adaptatif utilisant le z-score mobile. Nos nombreuses expérimentations ont montré la viabilité de notre approche par rapport aux autres méthodes existantes. Pour encore améliorer la compréhension spatio-temporelle de la normalité, nous avons introduit notre deuxième approche. Il consistait en un cadre qui exploite l'apprentissage non supervisé et auto-supervisé dans un auto-encodeur. Pour être précis, nous avons proposé de multiples tâches bien conçues (non supervisées et auto-supervisé) qui sont exécutés dans un seul auto-encodeur. Cette méthode est également entraînée de bout en bout et de manière conjointe, où les données d'entraînement sont sans anomalies ou intrusions. Pour la détection d'anomalies (ou d'intrusions), chacune des tâches fournit une anomalie score et le score combiné est utilisé pour la détection finale. Cette approche est générique et a été appliquée aux deux tâches. Nous avons obtenu des résultats au-delà de l'état de l'art pour les deux tâches, dans les principaux jeux de données publics.

**Mot clés :** apprentissage profond, vision par ordinateur, apprentissage non supervisé, apprentissage auto-supervisé, vidéosurveillance, détection d'anomalie vidéo, détection d'intrusion périmétrique

# Contents

# Introduction

## General context

Over the years, the video surveillance systems have evolved from simple video acquisition and display systems to intelligent autonomous systems (Ibrahim, 2016). The systems of today use some of the most sophisticated video analysis and decision-making algorithms to function. The massive installations of cameras in different sites, from banks to supermarkets and in prominent streets, have further helped in developing and testing these systems.

Visually surveying a site can include various tasks, such as object detection, object tracking, object segmentation, abnormal event detection, *etc* (Valera et Velastin, 2005). Out of these tasks, the abnormal event (anomaly) detection task has gained a major attention of the computer vision community (Ramachandra *et al.*, 2022). Anomalies are patterns in data that do not follow a well-defined notion of normal behaviour (Chandola *et al.*, 2009). Based on context and nature of input data, anomalies can refer to different patterns, such as abnormal sections in a time-series data, abnormal patches in an image, abnormal spatio-temporal volumes in a video, *etc.*, as pictured in Figure 1.

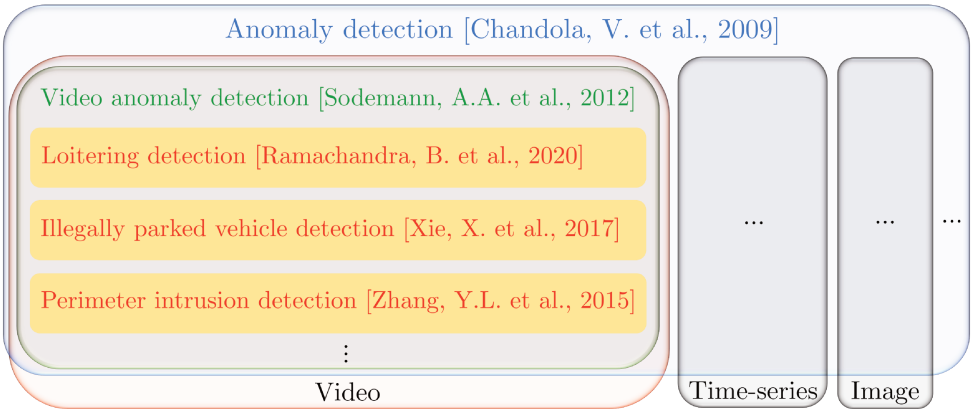



Figure 1: Venn diagram to illustrate the taxonomy of tasks in anomaly detection.

For data in video form, video anomaly detection (VAD) refers to the detection of unusual appearance or motion attributes in the video (Feng *et al.*, 2021; Li *et al.*, 2022). For example, for a video surveilled site where only pedestrians are authorized, all the vehicles are anomalies. Depending on the context, video anomaly detection can be further classified into specific tasks, such as detecting abandoned object (Luna *et al.*, 2018), loitering (Ramachandra et Jones, 2020), illegally parked vehicle (Xie *et al.*, 2017b), *etc.* The task of perimeter intrusion detection (PID) also falls into this category (Saligrama *et al.*, 2010; Sodemann *et al.*, 2012; Nayak *et al.*, 2021). It aims at detecting the presence of an intrusion in a secured perimeter. Intrusions are a particular type of anomalies, classified

as point and contextual anomalies by (Chandola *et al.* (2009), Section 3.5). Moreover, the notions of perimeter, intruder movement and site protection time are crucial for the PID task, *i.e.*, anomalous/unauthorized objects present in the video are intruders only if they are in movement inside the designated perimeter when the site is being surveyed.

In this thesis, we tackle the tasks of video anomaly detection and perimeter intrusion detection. Furthermore, both the tasks are carried out in the outdoor environment. The fact that it is an outdoor environment is very important here as it induces challenges such as changing weather and light conditions, obtrusion due to insects, animals, vegetation, *etc.*, contrary to an indoor environment (Matern *et al.*, 2013; Villamizar *et al.*, 2018). Since the input to both tasks are videos, *i.e.,* data with spatial and temporal dimensions. Anomalies or intrusion in videos too, can have the spatio-temporal nature. Therefore, it is essential to understand the spatio-temporal representation in videos correctly, in order to detect anomalies or intrusions. Some works address spatial (frames) and temporal dimensions of video separately and then process them sequentially. But these approaches do not learn the spatio-temporal nature of videos aptly (Hasan *et al.*, 2016; Chong et Tay, 2017; Simonyan et Zisserman, 2014). Thus, we seek to develop approaches that learn the spatio-temporal nature of videos in a joint and cohesive manner.

Traditionally, most of the works used trajectory-based anomaly detection (Popoola et Wang, 2012). The main idea is if the objects of interest are not following the learned normal trajectories, then they are considered anomalous. This approach suffers from occlusions as it needs to monitor continuously the objects of interest. Some other approaches used low-level appearance, motion and texture features to enhance the comprehension of normality. Other approaches can be found in the works of (Li et Cai, 2016; Kaur *et al.*, 2018). All these traditional approaches suffer to generalize across datasets and anomaly types, often rely on handcrafted features, and are unable to adapt to anomalies that they have never seen before (Hu *et al.*, 2016; Medel et Savakis, 2016). For a sterile zone with very few distractions, these approaches can function to some extent for the PID task but in the case of VAD, these approaches perform poorly. Over the past few years, deep learning has surpassed traditional approaches in many major computer vision tasks. Concerning video anomaly detection too, deep learning based approaches have outperformed their traditional counterpart. This is because, given enough data, deep learning based approaches can learn the features themselves, instead of using handcrafted or chosen features. These approaches generalize with different scenes and types of anomalies. Therefore, in this thesis, we consider developing deep learning based approaches. Most of the existing works for spatio-temporal representation learning in video rely on huge volume of labeled data to obtain relevant results. But since in VAD and PID task, the examples of anomalies or intrusions are rare, it is extremely difficult to obtain annotated data. Thus, the goal of this thesis is to learn pertinent spatio-temporal representation from unlabeled videos in order to detect abnormal behaviors like anomaly or intrusion. Furthermore, the proposed methods should be suitable to run in videos of long length (hours, even days), in order to test if they are robust to real-world challenges like change in weather, luminosity, *etc.*

# Industrial context

This thesis is carried out within the framework of a CIFRE (Convention Industrielle de Formation par la REcherche) partnership between the laboratory LIRIS (Imagine team)

and the company Foxstream (CIFRE n°2019/1709). The Imagine team of laboratory LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information) is specialized in computer vision, machine learning and pattern recognition. On the other hand, the company Foxstream is specialized in the analysis and automatic processing of real time video content. The video analysis algorithms developed by the company (perimeter intrusion detection, object segmentation, object tracking, *etc.*) use "classic" signal processing or image analysis algorithms such as modeling of the static components of the scene or using low level descriptors such as SIFT, Haar wavelets, HOG, *etc.* With this thesis, they want to explore deep learning based solutions for perimeter intrusion detection and video anomaly detection tasks. Furthermore, since intrusions and anomalies are rare events, the company cannot provide labeled data as one cannot anticipate all possible anomalies beforehand. Therefore the proposed methods should be able to work directly in unlabeled data. Overall, the work from this thesis would help the company to move towards deep learning approaches in many of its major applications and that too, without additional cost of manual annotation.

# Plan of manuscript

This manuscript is composed of 5 chapters:

- Chapter 1 details various data acquisition schemes and associated challenges in video surveillance. It also describes various datasets used in this work along with their challenges and shortcomings.
- Chapter 2 corresponds to a literature review of various topics: spatio-temporal representation learning in videos with and without annotations, video anomaly detection and perimeter intrusion detection. The basic concepts needed to understand this manuscript are also defined there.
- Chapter 3 is concerned with formally defining the tasks of video anomaly definition and perimeter intrusion definition, and revisiting their evaluation frameworks. We mathematically define the tasks and discuss existing evaluation protocols. We also propose a new, more suitable evaluation protocol.
- Chapter 4 focuses on the task of perimeter intrusion detection. For this, we propose an unsupervised autoencoder with adaptive thresholding mechanism. We provide extensive experimental results of our method in comparison with existing methods, using different evaluation protocols.
- Chapter 5 introduces a new approach for spatio-temporal representation learning from unlabeled videos. It leverages unsupervised and self-supervised learning and is applicable to both VAD and PID tasks. We perform exhaustive experiments on major public VAD and PID datasets and provide comparative analysis of our method with existing methods.

# Contributions

The scientific contributions made during this thesis and detailed in the chapters of this manuscript have been published in international peer-reviewed conferences and journals.

Our first article was dedicated to the task of fall detection and perimeter intrusion detection. It was published in the Reproducible Research in Pattern Recognition (RRPR) workshop of ICPR 2021 conference (Lohani *et al.*, 2021). Both these tasks are subsets of VAD task and do not contain training labels. In this work, we extended the original work to perimeter intrusion detection task. We also proposed to use precision-recall curve and it provided insightful results on the two tasks. Our next article was published in the special issue "Unusual Behavior Detection Based on Machine Learning" of the Sensors 2022 journal (Lohani *et al.*, 2022b). In this article, we provided mathematical definition of the PID task, along with a review of existing methods and proposed a new suitable evaluation scheme. The work of adaptive PIDS was published in the IEEE International Conference on Image Processing (ICIP) 2022 (Lohani *et al.*, 2022a). In our final contribution, we proposed multiple tasks (unsupervised and self-supervised) for video anomaly detection. We proposed these tasks on a single autoencoder and the main idea is to enrich the spatio-temporal understanding of normality. Experiments were performed in major public VAD datasets and we obtained state-of-the-art results in each of them. This work has been accepted in the International Conference on Computer Vision Theory and Applications (VISAPP), 2023.

# Chapter 1

# Video surveillance: Data and challenges

In this chapter, we first explain how using different acquisition schemes, the data can be acquired in a surveillance system. Then, we explain various forms of challenges associated either with scene capturing, objects of interest or complexity of the scene. Finally, we detail the datasets that we use along with their difficulties and shortcomings.

## Contents

## 1.1 Introduction

Main steps of a modern surveillance system are data acquisition, data pre-processing, data analysis and returning an output signal (like an alarm). In this chapter, we focus on the data acquisition step. This step is very important since the input of analysis part is obtained from here. It is the basis of any surveillance system. The data acquired depends on the type of camera used like visual camera, infrared camera, thermal camera, *etc.* Each of these cameras have their strengths and weaknesses, and therefore they should be used carefully. Thus, in this chapter, we first explain various data acquisition schemes. Since we are interested in outdoor video surveillance, the cameras are placed in the outdoor environment. In outdoor environment, they face various challenges, *e.g.,* challenges due to change in environmental conditions like rain, dust, or bad weather (Hu et al., 2018). Therefore, there are various challenges associated with data in video surveillance and they will be also detailed in this chapter. Finally, we also explore different existing VAD and PID datasets, along with their associated challenges.

## 1.2 Data Acquisition in surveillance systems

In this section, we describe the various data acquisition systems with their associated advantages and drawbacks. The area to be surveyed is observed with the help of cameras. All cameras can acquire the video stream, but the nature of data depends on the type of camera used. Broadly, the following categories of video capture devices are used.

### 1.2.1 Visual Camera

It is also called as the 'classic' or 'color' camera as it captures the visible light in form of grey-scale or RGB images and was the first type of cameras to be used for the video surveillance. The advantage is that it renders an image visually closer to the naked eye. However, for proper functioning, it needs a certain level of brightness in the scene from external light source like sun and furthermore it is sensitive to illumination changes (Van De Sande *et al.*, 2009; Ibrahim, 2016). During night, it needs additional lighting for the sensor to generate a sufficiently contrasted image (Gade et Moeslund, 2014). Adverse weather conditions such as fog, rain, snow, *etc.*, further limit the observation of objects to a short distance from the camera (Robinault, 2021), and thus make detection difficult. Even after all these drawbacks, these cameras have been used extensively in video surveillance systems (Valera et Velastin, 2005; Kim *et al.*, 2011).

### 1.2.2 Infrared Camera

To address the issue of scene brightness, the infrared cameras were proposed as they capture near-infrared emissions from objects (Ibrahim, 2016). To be precise, the term infrared camera is a bit of misnomer because the classic visual camera can also capture the near-infrared emissions, but it is restricted to visible bandwidth using a filter (Robinault, 2021). Therefore, the infrared camera is nothing, but a visual camera coupled with an infrared (IR) lighting. The IR lighting illuminates the scene with near infrared radiation (0.7-1.4 $\mu m$) and the camera captures the reflected radiation from objects in the scene in both the visible and near infrared spectrum (Gade et Moeslund, 2014). Thus, it can work during night too and can provide better contrast when an object moves past the

camera (Haritaoglu *et al.*, 2000). However, it is difficult to detect objects during rain in this camera as rain drops appear as thick stripes in front of the camera (Liang *et al.*, 2012). Furthermore, due to the heat from IR lighting, this camera attracts flying insects and spiders that can raise false alarms and, consequently, impact detection. Figure 1.1 demonstrates these issues with infrared camera.

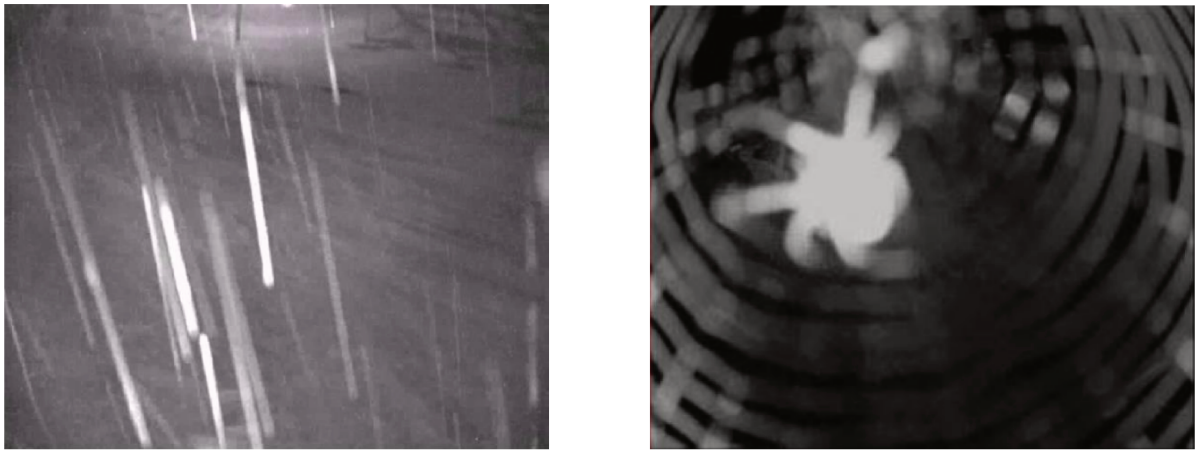


Figure 1.1: Issues with the infrared camera: low visibility in rain (left) and insect on camera (right).

### 1.2.3 Thermal Camera

Unlike the visual and infrared camera, the thermal camera does not require an external energy source for illuminating the scene. In fact, it relies on the infrared radiation emitted by all objects with a temperature above absolute zero. The thermal camera captures this radiation which is in the mid and long infrared wavelength spectrum (3-14 $\mu m$) (Gade et Moeslund, 2014). Since it does not depend on illuminating the scene, it can work even in complete darkness (Ibrahim, 2016). The thermal image is rendered grayscale where the high intensity pixels relate to high temperature. In other words, the higher the temperature of an object in the scene, the brighter it will appear in the thermal image. Advantages of this camera include lower sensibility to weather conditions and object shadows (Liang *et al.*, 2012), and long-range detection (Xu *et al.*, 2016; Hu *et al.*, 2018). However, the main drawback of the thermal camera is that it is difficult to distinguish an object from its background when both have almost the same temperature (Robinault, 2021).

Figure 1.2 shows examples of data acquisition by color and thermal camera. During the day, we can observe that three persons and vehicles are visible in both the cameras. While thermal camera provide better contrast for first two persons and is less affected by shadows, the third person is better visible in color camera since in thermal camera, the object is camouflaged with the car behind it. During night, we cannot see much from the color camera, just the light emitted by an electric scooter. In complete contrast to this, the thermal camera shows the complete scene with details and the person with its scooter is clearly visible.

Figure 1.2: Data acquisition on a site by color (left) and thermal camera (right). Each camera captures two scenes at the same instant, one during day (top row) and the other during night (bottom row).

### 1.2.4 Other Types of Acquisitions

The depth information of an object can be a useful information for its detection (Villamizar *et al.*, 2018; Aravamuthan *et al.*, 2020). This depth information can be added to any camera type using an additional depth sensor. It determines the perspective size of an object in the scene. However, using a depth sensor also has several problems, such as mixed, lost, and noisy pixels in the depth image (Kim *et al.*, 2013; Shao *et al.*, 2014).

Another important information in the scene is the motion information. The camera type specializing in this are known as the event camera (Lichtsteiner *et al.*, 2008). It generates events at the microsecond resolution. An event is triggered each time a single pixel detects a change in intensity value. It finds its application in motion detection, object segmentation, pose estimation, motion tracking, *etc.* (Gallego *et al.*, 2020). Advantages include low latency (no motion blur), high temporal resolution, high dynamic range, and ultra-low power consumption. However, it cannot capture static motion and absolute intensity, and therefore is often used together with other camera types. Event cameras are not used for tasks such as video anomaly detection and perimeter intrusion detection yet, where both spatial and temporal information are essential. The main reasons for this are its extremely high cost and inability to capture visual features, such as texture and color.

Finally, we can also have acquisitions from multi-camera systems where camera types

can be homogeneous or heterogeneous. Intuitively, combining multiple cameras can provide more accurate information about the targeted object and may help to overcome occlusions. Surveillance systems with multiple cameras has been studied extensively (Cai et Aggarwal, 1999; Hu *et al.*, 2004; Wang *et al.*, 2009). Recently, an intelligent perimeter intrusion detection system was proposed using an integrated image acquisition device that combines visual and thermal camera (Kim *et al.*, 2018b). However, a multi-camera system also brings new challenges, such as camera installation, camera calibration, object matching, automated camera switching and data fusion (Hu *et al.*, 2004).

To conclude, no acquisition scheme is perfect. Visual camera is still largely used but it does not work well when the scene illumination is low. The infrared camera can work in low illumination but it suffers in bad weather conditions, particularly during rain, and furthermore infrared lighting attracts insects, which can occlude the view. Even though scene illumination does not affect thermal cameras, but it has other problems like an object is indistinguishable from the background if their temperature is similar. Additional sensors, like the depth sensor, can also be added to any of the camera types to improve scene comprehension. Also, a multi-camera system can be made using various camera types. But even these configurations have their drawbacks. Therefore, the video acquisition scheme should be chosen according to the requirements of the site and concerned task.

## 1.3 Open Challenges

In the context of video surveillance, the cameras are generally fixed in a static position to monitor a site (Valera et Velastin, 2005; Bouwmans, 2011). This site can be either indoors like a bank or agency, or outdoors like an industrial site, private property, or public places such as a city square. We focus on outdoor sites and they are more challenging due to environmental conditions like rain, dust, or bad weather (Hu *et al.*, 2018). The system must be operational continuously for many days and encounter changing light and weather (i-LIDS Team, 2006). Several authors have identified different challenges in the outdoor video surveillance (Bouwmans, 2011; Brutzer *et al.*, 2011). We can group all these challenges into three main categories. The first category is specific to the data acquisition. The second category corresponds to the captured scene as a whole and the third is linked to the objects of interest.

### 1.3.1 Challenges in capturing the scene

These challenges are related to the acquisition and transmission of the video signal.

**a) Noise:** It is characterized by a random variation in pixel intensities or color components (Xu *et al.*, 2016). In video surveillance, there are two main sources of noise (Brutzer *et al.*, 2011). Noise related to the quality of the sensor and noise related to compression. In the case of the sensor, noise is often present when the brightness of the scene is low. This noise is more frequent on thermal cameras (Gade et Moeslund, 2014). Compression noise is related to encoding and depends on the bandwidth available on the network.

**b) Automatic adjustments:** Cameras try, as far as possible, to maximize the dynamics of their sensor to present a correctly contrasted image. They accomplish this with

functionalities like auto focus, automatic gain control, automatic white balance, and auto brightness control (Bouwmans, 2014). These adjustments results in an overall change in brightness and color levels among different frames in the video. This phenomenon is more important on thermal cameras, where the sensors also auto-adjust depending on the scene temperature (Gade et Moeslund, 2014).

**c) Vibrations or jitters:** It refers to all the untimely movements of the camera which cause a displacement of the optical center. We are in the case of a fixed camera where the optical center and all the shooting conditions must not be modified during the acquisition of the video. However, it is not uncommon that the camera experiences some vibrations due to wind or other factors (Xu *et al.*, 2016; Sehairi *et al.*, 2017). These vibrations cause false motions in the scene (Bouwmans, 2014). Mechanical or software solutions can be used to correct this defect (Bouwmans, 2011). If these solutions prove to be insufficient, the same pixel coordinates on two successive images no longer represents the same structure in the scene.

**d) Dirty or Misadjusted lens:** When the cameras are outdoors, projections of rain, dust or bad weather can gradually degrade the quality of the image (Hu *et al.*, 2018; Zou *et al.*, 2019). Likewise, the camera lens adjusted during installation may become out of focus over time. As a result, the video acquired by the camera gradually lose sharpness and becomes blurry (Liang *et al.*, 2012). This problem should not be taken lightly because it corresponds to many real cases. A blurred image causes a significant loss of the information (both texture and color) contained in the scene (Tsakanikas et Dagiuklas, 2018).

This list is not exhaustive. We can add the case of so-called day / night cameras which transmit color frames when the scene brightness is sufficient and switch to grayscale frames when the brightness is low. We can also add the case of cameras with integrated infrared lighting that we have briefly presented previously or the problems of dazzling by vehicle headlights.

### 1.3.2 Challenges related to the complexity of the scene

This second category of challenges is linked to the complexity of the scene, regardless of the presence or absence of objects of interest. In the context of outdoor video surveillance, we can be confronted with a multitude of situations which can range from a parking lot without trees, to a field bordered by tall trees and even a riverbank. In addition, the algorithms must also consider different climatic conditions and different lighting conditions. These different situations and conditions can be summarized by the following set of cases.

**a) Progressive illumination changes:** These variations are generally caused by the change in light conditions which progressively evolve over time and by the course of the sunlight (Bouwmans *et al.*, 2008; Xu *et al.*, 2016). For example, it can be caused by the passage of cloud shadows in the scene.

**b) Sudden illumination changes:** It corresponds to a sudden change in illumination between two successive images (Brutzer *et al.*, 2011; Xu *et al.*, 2016). This change can be either in part of the image or in the full image (Cermeño *et al.*, 2018). It is generally

caused by the activation or deactivation of exterior lighting, but it can also be linked to the sudden introduction of a massive object into the scene (Bouwmans, 2011).

**c) Dynamic background:** Sometimes, some parts of the scene are dynamic and contain movements of irrelevant objects. Examples include tree branches or flag driven by the wind, the reflection of moving clouds on the surface of river, snowflakes, rain, *etc.* This can confuse the system to make false detections (Liang *et al.*, 2012; Bouwmans, 2014; Cermeño *et al.*, 2018).

**d) Shadows:** Managing shadows is another big challenge (i-LIDS Team, 2006; Al-Najdawi *et al.*, 2012). Shadows casted by moving objects of interest or static objects of scene can lead to false detections (specially in grayscale mode) (Liang *et al.*, 2012). Unlike color cameras, the thermal cameras do not have this problem of moving object shadows (Gade et Moeslund, 2014).

**e) Perspective:** This challenge is linked to the fact that we want to detect objects of interest both close and far from the camera. However, the same object does not have the same size depending on its distance from the camera (see Figure 1.3) (Hu *et al.*, 2004). Besides, weather conditions can attenuate the contrast between the object of interest and the background of the scene. This, along with object's perspective, makes detection more difficult (Buch et Velastin, 2014). Furthermore, as the apparent object size decreases with the distance, the apparent object speed also decreases.

Figure 1.3: Effect of perspective: three people (with blue bounding boxes) in the same image with different apparent size. Image source: ShanghaiTech dataset (Luo *et al.*, 2017a).

### 1.3.3 Challenges related to the objects of interest

This last category of challenges concerns more specifically the objects that one seeks to detect in the scene. These challenges are therefore much more linked to the problem we are trying to solve. Most video surveillance tasks face the following challenges.

**a) Camouflage:** Camouflage, whether intentional or not, occurs when the difference in appearance between the object of interest and the background is small (Maddalena et Petrosino, 2008; Brutzer *et al.*, 2011). It is a particularly difficult challenge when the scene is full of different sized objects as shown in Figure 1.4. It is often possible to tune the parameters of an algorithm to detect objects on a short camouflaged video sequence (Cermeño *et al.*, 2018; Kim *et al.*, 2018b). We can therefore manage to detect camouflaged objects by increasing the sensitivity of the algorithm. However, when these very sensitive parameters are used for monitoring a site 24 hours per day, the number of false alarms can become impossible to manage.

Figure 1.4: Camouflaged image with three persons to detect. Source: (Robinault, 2021).

**b) Hole in the object of interest (Foreground aperture):** This problem occurs when objects having insignificant texture difference with the scene background, move slowly in the scene or in the axis of the camera. Due to this, the object pixels are difficult to detect because they can be embedded in the scene background (Maddalena et Petrosino, 2008; Bouwmans, 2014). Thus, the same object may appear as several small objects, as if there are holes in the object.

Among other challenges, we have the speed of objects and their manner of movement (i-LIDS Team, 2006). Very slow or very fast moving objects can be difficult to detect. Furthermore, the manner in which the object moves can create ambiguity, *e.g.,* person crawling or imitating an animal.

## 1.4 Datasets

Since we investigate the problem of abnormal event detection in videos, we explore the concerning datasets. Concretely, we describe the datasets for video anomaly detection (VAD) and perimeter intrusion detection (PID). As mentioned previously (refer Section 1.1), we are interested in systems that operate in the outdoor environment and consequently we choose datasets that follow this criterion.

### 1.4.1 VAD datasets

Anomalies in video can be caused by anything like riding a bicyle, fighting, dancing, playing, running, *etc.* In other words, they depend on context and can differ from one dataset to another. A formal definition for the VAD task will be presented in Chapter 3.

VAD methods rely on a large number of datasets in the literature such as: subway entrance (Adam *et al.*, 2008), subway exit (Adam *et al.*, 2008), UMN (Mehran *et al.*, 2009), UCF-Crime (Sultani *et al.*, 2018), UBnormal (Acsintoae *et al.*, 2022), UCSD Pedestrian (Li *et al.*, 2013), CUHK Avenue (Lu *et al.*, 2013), ShanghaiTech (Luo *et al.*, 2017a), *etc.* However, most of recent works only use the last three datasets of the list for unsupervised video anomaly detection (Zhao *et al.*, 2017; Lee *et al.*, 2018; Liu *et al.*, 2018; Gong *et al.*, 2019; Ionescu *et al.*, 2019; Park *et al.*, 2020; Astrid *et al.*, 2021b; Liu *et al.*, 2021b; Lv *et al.*, 2021; Park *et al.*, 2022). The reasons for not using the other datasets are the following: (a) Subway entrance and subway exit dataset: the scenes are in indoor environment rather than outdoors. No clear training sets, for instance, Cheng *et al.* (2015) use the first 15 minute videos for training, while Lu *et al.* (2013) use more than half of the videos for training; (b) UMN: some scenes are in indoor environment, no clear training and testing split (Ramachandra *et al.*, 2022) and the dataset is already saturated (performance above 99% by many methods) (Acsintoae *et al.*, 2022); (c) UCF-Crime: used for weakly or semi supervised anomaly detection since it contains both annotated anomalies and normal events in the training set (Ramachandra *et al.*, 2022; Acsintoae *et al.*, 2022); (d) UBnormal: a recent dataset but it consists of synthetic data instead of real anomalies.

We detail below the three commonly used VAD datasets, *i.e.*, UCSD Pedestrian, CUHK Avenue, and ShanghaiTech, respectively.

#### 1.4.1.1 UCSD Pedestrian Anomaly Dataset

It is one of the most widely used datasets for video anomaly detection (Kiran *et al.*, 2018). This dataset consists of video clips recorded with a stationary color camera with grayscale frames, overlooking pedestrian walkways on the UCSD campus (Li *et al.*, 2013). It has two distinct scenes, leading to two subsets of the dataset, called as Ped1 and Ped2. The difference between the two subsets is the walking direction (towards and away from the camera in Ped1, parallel to the camera plane in Ped2) and frame resolution ($158 \times 238$ in Ped1 and $240 \times 360$ in Ped2). Most works use only the Ped2 dataset (Gong *et al.*, 2019; Ionescu *et al.*, 2019; Feng *et al.*, 2021; Georgescu *et al.*, 2021a; Liu *et al.*, 2021b; Cho *et al.*, 2022; Park *et al.*, 2022) because Ped1 has a considerably low video resolution (Hinami *et al.*, 2017; Doshi et Yilmaz, 2021; Ouyang et Sanchez, 2021; Le et Kim, 2022), is suitable for pixel-level and not frame-level anomaly detection (Xu *et al.*, 2015a; Luo *et al.*, 2019) and has ambiguity in anomaly annotation (some events labeled as normal in training set are considered as anomalies in testing set) (Nguyen et Meunier, 2019). For all these reasons, we also only focus on the UCSD Ped2 dataset.

There is a constant movement of people in this dataset, and it is considered normal. The abnormal events are due to the circulation of non-pedestrian entities, *i.e.*, bikers, skaters, carts, and wheelchairs. All these events occur naturally, *i.e.*, they were not staged for dataset collection. Figure 1.5 shows illustrative normal and anomalous frames of this dataset. This dataset contains 28 videos in total, with 120 to 200 frames per video. The training set has 16 videos (2,550 frames) without anomalies and the testing set comprises of 12 videos (2,010 frames) with anomalous events. Both pixel-level and frame-level annotations are provided with this dataset. The following challenges are present in this

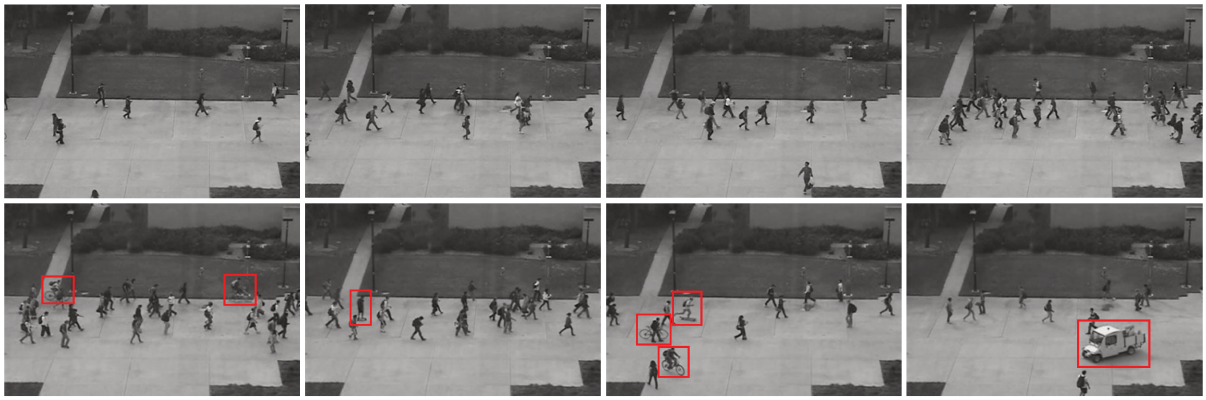


Figure 1.5: Example frames from the UCSD Ped2 dataset. First row contains frames without any anomaly, with increasing crowd volume (left to right order). Second row shows anomalous frames with anomalies highlighted in red bounding boxes.

dataset: dynamic background, shadows, perspective, and camouflage.

Despite being widely used, this dataset has various shortcomings. First, it is one of the smallest datasets in terms of videos, number of frames, total anomalies, and number of anomaly types. Therefore, most methods have almost perfect results on this dataset. Second, all the anomalies can be detected by only analyzing a single frame at a time. In other words, the anomalies have a dominant appearance / spatial difference from normality and it is easy to detect them based on spatial features, without requiring the temporal characteristics.

#### 1.4.1.2 CUHK Avenue Dataset

It consists of one scene captured in the Chinese University of Hong Kong (CUHK) campus avenue with a fixed color camera (Lu *et al.*, 2013). People walking towards various predefined directions is considered as a normal event. The abnormal events are due to either non-pedestrian objects like bicycle or unexpected pedestrian behavior like running, loitering, jumping, throwing bag, walking in prohibited direction, *etc.*. We must highlight that this was the first dataset to introduce loitering (static) as an anomaly, which is important for surveillance applications. Unlike UCSD Ped2, anomalies are mostly staged in this dataset. Some illustrative normal and anomalous frames of this dataset are shown in Figure 1.6

This dataset contains 16 video clips (15,328 frames) for training and 21 for testing (15,324 frames), with a frame resolution of 360 $\times$ 640. The training videos consists of normal events with the exception of few videos with small occurrences of anomalies. The testing videos have both normal and abnormal events. Like UCSD Ped2, we have both the pixel-level and frame-level annotations. This dataset presents the following challenges: camera shakes, perspective, dynamic background, and progressive illumination changes.

Avenue dataset also has still some deficiencies. Like UCSD Ped2, we have a very small amount of data for deep neural networks and some authors argue that the amount of normal events is not sufficient for training (Li et Chang, 2019). Furthermore, the annotations are ambiguous for some test videos, leading to their exclusion by some works (Hinami *et al.*, 2017; Ionescu *et al.*, 2019; Ouyang et Sanchez, 2021).

Figure 1.6: Example frames from the CUHK Avenue dataset. First row contains frames without any anomaly but with different scene dynamics. Second row shows the following anomalies in red bounding boxes (left to right order): bicycle, throwing bag and running person, loitering and person walking in wrong direction.

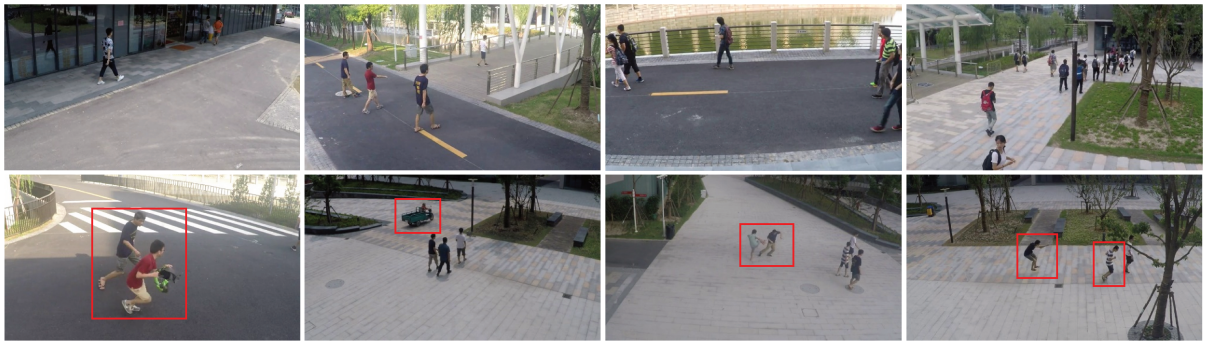


Figure 1.7: Some frames from the ShanghaiTech dataset, each from different scene. First row contains frames without any anomaly but with different scene dynamics. Second row depicts the following anomalies in red bounding boxes (left to right order): chasing, small truck, brawling and two people jumping.

#### 1.4.1.3 ShanghaiTech Campus Dataset

All previous datasets had a single camera view (or scene) in training and testing set. Having only a single view might lead some models to overfit by memorizing the fixed background. That is why it is essential to have different views in the dataset, so that the proposed detection method is aware of different scenes and generalizes to them. The ShanghaiTech Campus dataset is one of the most challenging datasets for video anomaly detection and includes 13 different views with various lighting conditions and camera angles (Luo *et al.*, 2017a). The videos are filmed at the ShanghaiTech university campus using color cameras with the frame resolution of 480 $\times$ 856. Pedestrians walking in the campus with regular pace is considered as a normal event. Abnormal events are caused either by the movement of unauthorized objects like bicycle, skates, strollers, *etc.* or by irregular pedestrian behaviors like brawling, chasing, jumping, throwing objects, robbery, *etc.* Figure 1.7 illustrates various normal and abnormal events in this dataset.

It is one of the biggest anomaly detection datasets with 330 training (274,515 frames) and 107 test videos (42,883 frames). Like other datasets, training videos only contain normal events and testing videos have both normal and abnormal events. For each anomalous event both pixel-level and frame-level ground-truth annotations are available. It is worth noting that recently Zhong *et al.* (2019) proposed a reorganization of this dataset to also have annotated anomalies during training for weakly supervised learning. We focus on

| Dataset | Number of frames | | | | | #Anomalies | #Anomaly types |
|---|---|---|---|---|---|---|---|
| | Total | Train | Test | Normal | Abnormal | | |
| UCSD Ped2 | 4,560 | 2,550 | 2,010 | 2,924 | 1,636 | 23 | 5 |
| Avenue | 30,652 | 15,328 | 15,324 | 26,832 | 3,820 | 47 | 5 |
| ShanghaiTech | 317,398 | 274,515 | 42,883 | 300,308 | 17,090 | 158 | 11 |

Table 1.1: Description of video anomaly detection datasets.

the original dataset setting as we focus on developing methods without annotations. This dataset presents the following challenges: noise, camera shakes, shadows, perspective, dynamic background, progressive and sudden illumination changes.

We still have the following deficiencies in this dataset. Despite being a very large dataset, we have only a small number of anomalies (158) belonging to only 11 anomaly types. Therefore, it lacks diversity and number of anomalies. Similarly, since there are overall 13 scenes, we have a small training set for each scene. This small per-scene training set is not fully representative of all the normal activities (Ramachandra *et al.*, 2022). Therefore, many works tend to use some form of data augmentation techniques to have enough representative data (Astrid *et al.*, 2021a; Feng *et al.*, 2021).

Table 1.1 summarizes the three datasets with frame-level details. It must be noted that none of these datasets have a validation set, which makes it difficult to choose hyperparameters while training any model.

### 1.4.2 PID datasets

A perimeter intrusion detection system (PIDS) concerns with the detection of intrusions in a secured perimeter. Like VAD, the intrusions depend on the context and vary from one dataset to another. A mathematical definition for PID task will be presented in Chapter 3. PID can include various subtasks such as detection and tracking, therefore historically algorithms were tested on the datasets of these subtasks. For example, CAVIAR (Crowley *et al.*, 2005), PETS2006 (Thirde *et al.*, 2006) and AVSS2007 (AVSS dataset, 2007) have been used to test the tracking module of the PIDS (Vijverberg *et al.*, 2010, 2013; Nayak *et al.*, 2019). Concerning the PID task, most works use their private datasets. One recent work introduces a new dataset, called SIC (Cermeño *et al.*, 2018), but it is available under strict conditions and, without annotations, thus omissions cannot be evaluated. Finally, there is only one public dataset, dedicated for the PID task: the i-LIDS sterile zone dataset (i-LIDS Team, 2006) and it has been extensively used in the literature (Buch et Velastin, 2008; Vijverberg *et al.*, 2013, 2014; Buch et Velastin, 2014; Cermeño *et al.*, 2018).

#### 1.4.2.1 i-LIDS sterile zone dataset

The imagery library for the intelligent detection systems (i-LIDS) consists a library of closed-circuit television (CCTV) video footage datasets for benchmarking video analytics systems. Concerning the PID task, they propose a sterile zone dataset, known as the i-LIDS sterile zone dataset (i-LIDS Team, 2006). This dataset comes with a proper annotation and evaluation procedure to ensure its relevance for the industrial application.


Figure 1.8: Example frames drawn from i-LIDS sterile zone dataset with various intrusion (intruders in red boxes) and non-intrusion frames. The color and black and white frames belong to view 1 and view 2 respectively. The time of the day in four columns are dawn, dusk, night, and day. The distractions (in blue boxes) in non-intrusion row (left to right order) are a rabbit, birds with rain, fox and an insect on camera respectively. In intrusion row (left to right order), we have two intruders, intruder with ladder during snow, log rolling intruder and crawling intruder respectively.

The PID objective in this dataset is to detect the presence of people in a sterile zone. It consists of two sites monitored by two different cameras (view 1 in color/monochrome and view 2 in monochrome) as shown in Figure 1.8. There is a security fence in each site (view) and the aim is to detect intrusion as soon as it enters the site. Intruders try to breach the fence in various ways. For example, people may walk, run, crawl, or roll towards the fence and, on occasion, may carry climbing aids, such as a ladder. The intruders can be at three different distances from the camera: close, middle, and far. The cameras recordings are over many days, capturing different times of the day, such as dawn, day, dusk, and night. They also include various weather conditions, such as cloudy, rainy, snowy, and foggy. Furthermore, there are various distractions that might trigger false alerts, like plastic bag or paper moving due to wind, bats, birds, foxes, insects, rabbits, squirrels, shadows through the fence, *etc.* Figure 1.8 illustrates various distractions and intrusion examples. All these factors like different weather conditions, different times of the day, numerous distractions, and various ways in which the intruder approaches the fence make it a very challenging and realistic dataset. Finally, the challenges present in this dataset are noise, automatic adjustments (switch from color to monochrome images and vice-versa), vibrations, progressive / sudden illumination changes, shadows, perspective, and camouflage.

It has two separate disks for training and testing. The PID system is developed using the training disk data and its performance is evaluated with the testing disk data. Each disk contains the two camera views, with over 20 hours of video recorded in the various previously cited situations. All videos are taken at 25 FPS with 720 $\times$ 576 frame size resolution. Each disk contains annotations, distractions, and other information for each video. The annotation provided is the time interval of each intrusion event in the video *i.e.*, the entry and exit time of people in the respective scene.

Table 1.2 summarizes the i-LIDS dataset where view 1 and view 2 corresponds to

| View | Set | #Videos | #Intrusions | Number of videos per intrusion count | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 1 | 10 | 13 | 15 | 17 | 31 |
| View 1 | train | 123 | 113 | 10 | 113 | 0 | 0 | 0 | 0 | 0 |
| | test | 17 | 113 | 10 | 0 | 2 | 1 | 1 | 2 | 1 |
| View 2 | train | 113 | 103 | 10 | 103 | 0 | 0 | 0 | 0 | 0 |
| | test | 16 | 103 | 10 | 0 | 1 | 1 | 1 | 2 | 1 |

Table 1.2: i-LIDS Sterile Zone dataset description.

the color camera and monochrome camera respectively. Each view has videos with and without intrusions. In both training and testing disk, we have 10 videos from each view without any intrusions. In training disk, we have 216 short videos ($\approx$ 3 minutes), with single intrusion per video. These videos can be used for training and/or validation for example. The testing disk have 7 and 6 very long videos in view 1 and view 2. These long videos range from 36 minutes to 92 minutes and contain between 10 and 31 intrusions per video. Even though the total number of intrusions are also 216 in testing disk, the long-length videos makes it more challenging. All i-LIDS videos contain various distractions and/or intruders but these long videos also have a drastic change in weather and luminosity. This makes this testing dataset even more relevant from practical perspective. Furthermore, the distractions, intrusions and other information is very well distributed in both the views and thus we do not have bias for the choice of a camera view. It must be noted that like VAD datasets, it does not contain an official validation set. However, in unsupervised context, the intrusion videos from the training set can be used as a form of validation set.

Although we have extensive real-life elements in i-LIDS dataset, it has some major drawbacks. First, we only have people as intrusion. This is a limitation because in real-life, intrusion can very well be some other object like a car, a bike, a truck, *etc.*. Consequently, any good human detector system might work very well in this dataset but that does not necessarily mean that it is a good PIDS. The second important limitation is that we have only two views with very similar settings. This makes it easy for the algorithms (particularly supervised learning based) to learn the scene. A multi-view dataset would have added an additional difficulty in the PID task.

## 1.5 Conclusion

In this chapter, we explored the data acquisition schemes in the surveillance systems, followed by the common challenges and finally the datasets used for the VAD and PID tasks.

Visual camera or color camera is still one of the most used cameras for video surveillance, even though it does not perform well when the scene has low illumination. The infrared camera eliminates this problem of visual camera with infrared lighting, but this lighting has other problems, like it attracts insects and performs poor in rainy condition. Thermal camera relies on the infrared radiation emitted by the objects and hence it does not require an external lighting, but it also has problems like we cannot distinguish an object from its background if their temperature is similar. Since none of the video acquisition device is perfect, often an additional sensor like the depth sensor is used, or multiple

camera types are used as a multi-camera system, but even these configurations have their drawbacks. To conclude, one should use the video acquisition scheme depending on the requirements of the site and concerned task.

We highlighted various challenges for surveillance systems into three categories: concerning capturing the scene, concerning complexity of the scene, and concerning objects of interest. The first category comprises challenges like noisy frames, camera vibrations and gain adjustments. The second category includes challenges like illumination changes, dynamic background, scene perspective and shadows. Finally, the last category includes camouflage and foreground aperture as challenges.

We explored various datasets for the VAD and PID tasks with their difficulties and shortcomings. Concerning VAD task, three datasets are mainly used. Out of these three datasets, UCSD Ped2 and CUHK Avenue are single scene dataset, *i.e.,* they have only one scene for training and testing. Ped2 is the smallest and easiest dataset, having anomalies with distinguishable appearance like bike, skate, *etc.* Avenue dataset is slightly more difficult as it also have anomalies where the normal object has temporal irregularity, like person throwing a bag, running, *etc.* Finally, the ShanghaiTech dataset is the biggest and most challenging dataset among the three, with 13 different scenes captured by color cameras. There is a good variety of anomalies in this dataset and contains both appearance and temporally irregular anomalies. The fact that the same model needs to understand 13 different scenes and detect anomalies is challenging. However, none of these datasets are perfect: they do not have a validation set, they lack proper anomaly description and correct test annotations (some normal events like ice-cream eating are present in test set but not the training set, causing ambiguity), they lack diversity and have small amount of anomalies, and they contain only short video clips. In real-life, the VAD system needs to protect a site for a long time (atleast few days or weeks) and must face changing scene dynamics like weather, lighting, *etc.*, therefore longer videos are essential. Ideally, we need a new multi-view dataset of long videos, having appropriate anomaly description and annotations, containing diverse anomalies, and accompanied with a validation set.

For the PID task, we have only one public dataset, *i.e.*, the i-LIDS sterile zone dataset and it has just two camera scenes. It is challenging as intruders enter the scene in various ways (like crawling), and there are distractions caused by animals/insects, along with changing scene dynamics like rain, snow, *etc.*. There is no official validation set in this dataset. It also lacks intrusion types and has only person as intruder. Furthermore, the two scenes are very similar to each other, they are like mirror images of each other. Therefore, we require more datasets for the PID task, each with a validation set. They should have multiple scenes and diverse intruder types like cars, bikes, *etc.*

# Chapter 2

# Literature review

This chapter presents a review of existing works for spatio-temporal feature learning in videos, with main focus on video anomaly detection and perimeter intrusion detection task. We first present all necessary background concepts needed for understanding the chapter. We next explore the existing spatio-temporal feature learning approaches, followed by unsupervised approaches. In the next two sections, we review existing methods concerning video anomaly detection and perimeter intrusion detection. Finally, we conclude this chapter with takeaway points along with our chosen direction to tackle these problems.

## Contents

## 2.1 Introduction

In this chapter, we present the literature review of existing works for spatio-temporal feature learning on videos, with particular focus on two unsupervised tasks, *i.e.,* video anomaly detection and perimeter intrusion detection. A video consists of spatial and temporal dimensions. We need to understand these space-time characteristics of the video, in order to have a meaningful interpretation of it, which helps in performing various tasks and applications like video object recognition, segmentation, anomalous behaviour detection, *etc.* Many works address this, but they use labeled data. However for some tasks, there is no available annotated data, such as for video anomaly detection and perimeter intrusion detection. Therefore, we need methods that can understand the spatio-temporal characteristics of videos without labeled data. In this chapter, we will first present some background concepts. Then, we will explore various ways for space-time feature learning from video, followed by reviewing unsupervised approaches. We will then consider methods of literature concerning video anomaly detection and perimeter intrusion detection respectively.

## 2.2 Background

This section aims to present in a concise manner all necessary background concepts required for understanding the subsequent sections. These concepts are on different types of Artificial Neural Networks (ANNs), which aims at learning patterns within data (Bishop et Nasrabadi, 2006). ANNs originated from various attempts to represent the human brain from a mathematical perspective (McCulloch et Pitts, 1943; Rosenblatt, 1957). The underlying hope was that such representation would provide with pertinent learning capabilities.

The main idea of the neural networks is to construct a model parameterized by weights (from a few hundreds up to a trillion (Fedus *et al.*, 2022)) in order to fit a given target function. The learning of models is done with gradient descent and more specifically by using the backpropagation algorithm (Rumelhart *et al.*, 1986). These models are constructed by stacking layers and various types of layers were proposed to handle different types of inputs. For example, convolutional layers (Fukushima et Miyake, 1982) were introduced to process images. They have a translational invariance property, which is highly useful for tasks such as object detection in images. For sequential data, recurrent layers were introduced (Jordan, 1990). These layers keep in an internal state, the memory of previous inputs, to help future outcomes.

In the following subsections, we begin reviewing the core concepts of neural networks via the standard feedforward network. Then, we explain the convolutional neural network, followed by recurrent neural network, autoencoder and finally the generative adversarial network.

### 2.2.1 Feedforward neural network

ANNs use artificial neurons that were designed to loosely mimic the biological human neuron. A human neuron receives inputs through axons and dendrites and, depending on the intensity of the electrical impulse received, in turn generates an impulse to subsequent neurons. For ANN, such functioning is formalized (McCulloch et Pitts, 1943) using the following equation:

$$f(\mathrm{x}, \mathrm{w}) = \sigma\left(\sum_i w_i x_i + b\right) \tag{2.1}$$

Each input signal $x_i$ is multiplied by a dedicated weight $w_i$, the resulting signals are summed along with a bias term and fed to an activation function $\sigma$ that decides if the neuron should activate or not, and propagate the signal to subsequent neurons. Any function can serve as activation function, as long as it is continuous and non-linear. Typical activation functions are described in next subsection.

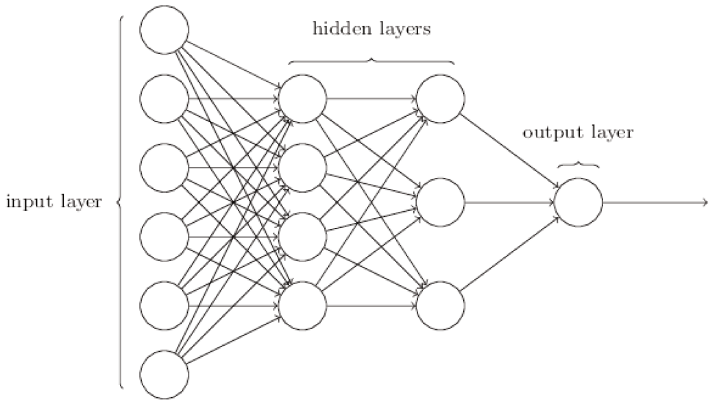



Figure 2.1: Pictorial representation of a feed-forward neural network. Source: (Nielsen, 2017)

The neurons are stacked to make a layer and stacked layers form a neural network model. The first layer is the raw input data (for example, pixel values of an image or video) and subsequent layers, called hidden layers, extract more abstract information of the input until one reaches the final layer, called the output layer. Figure 2.1 depicts a model with four layers. It is a feedforward neural network, where the term feedforward means that the information flows strictly in a forward direction from the input to the output. It is a fully connected network (or dense network) since each neuron on a layer is connected to all the neurons of the preceding layer. In other words, the activation value of a neuron depends exclusively on the activation values of the neurons of the preceding layer. Concretely, the activations of the neurons in the $l^{th}$ layer, *i.e.,* $\mathrm{x}^{(l)}$, are a linear combination of the values of the preceding layer $\mathrm{x}^{(l-1)}$, passed through a nonlinear function $\sigma$:

$$\mathrm{x}^{(l)} = \sigma\left(\mathrm{W}^{(l)}\mathrm{x}^{(l-1)} + \mathrm{b}^{l}\right) \tag{2.2}$$

where $\mathrm{W}^{(l)}$ and $\mathrm{b}^{(l)}$ are the weights matrix and the bias vector respectively. These are the parameters that are tuned during the training stage of the neural network. Usually, $\mathrm{x}^{(0)}$ is referred to as input, $\mathrm{x}^{(N)}$ (N being the number of layers in a model) as prediction or output and the remaining $\mathrm{x}^{(i)}$ $(i = 1, \ldots, N-1)$ as hidden layers.

#### 2.2.1.1 Activation functions

An activation function defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the neural network (Haykin et Network, 2004; Goodfellow *et al.*, 2016). The choice of activation function has a crucial impact on the capability and performance of the neural network, and different activation functions may be

used in different parts of the network. Typically, all hidden layers use the same activation function, while the activation function for the output layer depends upon the type of prediction required. Since neural networks are typically trained using the backpropagation algorithm (refer next subsection) that requires the derivative of prediction error in order to update the weights, therefore activation functions are also typically differentiable.

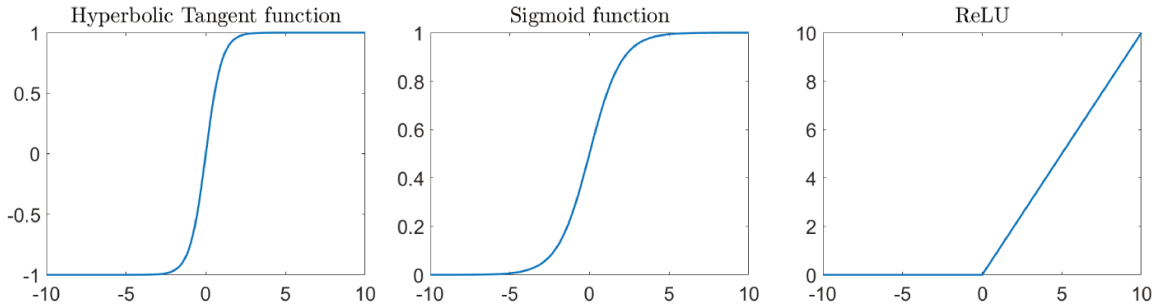



Figure 2.2: Common activation functions.

Some commonly used activation functions are shown in Figure 2.2 and are summarized below:

- **Sigmoid.** The sigmoid or logistic function squashes the input in the range 0 to 1.

$$\sigma(v) = \frac{1}{1 + e^{-v}} \tag{2.3}$$

  As shown in Figure 2.2, the large values of input incline towards the value 1 in output, while the small values tends closer to 0. This property of sigmoid output can be seen as a probability distribution, and thus it is used in binary classification (with single output neuron) to predict whether a certain example belongs to a fixed pre-defined class or not. However, there are two main issues with this activation function. The first issue is of the vanishing gradient. Since the output of sigmoid saturates (*i.e.* the curve becomes parallel to $x$-axis) for a large positive or large negative number, the gradients at these regions is almost zero. When $n$ hidden layers use an activation like the sigmoid function, $n$ small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers. A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training iteration, thus making the learning inefficient. That is why, sigmoid function is not used in hidden layers. The second issue is that the sigmoid outputs are not zero-centered and it is undesirable because it can indirectly introduce zig-zagging dynamics in the gradient updates for the weights.

- **Hyperbolic tangent.** It is also known as the tanh activation and it squashes the input to [-1, +1] range.

$$\sigma(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}} \tag{2.4}$$

  It is very similar to the sigmoid activation and even has the same S-shape as shown in Figure 2.2. Since the output is centered at zero, we do no have the problem of zig-zagging dynamics in gradient updates like sigmoid. We still have the same issue of vanishing gradient but the range of inputs that do not saturate are larger here.

Like sigmoid, it is not suitable for hidden layers but can be used in output layer for binary classification. Since the output of tanh is symmetric around zero, it leads to faster convergence of the neural network.

- **ReLU.** The rectified linear activation function (ReLU) is a piecewise linear function that outputs the input if it is positive, otherwise, it outputs zero. It is defined as:

$$\sigma(v) = max(0, v) \tag{2.5}$$

It is less susceptible to the vanishing gradient problem since the gradient for all inputs greater than 0 is 1. This is why it is one of the most used activation functions and is mostly used in hidden layers. The main issue with this activation function is called the dying ReLU. It occurs when the neuron gets stuck in the negative side and constantly outputs zero. Because gradient of 0 is also 0, it's unlikely for the neuron to ever recover. This happens when the learning rate is too high or negative bias is quite large. To address this issue, Leaky ReLU is proposed, which allows small negative values when the input is less than zero. ReLU or its variants are usually not in the output layer.

- **Softmax.** It is not a classic activation function that can be applied on a single neuron. The softmax function turns a vector of $k$ real values into a vector of $k$ real values that sum to 1. The input values can be positive, negative, zero, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. It is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution (Bishop et Nasrabadi, 2006). It can be seen as a generalization of the sigmoid function which was used to represent a probability distribution over a binary variable (Goodfellow *et al.*, 2016). Since softmax converts the input to a normalized probability distribution (between 0 and 1) that sum to 1, it is used in output layer of networks that do multi-class classification. It is not used for hidden layer activations. Furthermore, it should not be used for multi-label classification, *i.e.,* when an example have more that one labels like a dog and a bone. This is because softmax simply cannot produce more than one label with values close to 1. In this case, sigmoid function is used.

We presented above some of the most common activation functions, with particular focus to functions that are later used in this thesis. There are also other activation functions like maxout, gaussian linear unit (GLU), exponential linear unit (ELU), *etc.* For hidden layers, ReLU activation function is the most used. However, the application and architecture of the neural network often dictates which functions to use, for example in LSTM (a type of recurrent neural network), sigmoid and tanh functions are used as a ReLU would make learning unstable and maybe even impossible. Finally, for binary class classification with single output value, sigmoid function is used in output layer. Similarly, for multi-class classification, softmax function is the preferred activation function in output layer.

#### 2.2.1.2 Training

So far we detailed the general architecture of the neural networks along with the various activation functions, we now review the training procedure used to estimate the target function. The aim is to train a model $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ to approximate a unknown target

function, where $\mathcal{X}$ is called the *input space* and $\mathcal{Y}$ the *output space*. The training is done to extract the statistical properties of the inputs in order to maximize the posterior probability $p(\mathrm{y}|\mathrm{x})$ of getting the correct $y \in \mathcal{Y}$ given $x \in \mathcal{X}$. This maximization is done through the minimization of an objective function:

$$\min_{\mathrm{W}} \ \mathbb{E}_{\mathrm{x,y}} [\ \mathcal{L}(\mathcal{M}(\mathrm{x}, \mathrm{W}), \mathrm{y})\ ] \ , \tag{2.6}$$

where $\mathbb{E}$ is the expectation, $\mathcal{M}$ is the deep network, W represents the weights of $\mathcal{M}$ to be set and $\mathcal{L}$ is the loss function measuring how close $\mathcal{M}(\mathrm{x}, \mathrm{W})$ is to the true output. To do this minimization, gradient descent is usually used. The standard way to do such an optimization is Stochastic Gradient Descent (SGD) (Bottou *et al.*, 1991). It computes the average gradient

$$\Delta \mathrm{W}_{k-1} = \frac{1}{|B|} \sum_{i \in B} \Delta_{\mathrm{W}_{k-1}} \mathcal{L}\left(\mathrm{y}_i, \mathcal{M}\left(\mathrm{x}_i, \mathrm{W}_{k-1}\right)\right) \ , \tag{2.7}$$

where $(\mathrm{x}_i, \mathrm{y}_i)$ is the $i$-th element of the current batch $B$ sampled from the training data, and updates the weights using the backpropagation algorithm (Rumelhart *et al.*, 1986) as:

$$\mathrm{W}_k = \mathrm{W}_{k-1} - \eta \Delta \mathrm{W}_{k-1} \ , \tag{2.8}$$

where $\Delta \mathrm{W}_{k-1}$ is the gradient at iteration $k$ and $\eta$ is called the learning rate. For such an optimization method, the speed and convergence towards a minima is highly dependent on the shape of the surface of the objective function with respect to the weights. As shown in Figure 2.3a, the optimization process may bounce back and forth, as the slope is very steep (therefore creating a strong gradient). Multiple update rules have been further developed to prevent such problem: momentum (Qian, 1999) (Figure 2.3b), Adagrad (Duchi *et al.*, 2011), RMSprop (Tieleman et Hinton, 2012), Adam (Kingma et Ba, 2015), Adadelta (Zeiler, 2012), to name the most common.

(a) SGD without momentum

(b) SGD with momentum

Figure 2.3: SGD optimization with and without momentum. On the left, without momentum the optimization process bounces between the ravine's slopes whereas, on the right, with momentum, optimization is smoother. Source: (Orr *et al.*, 1999).

### 2.2.2 Convolutional neural networks

In the last subsection, we detailed the basic architecture of a neural network as well as the training procedure used to optimize it. However, so far, we have only considered one type of layer, *i.e.,* the dense layer, where each output neuron is connected to each input through a weight. While this layer is functional, it is not best suited for all types of input. For instance, for processing images, each output neuron would be connected to each pixel,

therefore requiring a tremendous amount of weights. These weights would not only be hard to optimize but can also lead to overfitting. Furthermore, the dense layer is not invariant to translations. Translation invariance is a very desirable property for image processing and it refers to the ability to ignore positional shifts, or translations, of the target in the image. For example, a cat is still a cat regardless of whether it appears in the left half or the right half of the image.

To address these problems, layers that operate in a sliding window fashion can be used. As the window slides over the input image, weights are re-used at every target position. Therefore, translated inputs result in feature maps with translated activated neurons. Moreover, as the computations of each neuron within a feature map are independent, one can leverage the parallelism to speed up processing. Layers with such behavior are called convolutional layers.

**a) Convolutional layer** A convolution operation can be mathematically defined as:

$$o(t) = (a * b)(t) = \int a(z)b(t-z)dz \tag{2.9}$$

where $a$ is input, $b$ is kernel and $o$ is the resultant output obtained by convolving input with kernel. This can be interpreted as a weighted operation applied around the neighborhood of the point $t$. We can easily extend this convolution operation to multiple dimensions. For image data, we have discrete data as input in the form of pixels. Equation (2.9) can be extended for image data as:

$$O(i,j) = (I * K)(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.10}$$

where $I$ is input image, $K$ is kernel and $O$ is the convolved output at pixel $(i,j)$. This output is also known as the feature map. Since convolution is commutative, we can observe that in the equation $(I * K) = (K * I)$. The region in the input space (here image) that the kernel is looking at the current point is known as the receptive field. Finally, the displacement in the image that the kernel performs after each convolution operation is termed as the stride. Figure 2.4 shows an example of convolution operation applied on an $7 \times 7$ image using a kernel of size $3 \times 3$ and stride $1 \times 1$. The orange region in the image shows the receptive field of the kernel and green unit in feature map shows the corresponding convolved output.

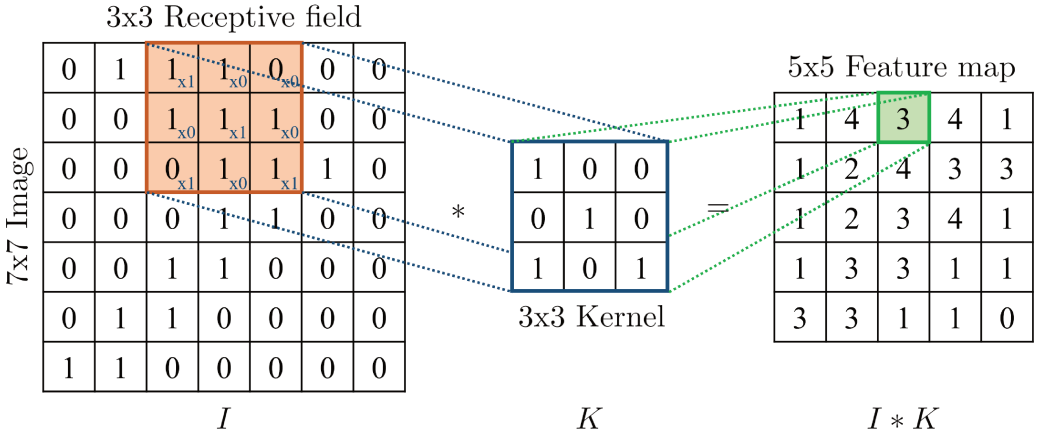



Figure 2.4: Convolution of an image $I$ with kernel $K$ using stride $1 \times 1$.

It is to be noted that various modifications to this classic convolution can be obtained by increasing kernel stride, or padding the input image, or using dilation. The reader can refer to the work of Dumoulin et Visin (2016).

**b) Pooling layer** This layer is also known as the downsampling layer. It is often placed after obtaining the feature map and it summarizes the most pertinent values of it. In other words, convolutional layers detect features, while pooling layers aggregate them. Pooling layers reduce the amount of weights or parameters in the network, thus speeding up the computation and reducing the chance of overfitting the training data. Like convolution, this layer also have a kernel or filter, which performs the pooling operation over the feature map and then this filter is slided with a certain stride. Two most common pooling operations are max and average, *i.e.,* taking maximum and average of the feature map for a given filter size. Figure 2.5 demonstrates these two operations. The feature map and filter size are $4 \times 4$ and $2 \times 2$ respectively. The filter is first placed on the top left of feature map (shown with yellow color) to perform the pooling operation (max or average), resulting in value (yellow color) shown in right side of the figure. Then, the filter displaces with stride two to reach the top right (shown in blue color) and performs the pooling operation again. This pooling process continues until the full feature map is covered. Finally, we can observe the overall results of max-pooling and average pooling layers in the figure.

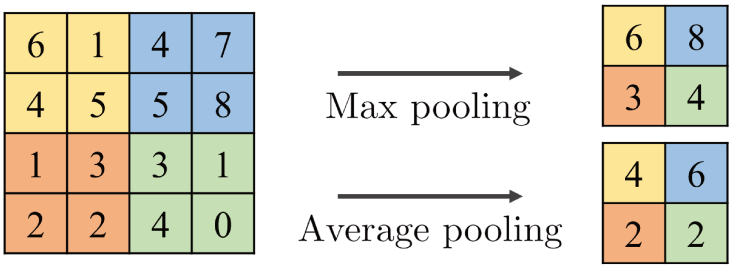



Figure 2.5: Examples of max pooling and average pooling operations with filter size and stride as two.

Convolutional neural network (CNN) is a type of neural network that uses convolution in at least one of its layers. However, given a model with a single convolutional layer, the context (receptive field) a given neuron can gather is limited to the size of the kernel. Increasing context information requires to increase the size of the kernel and therefore the computation needed as well as the number of weights. A simple solution to this problem is to stack multiple convolutional layers. By doing so, the receptive field will naturally increase, while keeping the desired translational invariance property. Therefore, typical CNNs are composed of many convolutional layers stacked on top of each other, where each layer is followed by an activation function like a standard ANN. As we advance in the layers, the size of feature map should generally decrease. For example, for image classification task, we need a single output from the image. To do so, generally pooling layers are used after each convolutional layer. Alternatively, strided convolutional layers can also be used as they decrease the feature map size. Finally, at the end of last layer in CNN, often a fully-connected (dense) layer is used and it reduces the final ouput to desired dimensions (like dimension 1 for image binary classification). The basic mechanism of training a CNN remains the same as for regular neural networks. Like weights of the

feed-forward network, in CNN, the kernel parameters are meant to be learned. The initial weights of an untrained CNN are randomly chosen. Then given the training data and task, a loss function is used to calculate the error. This error is backpropagated to update the kernel parameters. Like the regular ANN, the training is continued until the model converges.

To conclude, CNNs are extremely versatile tools. Thanks to their limited number of weights, they help mitigate over-parameterization problems inherent to dense layers. Furthermore, they are also translation invariant and allow for easy parallelisation by design. As a result, convolutional layers are used nowadays throughout the scientific literature. For instance, 1D-CNNs have been used for automatic speech recognition (Kiranyaz *et al.*, 2021) and 2D-CNNs are the standard for image processing (Karpathy *et al.*, 2014; Goodfellow *et al.*, 2016; Liu *et al.*, 2016; Krizhevsky *et al.*, 2017).

### 2.2.3 Recurrent Neural Networks

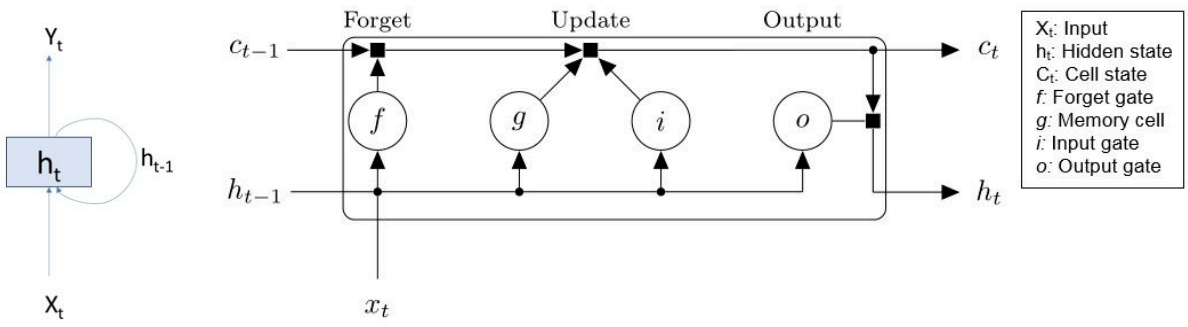
For the moment we described networks that takes as input spatial signals such as images. However in this manuscript, we are also interested in working with sequential data since we want to extract information from videos. In this case the input data $x = (x_1, \ldots, x_t, \ldots, x_T)$ is a sequence composed of $T$ elements.

By design, the feedforward and convolutional neural networks are not suited to model long sequential data like a time series, speech, or text. Besides, the CNN can only gather information up to the receptive field size. Although such behavior might be sufficient for entries of fixed shape, for entries of varying length, it would be better to manage to implement some form of memory that can gather a potentially unlimited amount of information. For this purpose, there is another neural network variant, called recurrent neural network (RNN) (Jordan, 1990). RNNs employ hidden vector denoted $h$ which is recursively updated at each timestep using the element from the input, and from which the output $y$ is predicted as:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.11}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{2.12}$$

where $\sigma_h$ and $\sigma_y$ are activations function and $W_h$, $W_y$, $b_h$, $b_y$, $U_h$ are parameters matrices and biases.




(a) RNN (b) LSTM

Figure 2.6: Pictorial representation of a simple RNN and an LSTM. Source: (MathWorks, 2022).

As shown in Figure 2.6a, in an RNN, the information feeds cells through a loop. When it makes a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously. This is also demonstrated from Equations (2.11 and 2.12). The RNN has two inputs, the present and the recent past. This is important because the sequence of data contains crucial information about future information. It allows the previous inputs to affect the current output. Due to this internal memory, they have a particular advantage to form a much deeper understanding of a sequence and its context. Classical RNN suffer from two major problems: exploding and vanish gradient (Le Cun *et al.*, 1997). The first problem refers when algorithm assigns high importance to the weights, without much reason. The second issue occurs when the values of a gradient are too small and the model stops learning.

These problems were solved with the introduction of the Long Short-Term Memory (LSTM) by Hochreiter et Schmidhuber (1997). LSTM is special kind of RNN, capable of learning long-term dependencies. In standard RNNs, the main unit is composed by a simple structure, such as a single *tanh* layer. However, LSTM introduces a structure based on four gates, which are used to remember important information, to forget irrelevant information and to select which type of information is used in each iteration during the learning. This is illustrated in Figure 2.6b. This mechanism allows the gradient to backpropagate more easily essentially by smoothing out the update of the hidden vector $h$ at each timestep by using activation functions. Finally, another way to solve the problems of a standard RNN is to use a Gated Recurrent Unit (GRU) (Chung *et al.*, 2014). The GRU is a simplified version of LSTM with only two gates: update gate and reset gate. The update gate is responsible for determining the amount of previous information that needs to pass along the next state. The reset gate is used from the model to decide how much of the past information is needed to neglect; in short, it decides whether the previous cell state is important or not. Finally, these two gates decide what information should be passed to the output. In comparison to LSTM, GRU have lesser training parameters and therefore uses smaller memory and executes faster. However, LSTM can provide better accuracy in larger datasets. Therefore, the choice of RNN should depend on the task and memory constraints.

While extremely effective, the LSTM or GRU networks are not suited for sequence of images or videos as they only process vectors. Therefore, to handle such data, one must first generate a meaningful vectorial representation and then apply the classical LSTM architecture. This representation has the major drawback of not using the spatial information for temporal processing. In order to correct this issue, Convolutional Long Short-Term Memory (ConvLSTM) networks were introduced by (Shi *et al.*, 2015). This architecture uses the convolutions in both the input-to-state and state-to-state transitions and adds peephole connections that allow the network to look at the cell state to make decisions. Due to both convolutions and recurrent neural network in its design, it models well the spatiotemporal relationships in a video. Similarly, ConvGRU (Ballas *et al.*, 2016) also have a similar functioning.

To conclude, RNNs facilitate the modeling of long-term dependencies through their potentially infinite memory. Comparatively, CNNs are limited to a fixed receptive field, which is cumbersome in case of sequences. However, the main drawback of the RNNs formulation is its limitation for parallel computations. As the previous state needs to be computed to output the next one, it becomes mandatory to run the calculations sequentially. Yet, despite such limitation, RNNs have been used with great success for natural language processing (Lipton *et al.*, 2015; Yonghui *et al.*, 2016), speech recognition

(Graves *et al.*, 2013; Sak *et al.*, 2015), video understanding tasks (Ballas *et al.*, 2016; Dwibedi *et al.*, 2018; Kim *et al.*, 2018a; Li *et al.*, 2018), *etc.*

### 2.2.4 Autoencoder

The Autoencoder (AE) is a type of feed-forward networks which possesses the auto-association property (Hinton et Zemel, 1993; Wang *et al.*, 2016b). It is an unsupervised learning algorithm and provides an alternative to dimensionality reduction techniques like principal component analysis. It projects the data from a higher dimension to a lower dimension using linear transformations and tries to preserve the important features of the data while removing the non-essential parts. The main goal of this type of networks is to learn how to reconstruct the data from a lower dimensional space representation. Figure 2.7 depicts the classical architecture for AE. It tries to learn an approximation of the identity function, to output $\hat{x}$ that is similar to $x$. The identity function seems a trivial solution for this, but it can be avoided by placing constraints on the network, such as by limiting the number of hidden units.

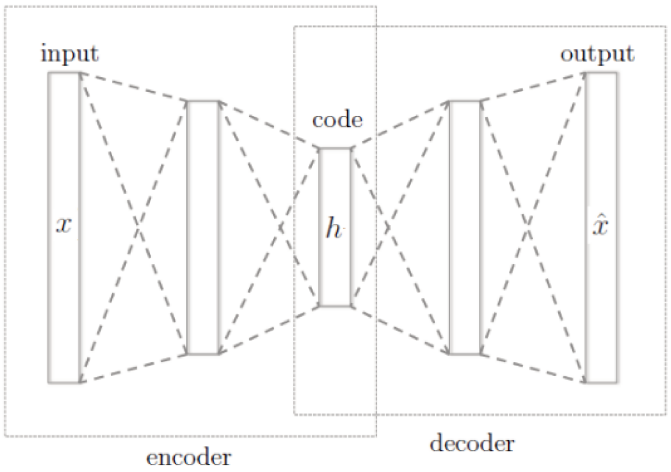



Figure 2.7: Classic autoencoder architecture. Source: (Wikipedia contributors, 2022)

AE can be divided into three parts: encoder, code and decoder. The encoder compresses or downsamples the input into a lower dimensional representation. The space represented by this new dimensionality is often called the latent-space or bottleneck and contains the semantic representation or the code of the input. The decoder intends to reconstruct the input using only the encoding of the input. AEs have generalized the idea of encoder and decoder beyond deterministic functions to stochastic mappings $f_{encoder}\,(h|x)$ and $g_{decoder}\,(\hat{x}|h)$ . The goal is to minimize $\arg\min_{f,g} \|x - (f\ o\ g)(x)\|^2$. The traditional AE and PCA are not suitable for images or video frames as they ignore the spatial structure and location of pixels, which is termed as being permutation invariant. Furthermore, it is important to note that when working with images, even with a small size like 100 $\times$ 100, these methods introduce large redundancy in network parameters due to dense connections. Therefore, for spatial data of images, we need a new form of autoencoder, which is presented below.

#### 2.2.4.1 Convolutional autoencoder

The CNN successfully understands spatial characteristics of images, with many times less parameters than the fully connected networks due to convolution and pooling operations. Inspired from this, these operations are used in the autoencoder to obtain a convolutional autoencoder (CAE) (Masci *et al.*, 2011). CAEs have fewer parameters on account of their kernels being shared across many input locations/pixels. Like CNN, they use convolutional and pooling layers, or strided convolutional layers, for encoding the input image. Similarly, for decoding, the deconvolutional layer or unpooling layer, or both are used. Finally, the mean squared error loss between input and reconstructed images are back-propagated for learning. We present below the deconvolutional and unpooling layers.

**Deconvolutional layer and unpooling**

A deconvolutional layer in a neural network is a layer which is able to obtain a dense map from downsampled and course input (Dumoulin et Visin, 2016). A more appropriate name is the transposed convolutional layer, as the term deconvolution is misleading since deconvolutional layers also perform convolutions. A transposed convolution has a transformation going in the opposite direction of a normal convolution, *i.e.,* from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution. Pooling layers in convolutional networks are required in order to decrease the number of network parameters. Unpooling layers perform the reverse of pooling layers (see Figure 2.8).

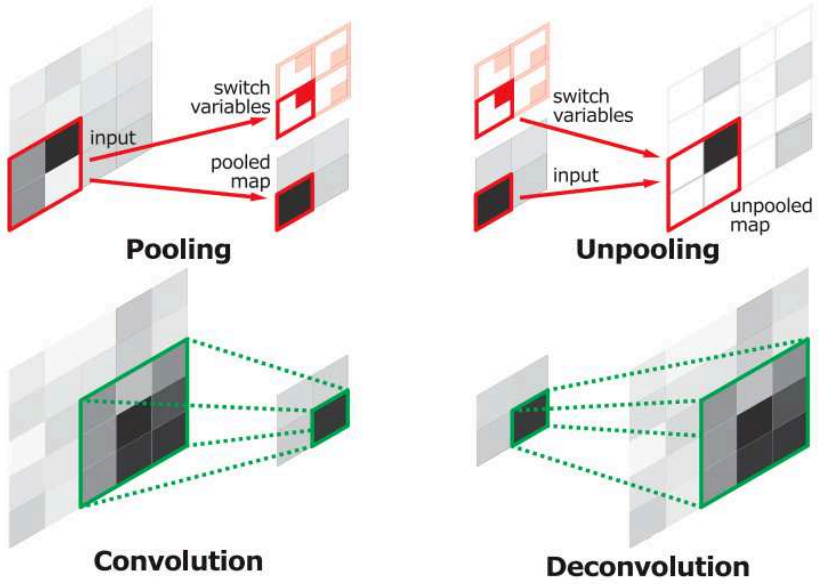



Figure 2.8: Schematic representation of pooling, unpooling, convolution and deconvolution. Source: (Noh *et al.*, 2015).

The location of the maximum activation in the pooling layer is recorded in switch variables (Noh *et al.*, 2015), in the unpooling layer it is placed back. The output of such an unpooling layer is sparse, as it is an enlarged version of the input map. The deconvolutional layer can produce a dense output map from the unpooling layer output. In summary, convolutional layers map multiple activations in a receptive field to a single activation, deconvolutional layers map one single activation to a field or window of multiple

activations. Convolutional layers learn a filter which map a receptive field to one value, consequently deconvolutional layers learn filters that perform the opposite.

Concerning video data, CAE can be used in each image of the sequence but it will not take into account the temporal dimension. Therefore, Convolutional LSTM based autoencoder (CLSTM-AE) was proposed (Chong et Tay, 2017). It uses first few convolutional layers for encoding, then later in deeper layers, it uses the ConvLSTM layers. Similarly for deconding, it first uses the ConvLSTM layers, followed by deconvolutional layers to get the final video reconstruction. In conclusion, autoencoder is an easy to use, unsupervised learning algorithm, and can be applied to vector, images, videos, *etc.*

### 2.2.5 Generative adversarial network

A Generative adversarial network (GAN) is a category of generative models, first proposed by Goodfellow *et al.* (2014), that consists of two sub-networks in competition, a generator and a discriminator network, as shown in Figure 2.9. During the learning phase, the generator try to generate convincing data to fool the discriminator who in turn tries to detect whether the data is real or fake. In this way we obtain two trained networks, one to generate realistic data and the other to distinguish between real data and generated data.

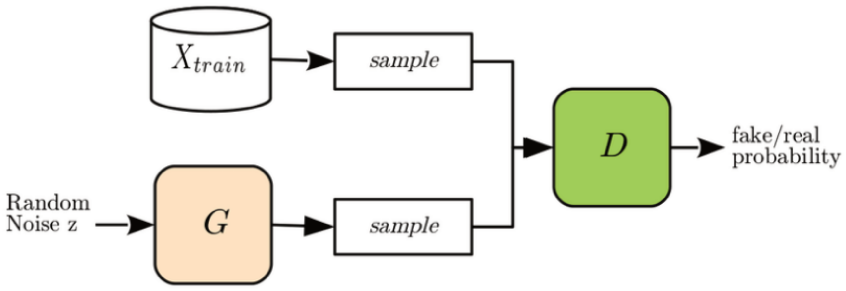



Figure 2.9: Illustrating Generative adversarial network (GAN). Source: (Hayes *et al.*, 2017)

The generator network is often an autoencoder which generates some input for the discriminator, given a noisy input. The discriminator is usually a fully connected network that takes the real and generated input and predicts whether the generated input is fake or not. Like last subsection, for image data, convolutional autoencoders are use as generators and CNNs are used as discriminators. Similarly, for sequences of images, ConvLSTM or ConvGRU can be a part of the GAN architecture. Finally, the ability of GANs to generate meaningful representations has gained a major interest in many applications like object detection (Li *et al.*, 2017), data generation (Ehsani *et al.*, 2018), super resolution (Ledig *et al.*, 2017), *etc.*

## 2.3 Spatio-temporal feature learning from videos

In this section, we present different methods for learning spatio-temporal features from a video stream. Since videos have both spatial and temporal dimensions, we explore in this section, various ways to extract this spatio-temporal information. The methods presented below are supervised as they were the first to be developed for spatio-temporal feature learning and furthermore, they act as a baseline for unsupervised approaches. As

a reminder, in supervised learning we have as input the data with its label for training the neural network. We present below various approaches, based on how they treat the spatial and temporal dimensions of video.

### 2.3.1 Modeling spatial and temporal dimensions independently, processing them sequentially

One of the first works to obtain features from video was to use image-based 2D CNN extractors on each frame of the video and then pooling their predictions across the whole video (Karpathy *et al.*, 2014). It is very convenient to use, but it ignores the temporal structure of the video, *e.g.,* these models potentially cannot distinguish between opening and closing of a gate. Therefore, it is essential to model temporal data in the video stream.

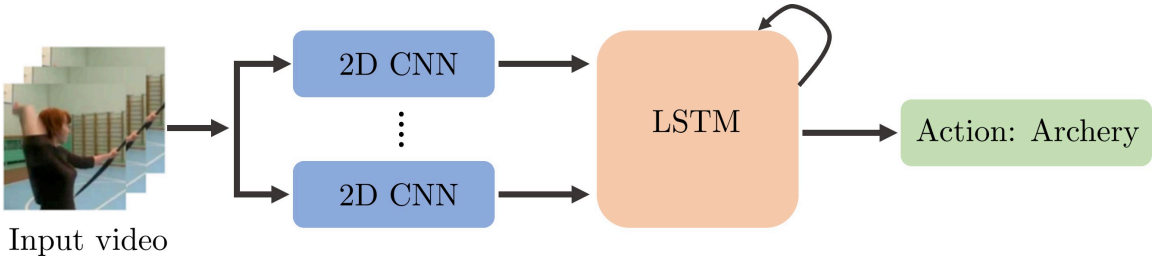



Figure 2.10: Illustrative example of 2D CNN for spatial feature extraction and an LSTM to model this spatial information over time, applied to action recognition task.

One way to continue using image-based extractors without compromising temporal information is to first extract spatial feature vector by using a 2D CNN shared over all frames and then modeling the temporal representation of this sequence of spatial features using a recurrent neural network (Donahue *et al.*, 2015; Yue-Hei Ng *et al.*, 2015). Since traditional vanilla RNN suffers from the vanishing gradient problem, most of the existing methods adopt gated RNN architectures like LSTM, to model the long-term temporal dynamics in video sequences (Du *et al.*, 2017; Sun *et al.*, 2017; Perrett et Damen, 2019). Figure 2.10 demonstrates this approach, where first the spatial information is extracted from each video frame using a 2D CNN, then this sequence of spatial information is fed into an LSTM network which temporally relates it, and finally the overall information is used for the action recognition task. Some other notable works are described as follows. Wu *et al.* (2019a) leveraged two LSTMs operating on coarse-scale and fine-scale CNN features cooperatively. Majd et Safabakhsh (2020) proposed the $C^2$LSTM model which incorporates convolution and cross-correlation operators to learn motion and spatial features while modeling temporal dependencies. To learn the temporal information from both the past and future, some works adopt the Bi-directional LSTM consisting of two independent LSTMs, called the forward and backward LSTM respectively (Ullah *et al.*, 2017; Zhao et Jin, 2020). The forward LSTM is used with past frame sequence like a single LSTM network, while the backward LSTM takes a future frame sequence as input. Finally, the information from both LSTMs are fused together to perform the target task (like action recognition).

This approach of modeling spatio-temporal data also benefited from the introduction of attention mechanisms, whether it is spatial attention (Sharma *et al.*, 2016; Ge *et al.*, 2019; Sudhakaran *et al.*, 2019), temporal attention (Meng *et al.*, 2019; Wu *et al.*, 2019b), or spatio-temporal attention (Li *et al.*, 2018; Liu *et al.*, 2020). Sharma *et al.*

(2016) designed a multi-layer LSTM model, which recursively outputs attention maps weighting the input features of the next frame to focus on the important spatial features. Sudhakaran *et al.* (2019) introduced a recurrent unit with built-in spatial attention to spatially localize the discriminative information across a video sequence. Li *et al.* (2018) proposed a Video-LSTM incorporating convolutions and motion-based attention into the soft-attention LSTM (Xu *et al.*, 2015b), to better capture both spatial and motion information. Compared to LSTM, Gated Recurrent Unit (GRU) has fewer gates, leading to fewer model parameters. Therefore, many works use GRU instead, for modeling spatio-temporal data in videos (Ballas *et al.*, 2016; Dwibedi *et al.*, 2018; Kim *et al.*, 2018a).

Some drawbacks of the presented approach to model spatio-temporal are as follows. Since this approach is sequential, if the 2D CNN fails or performs poorly, the complete approach will fail. Networks like LSTM are often costly in terms of memory and time. Finally, this approach has difficulties to model fine-grained action since local motion is generally hard to model with it. This is because the recurrent network models sequence of spatial information extracted for the whole frames, thus it captures the propagation of global frame level spatial information over time, thus not paying special attention to local motion.

### 2.3.2 Two-stream modeling

Different from the last approach, in this approach two streams, *i.e.,* appearance (spatial) and motion (temporal) are used independent of each other and later fused to obtain the overall spatio-temporal features. It usually works better than the last approach since it does not process space and time dimensions sequentially. Simonyan et Zisserman (2014) first introduced the two-stream network consisting of two parallel networks, *i.e.,* spatial and temporal network. The spatial network accepts raw video frames while the temporal network gets multi-frame-based optical flows as input. The final score is obtained by fusing scores from both streams. Optical flow provides an effective motion representation in the scene and can effectively remove the non-moving background information (Horn et Schunck, 1981). The proposed network had very high performance on existing benchmarks, while being very efficient to train and test. Figure 2.11 shows the two-stream concept of Simonyan et Zisserman (2014).

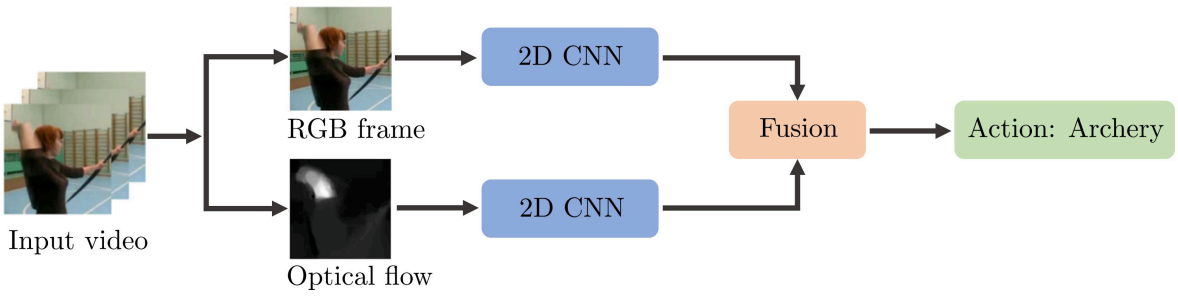



Figure 2.11: Illustration of Two-stream model for video action recognition proposed by Simonyan et Zisserman (2014).

In recent years, the two-stream modeling has seen big improvements and is being used massively in many video tasks. Regarding neural networks, deeper networks have been introduced for both streams and they improve the performance (Wang *et al.*, 2015b, 2016a). But deeper networks tend to overfit on the video dataset (especially smaller

datasets) (Wang *et al.*, 2015a) and therefore Wang *et al.* (2015b) introduced a series of good practices, including cross-modality initialization, synchronized batch normalization, corner cropping, data augmentation, large dropout ratio, *etc.* to prevent it. Since there are two streams, there is a stage called spatio-temporal fusion, where results from both the networks are merged to obtain the final score. The standard way is late fusion, which is a weighted average of final scores from both streams (Simonyan et Zisserman, 2014; Wang *et al.*, 2015b). Many works claim that late fusion is not the optimal solution (Feichtenhofer *et al.*, 2016, 2017). For example, Feichtenhofer *et al.* (2016) shows that a fusion at an intermediate layer not only improves the performance but also reduces the number of parameters significantly as compared to original work of Simonyan et Zisserman (2014). As video is a temporal sequence of frames, researchers have also explored various recurrent neural networks with two-stream networks like LSTM (Sun *et al.*, 2017; Li *et al.*, 2018), bi-directional LSTM (Ullah *et al.*, 2017), hierarchical multi-granularity LSTM (Li *et al.*, 2016), *etc.* Li *et al.* (2018) proposed VideoLSTM, a two-stream network with an LSTM-based spatial attention mechanism and a lightweight motion-based attention mechanism. It improves results on action recognition task and furthermore, the learned attention can be used for action localization as well. Other relevant works are Lattice-LSTM (Sun *et al.*, 2017), ShuttleNet (Shi *et al.*, 2017), FASTER (Zhu *et al.*, 2020), *etc.* Two-stream networks still cannot capture long-range temporal information. To address this issue, Wang *et al.* (2016a) proposed a temporal segment network which first divides a video into uniformly distributed segments, then randomly selects one video frame within each segment and feeds them to the network which shares weights for input frames from all the segments. In the end, a segmental consensus is performed with one of the operators like average pooling, max pooling, weighted average *etc.*, to aggregate information from the sampled video frames. Since this network uses the whole video as input and provides video-level prediction, it models long-range temporal structure. Other relevant works are DVOF (Lan *et al.*, 2017), TLE (Diba *et al.*, 2017b), TRN (Zhou *et al.*, 2018), TSM (Lin *et al.*, 2019), *etc.*

To conclude, the two-stream approach is better than the last approach since it takes care of both spatial and temporal dimensions of the video stream. This approach does not treat the spatial and temporal dimensions jointly but instead relies on an external component to extract the motion information. If the motion information is correctly extracted, the spatial and temporal networks can learn the features of video, which are later fused together to make the final decision. This approach has seen a big evolution since its beginning and is still being largely used in many video tasks. If the joint spatio-temporal information is not very important for an application and one can rely on external detectors for motion information, then this approach provides a good alternative.

### 2.3.3 Jointly modeling spatio-temporal dimensions

A conceptually simple way to understand a video is to consider it as a 3D tensor with two spatial and one temporal dimension. One of the simplest ways to jointly model spatial and temporal dimensions is to extend 2D convolutions into three dimensions. It is called a 3D convolution and it is an extension of 2D convolution which consists of learning space-time kernel filters instead of space kernel filters only. Figure 2.12 demonstrates the difference between 2D and 3D convolutions.

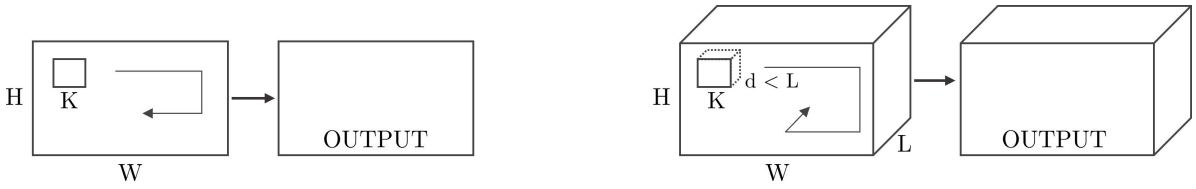



Figure 2.12: Comparison of 2D convolution (left image) and 3D convolution (right image) operations. Figure reproduced from (Tran *et al.*, 2015).

A 3D CNN is basically a network like 2D CNN where 2D convolution and pooling operations are replaced by 3D convolution and pooling operations. The seminal work for using 3D CNNs is of Baccouche *et al.* (2011). While inspiring, the network was not deep enough to show its potential. Tran *et al.* (2015) extended it to a deeper 3D network, named C3D. C3D follows the modular design and learns the spatio-temporal features from raw videos in an end-to-end learning framework. It showed strong generalization capability and encouraging performance on video action recognition task but was computationally more expensive than its 2D counterpart. Figure 2.13 demonstrates the schema for video action recognition task using 3D CNN. We can observe that the 3D CNN takes a video volume as input and jointly understand the spatio-temporal features (like 2D CNN do for images) to predict the output. Important characteristics of 3D CNNs are that they directly create hierarchical representations of spatio-temporal data and are very powerful in modeling discriminative features. However, the main issue is that they have many more parameters than 2D CNNs because of the additional kernel dimension, and this makes them harder and longer to train.

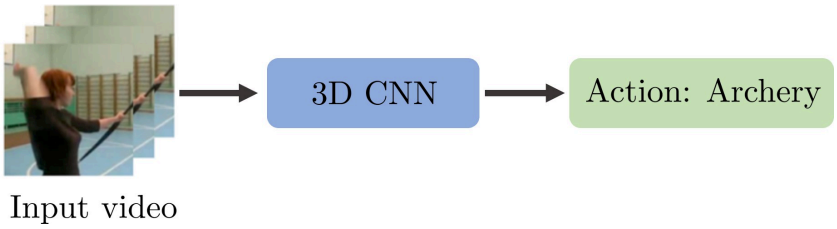



Figure 2.13: Illustration of 3D CNN for video action recognition.

Since 3D CNN needs high memory space and running time (Tran *et al.*, 2015; Qiu *et al.*, 2017), some works propose to factorize 3D convolutions (Xie *et al.*, 2017a; Tran *et al.*, 2018). Specifically, they decompose 3D convolution kernels (*e.g.*, $3 \times 3 \times 3$) into 2D spatial kernels (*e.g.*, $1 \times 3 \times 3$) followed by a 1D temporal kernels (*e.g.*, $3 \times 1 \times 1$), called as (2+1)D kernel. This allows to reduce the number of trainable parameters significantly while still modeling space-time features. Another alternative to reduce the complexity of 3D CNN is to use 2D convolutions in place of some 3D convolutions in the same network. A temporal shift module is proposed by Lin *et al.* (2019), which shifts a part of the channels along the temporal dimension to perform temporal interaction between the features from adjacent frames. Different from these parameter-free temporal shift operations, Sudhakaran *et al.* (2020) introduced a lightweight Gate-Shift Module (GSM), which uses learnable spatial gating blocks for spatial-temporal decomposition of 3D convolutions. S3D (Xie *et al.*, 2018) combines the merits of approaches mentioned above. It replaces the 3D convolutions at the bottom of the network with 2D kernels and factorizes the remaining 3D kernels into (2+1)D kernels. It demonstrates that this kind

of top-heavy network has higher recognition accuracy, along with reduction in model size and training complexity.

The 3D CNN based methods generally perform spatio-temporal processing over fixed time intervals via the window-based 3D convolutional operations, where each convolution operation only attends to relatively short-term context in videos (Baccouche *et al.*, 2011; Ji *et al.*, 2012; Tran *et al.*, 2015). Thus, by design and due to high computational requirements, they are not suitable for long-range spatio-temporal relations in videos. Hence, several approaches focused on modeling these long-term dependencies. Diba *et al.* (2017a) proposed temporal 3D convolutional networks, where the temporal transition layer models variable temporal convolution kernel depths. It can efficiently capture the appearance and temporal information at short, middle, and long terms. They reinforced this in their subsequent work (Diba *et al.*, 2018), with the introduction of a new block with residual connections, which can model the inter-channel correlations of a 3D CNN with respect to the temporal and spatial features. A long-term temporal convolution framework was proposed by Varol *et al.* (2017) to model the long-term temporal information in videos. They increase the temporal extents of 3D convolutional layers at the cost of reducing the spatial resolution. Finally, Hussein *et al.* (2019) proposed Timeception, a multi-scale temporal-only convolutional network to account for large variations and tolerate a variety of temporal extents in complex and long actions.

This approach is a good alternative to the last two approaches. The most important quality of this approach is that it learns jointly the spatio-temporal features of the video. This is an essential point since it depends only the video in hand to learn its feature and does not require external supervision. Even though this approach has seen significant improvement throughout the years, it still remains computationally expensive and cannot treat a large video sequence at once. As a concluding remark, this approach should be used when we want to model the spatio-temporal dimension correctly in the video and when the long range temporal dependencies are not important.

### 2.3.4 Hybrid approaches

To further enhance the spatio-temporal comprehension of videos, several works have investigated using 3D CNN based models with two-stream or multi-stream designs (Carreira et Zisserman, 2017; Wang *et al.*, 2017; Feichtenhofer *et al.*, 2019; Li *et al.*, 2020). Carreira et Zisserman (2017) introduced the two-stream inflated 3D CNN (I3D), made by inflating the convolutional and pooling kernels of a 2D CNN with an additional temporal dimension. Similarly, Wang *et al.* (2017) integrated a two-stream 3D CNN with an LSTM model to capture the long-range temporal dependencies. SlowFast network (Feichtenhofer *et al.*, 2019) extend the idea of two-stream approaches but without the need of optical flow as input. This 3D CNN based two-stream network have a slow pathway and a fast pathway that operate on frames at low and high frame rates to capture semantic and motion, respectively. At each layer of this network, the features of the fast and slow pathways are fused (by summation or concatenation) to share the motion and semantic information among sub networks. Finally, Li *et al.* (2020) introduced a two-stream spatio-temporal deformable 3D CNN with attention mechanisms to capture the long-range temporal and long-distance spatial dependencies. Recently, a new neural network, called the Transformer was introduced (Vaswani *et al.*, 2017). It mainly utilizes the self-attention mechanism (Bahdanau *et al.*, 2015) to extract intrinsic features and shows great potential for extensive use in AI applications. It was first applied to NLP

for machine translation tasks (Vaswani *et al.*, 2017; Kenton et Toutanova, 2019). Only recently, it has been applied in vision tasks (Dosovitskiy *et al.*, 2020; Chen *et al.*, 2020), but it has shown a very strong potential with performance equal or better than CNN based approaches for various tasks (Zeng *et al.*, 2020; Khan *et al.*, 2022). Transformers also have various drawbacks like they require very massive datasets for large-scale training, extremely heavy computational requirements, and furthermore generalization and robustness of transformers is still an open issue (Han *et al.*, 2022).

In this section, we studied different types of approaches to learn spatio-temporal features from video. We observed that two-stream, 3D CNN and hybrid approaches all rest viable for effective learning in video. While two-stream approaches require an external detector for motion and does not treat the spatial and temporal dimensions jointly, the 3D CNN based approach requires high computational power. Finally, the hybrid approaches takes best of the both worlds but their drawbacks does accompany these approaches, like they remain computationally expensive. Finally, the choice of approach depends on the task at hand and requirements. In this thesis, we want to learn spatio-temporal features jointly in video and furthemore we want to use minimum supervision possible. For the applications, *i.e.,* video anomaly detection and perimeter intrusion detection, we do not require long spatio-temporal information. For example, an intrusion that was detected 10 minutes ago, does not have any direct relation in detecting the current intrusion. Therefore, we choose the 3D CNN based approach in this thesis. The next section will present how can we use these approaches in an unsupervised context.

## 2.4 Unsupervised spatio-temporal video understanding

This section presents various approaches for unsupervised learning of spatio-temporal features from the video stream. To recall, in unsupervised learning, we do not have labels with data during training the neural network. In the last section, we explored various approaches to obtain spatio-temporal features. All the approaches, whether 2D CNN with a recurrent network, two-stream modeling or 3D CNN, can be used in the context of unsupervised learning. We just need to employ them either in reconstructive, predictive, or generative models, that are presented below.

### 2.4.1 Reconstruction models

The main idea here is to reconstruct a given input, in order to learn pertinent features of the input data. The input data can be a vector, image or a video sequence. In our case, it is the video sequence. These models include methods such as auto encoders (refer Section 2.2.4) or sparse coding, that are used to extract different linear and non-linear representations of appearance (image) or motion (stream) or both, to model meaningful patterns of unlabeled videos. We can simply extend the idea of CAE for images to videos and use many of the video feature extraction approaches described in last section. Hasan *et al.* (2016) proposed a spatio-temporal stacked autoencoder with a video clip of ten frames as input. It consists of a stack of 2D convolutions and 2D deconvolutions for encoding and decoding, respectively. Similarly, Chong et Tay (2017) introduced a convolutional LSTM based autoencoder, with the same input of ten frames. They used 2D convolutions with LSTM for encoding and LSTM with 2D deconvolutions for decoding. By using LSTM, they were able to better understand the temporal characteristics of video stream. Similarly, 3D convolutional autoencoder (Zhao *et al.*, 2017) encodes and decodes

using 3D convolutions and 3D deconvolutions respectively. In theory, any approach of learning spatio-temporal features from videos described in Section 2.3 can be used with an autoencoder. More works on unsupervised deep learning on videos will be detailed in upcoming sections for video anomaly detection and perimeter intrusion detection tasks.

### 2.4.2 Predictive models

Another approach based on unsupervised deep learning tends to use predictive modeling. It is well-known in time series analysis under auto-regressive models. Different from reconstructive models where the objective is to train a model to reconstruct the input data, predictive models try to predict a current sequence of input using the previous sequences. Concerning video data, the objective is to model output of current frame (or future frame) as a function of past $t$ frames. Since it concerns temporal sequence, traditionally recurrent neural networks like LSTM were used (Srivastava *et al.*, 2015; Luo *et al.*, 2017b). Recently, there have also been attempts to perform efficient video prediction using convolutional autoencoder networks. The function of an AE can be determined by considering its output values. When the output values are only the reconstruction of the inputs, the AE is a reconstructive model. When the output values are the values after the input values in the time axis, the model is said to be predictive. Medel et Savakis (2016) introduced a ConvLSTM-based AE where the encoder extracts representations from an input sequence, a first decoder that uses these representations to reconstruct the input sequence, and a second decoder that uses the same representations to predict the next frame. Similarly, Zhao *et al.* (2017) proposed a network made up of an encoder and two decoders, the first for reconstruction and the second for prediction. In this network, 3D convolution layers are used instead of ConvLSTMs, for the learning of spatiotemporal representations. More works will be discussed with VAD as the application in later sections.

### 2.4.3 Generative models

Generative models like Generative Adversarial Networks (GAN), Adversarially trained AutoEncoders (AAE) and Variational Autoencoders (VAE) are used for the purpose of modeling the likelihood of video samples in unlabeled data. These models are also used massively in unsupervised video learning tasks like video anomaly detection (Ravanbakhsh *et al.*, 2017; Kiran *et al.*, 2018). For example, Lee *et al.* (2018) proposed a generative adversarial network, called STAN (spatio-temporal adversarial networks) for this VAD task. They use a video clip as input. The middle frame of the video clip is removed and the resultant clip is fed to a spatio-temporal generator. The generator is a convolutional autoencoder with convolutions, ConvLSTMs and deconvolutions. It generates the missing middle frame, which is added to its position in the video clip. The spatio-temporal discriminator, made using 3D convolution layers, takes the generated video clip and original video clip as input and tries to distinguish the clips. Once the two networks are trained, the detection of abnormal events is done using the losses from both generator and discriminator. More unsupervised generative works are detailed in later sections.

In this section, we presented various ways to unsupervisely learn spatio-temporal features of video. We can observe that all of these approaches are different and none of them is more suitable than others. Infact, they can also be combined together if required, and can potentially lead to better generalization. All of these approaches have been extensively developed in the last few years. Choosing the correct approach, depends on

the video task being tackled, its requirements, and the spatio-temporal feature learner (like a 2D CNN, 3DCNN, *etc.*) being used. Our chosen 3D CNN based approach translates to 3D CAE in unsupervised learning. This 3DCAE can be used as a reconstructive model, predictive model and even as the generator of a generative model. In this thesis, we explore the first two ways and even combine them. The generative model requires extra computation on top of the generator via discriminator, therefore we avoided that approach to be computationally light.

## 2.5 Video anomaly detection (VAD) methods

Given a video stream, this task deals with the detection of anomalies. The anomalies are context dependent and vary from one dataset to another as described in Section 1.4.1. Since anomaly is a rare event and the datasets are usually not annotated. Therefore, most methods use unsupervised deep learning, with or without external supervision from pre-trained detectors. We focus only on these methods. It must be noted that there are other methods which rearrange the current VAD datasets to have weak labels and thus use weakly supervised learning. Some methods use semi-supervised learning by using a portion of labeled data for training. Finally, there are also a few methods that reorganize VAD datasets to use supervised learning. We do not review these kind of methods as they are well out of the scope of this study.

To systematically review the VAD methods, we first explore several types of input that they can take, followed by different proxy tasks that they can perform and finally some auxiliary components that they use to enhance their performance.

### 2.5.1 Types of input

To understand different approaches for video anomaly detection, we must first consider the different types of input they use. Even though the task is detecting anomalies in video, the methods can use frames of video, frame patches, objects in frame, objects in video snippet, *etc.* as input. We explain below different input types used for VAD.

1. Full Frame (FF): This input means using a single frame as input. This means that the video is read frame by frame and each frame is an input to the VAD method. It can be used with any VAD task but has been predominantly used in the reconstruction task (Ravanbakhsh *et al.*, 2017; Nguyen et Meunier, 2019; Ye *et al.*, 2019). One major issue with using FF as input is that we miss to capture the temporal information of video sequence.

2. Frame patch (FP): This input is made by dividing the video frame into smaller patches. The video is read as frames which are converted into smaller patches and each patch is considered as a separate input. If an anomaly is detected in one of the frame patches, then the frame is labeled as anomalous (Sabokrou *et al.*, 2018). We have similar problem here as we lose the temporal dimension and due to patches, we loose part of the spatial information.

3. Frame object (FO): It refers to the objects in the video frame. A pre-trained object detector is first used to detect all possible objects of interest and then these objects are used as the input to the method (Ionescu *et al.*, 2019; Georgescu *et al.*, 2021b). Since the input is object, rather than the whole frame, we do not have background

noise. The problem is that it is highly biased to the external dataset where the object detector was trained, it assumes that objects of interest causing anomalies are known in advance and finally if the object detector fails, the VAD will fail too.

4. Video clip (VC): It refers to a video snippet with a fix number of consecutive frames. It is one of the most widely used input for VAD task (Hasan *et al.*, 2016; Lee *et al.*, 2018; Dong *et al.*, 2020; Chang *et al.*, 2020; Cho *et al.*, 2022). Since it is a video, we have both spatial and temporal dimensions to exploit. It has been used with all the proxy tasks for VAD. However, it takes more time and resources to process video clips and furthermore, it is comparatively difficult to extract pertinent information since often the anomalies occupy a small spatio-temporal volume in the video clip.

5. Video clip patch (VCP): It corresponds to 3-dimensional patches extracted from a video clip. Unlike frame patch, here we include the temporal dimension into the input. Various methods propose different patch extraction techniques (Tran et Hogg, 2017; Park *et al.*, 2022). Since not the whole video clip is used for creating these 3D patches, it works faster than video clip as input. These patches are generally created on parts of the video clip where foreground objects occur (assuming anomaly is in foreground), therefore they either use moving object detector as a pre-processing, or they use some heuristics to create patches only in certain parts of the video clip, where foreground objects should be present (like Park *et al.* (2022) exclude a margin of 12.5 percent from the top and bottom in each frame of the video clip).

6. Video clip object (VCO): It refers to a video clip composed of objects detected by an object detector. It is an extension of frame object to video clip. It is being used by many recent methods as it is robust to background noise and have both spatio-temporal dimension like a video clip (Yu *et al.*, 2020; Ouyang et Sanchez, 2021; Liu *et al.*, 2021b; Georgescu *et al.*, 2021a). The problem with this input is same as that of frame object due to its dependence on pre-trained object detector.

### 2.5.2 Types of tasks

Since anomaly is a rare event, there is lack of anomaly examples for training a supervised two-class classifier. Due to this, most works address the VAD task using a proxy task like frame reconstruction, prediction, *etc.* These tasks are called proxy tasks since they address the main task (VAD is binary classification task) indirectly. We describe below the classification of approaches based on the proxy task that they use.

#### 2.5.2.1 Reconstruction task

This task concerns the reconstruction of an input entity like frame, video clip, object, *etc.* The aim is to learn normality by trying to reconstruct entities in normal videos (without any anomaly). Since the representations were learned from normal videos, it is assumed that anomalies will be harder to reconstruct and thus they can be separated from correctly reconstructed normal entities. This task is usually performed using a convolutional autoencoder (Masci *et al.*, 2011) or an adversarial network like a GAN (Goodfellow *et al.*, 2014). Autoencoder (AE) encodes the input data into the latent space through an encoder and then reconstructs it using a decoder. The anomaly measure is the reconstruction error, which is assumed to be high when anomalies occur since the training of AE is only done on normal videos.

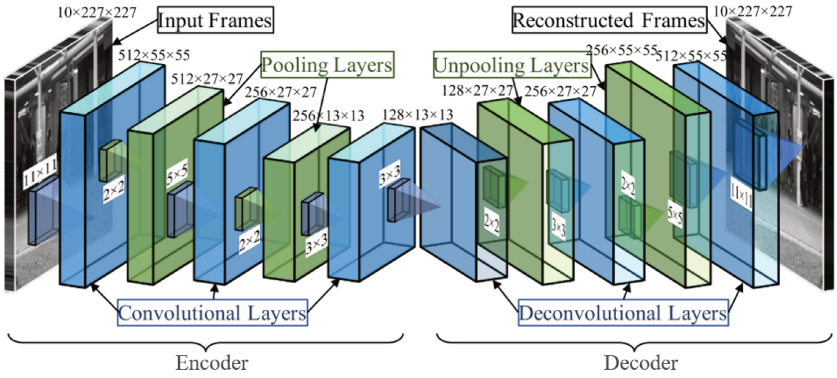



Figure 2.14: The architecture of the convolutional autoencoder proposed in Hasan *et al.* (2016) for video anomaly detection.

Figure 2.14 presents an example of convolutional autoencoder proposed by Hasan *et al.* (2016), using reconstruction task for video anomaly detection. They use a video clip of 10 frames as input and reconstructs it. They use 2D convolutions and pooling layers for encoding and 2D deconvolutions and unpooling layers, for decoding. The mean squared error (MSE) between input clip and its reconstruction is used as the reconstruction error (anomaly measure). By using only 2D convolutions, the temporal features of the video clip cannot be taken into account. To address this, Chong et Tay (2017) proposed to add convolutional long-short term memory (LSTM) layers in both encoder and decoder. Taking it one step further, Zhao *et al.* (2017) proposed a 3D convolutional autoencoder with 3D convolutions and 3D deconvolutions for VAD. Unlike 2d convolutions plus LSTM, the 3D convolutional autoencoder jointly captures the spatio-temporal features from normal videos. Recently, many works used some AE variant for VAD, using FF, VC or VCO as input (Ye *et al.*, 2019; Ouyang et Sanchez, 2021; Astrid *et al.*, 2021b; Cho *et al.*, 2022).

One of the earliest works for VAD using GANs, with reconstruction as proxy task, was proposed by Ravanbakhsh *et al.* (2017). Two conditional GANs are trained, with input as pairs of frames and noise vectors, which generate corresponding frames of a different modality (raw frames to optical flows and vice versa in the two GANs). The discriminators are asked to classify pairs of input and generated representations of frames as real or fake. Assuming that anomalies are not reconstructed well, they fuse reconstruction errors from both modalities as an anomaly score. Some works use both AEs and GANs for the reconstruction task (Nguyen et Meunier, 2019; Ye *et al.*, 2019). One such work is proposed by Nguyen et Meunier (2019), where they learn a correspondence between common object appearances and their associated motions in a two-stream model. Using an FF as input, they use a single encoder coupled with two decoders, one that predicts motion and another that reconstructs the input frame. This entire network is considered as a generator in a conditional GAN setup, where the discriminator is another network that distinguishes between pairs of input frames and corresponding real/estimated flow fields. For testing, they calculate loss scores at a patch-level. For other VAD works with reconstruction tasks, the reader can refer to (Kiran *et al.*, 2018; Ramachandra *et al.*, 2022). The main drawback of using this task is that sometimes the neural network generalize too well and even reconstruct the anomalies very well. In that case, the difference between normal and abnormal sample is neglible and thus anomaly cannot be detected.

#### 2.5.2.2 Prediction task

This task deals with the prediction of an unknown entity, given a known entity. For example, predicting a future frame, given a video clip with present and previous frames. Concerning VAD, the future frame prediction task is one of the most used tasks (Liu *et al.*, 2018; Lv *et al.*, 2021; Park *et al.*, 2020). Other prediction tasks for VAD include future object prediction (Liu *et al.*, 2021b) or missing bounding box prediction (Georgescu *et al.*, 2021a). The input entities used are VC, VCO and VCP, with VC being the most common (Lee *et al.*, 2018; Dong *et al.*, 2020). Models performing prediction task are also trained only on normal videos, with the assumption that they can precisely predict the entities in normal test sequences but will fail to correctly predict in anomalous test sequences. This requires comprehension of how normal spatio-temporal patterns propagate along the video clip. The anomaly score for this task can be calculated by measuring the difference between real and predicted entities or by calculating the conditional probability of a new observation based on the previous samples (Kiran *et al.*, 2018). Autoencoders, GANs and their combination are typically used for this task.

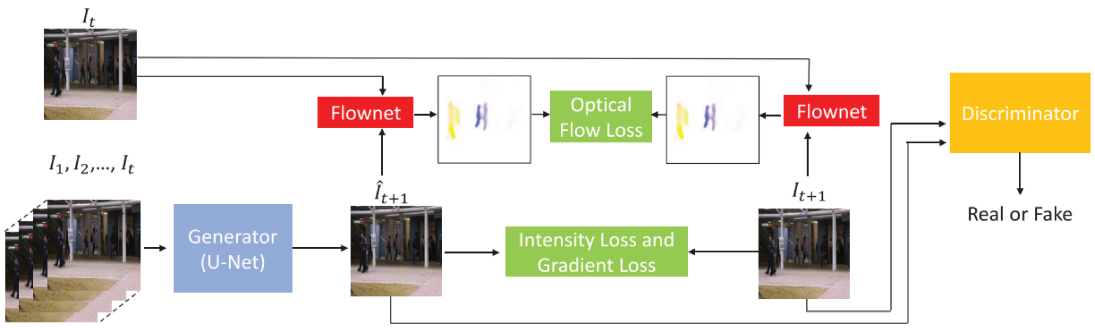



Figure 2.15: The future frame prediction architecture proposed by Liu *et al.* (2018) for video anomaly detection.

Figure 2.15 presents an example of future frame prediction architecture proposed by Liu *et al.* (2018). They propose a GAN, where the encoder is a U-Net style network (Ronneberger *et al.*, 2015) that takes training video clips of length $t$ as input and predicts the future frame $t + 1$. The discriminator network checks if the future frame is real (ground truth) or predicted, by minimizing the intensity and gradient loss. Furthermore, they add motion (temporal) constraint by using FlowNet (Dosovitskiy *et al.*, 2015) to estimate pairs of optical flow maps between the frame at $t$ and real or predicted frame at $t + 1$. The anomaly score is composed of L1 score between flow maps and intensity plus gradient scores from frame prediction. Some other prominent VAD works that use GANs for prediction task are (Lee *et al.*, 2018; Dong *et al.*, 2020), while works using AE for prediction tasks are (Park *et al.*, 2020; Lv *et al.*, 2021; Le et Kim, 2022; Park *et al.*, 2022). Since prediction task depends on the given input to learn the features in order to predict the unknown entity, it can fail when the input contains only stationary anomalies. For example, for the future frame prediction task, when the input clip contains an stationary car (anomaly), the model is likely to predict this car in future frame too. Thus, the prediction error would be too small to differentiate it from the normality.

#### 2.5.2.3 Self-supervised task

This type of task uses self-supervisory signals from the data itself and does not require external annotations. In other words, identify any hidden part of the data from any unhidden part of the data, *e.g.,* predict the missing patch of an image, given the image without patch. This learning paradigm is known as self-supervised learning. In recent years, the self-supervised learning is being used massively for different applications in various domains (Liu *et al.*, 2021a) like natural language processing (NLP), robotics, computer vision: applied to image and video analysis, *etc.* It is typically used as a pre-training step (pretext task) to enrich a learning module, which is later used for supervised downstream tasks like video classification, detection, *etc.* (Jing et Tian, 2020). Some well-known self-supervised tasks for video representation learning are video playback rate perception (Yao *et al.*, 2020), video pace or speed prediction (Benaim *et al.*, 2020; Wang *et al.*, 2020), relative speed perception (Chen *et al.*, 2021), video cloze procedure (Luo *et al.*, 2020), *etc.*

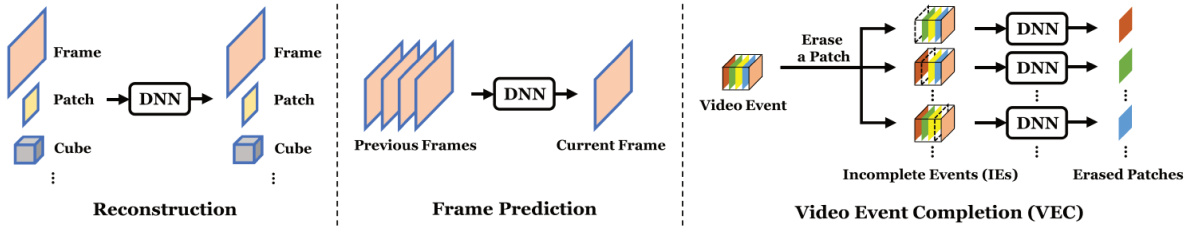



Figure 2.16: Comparison of VAD tasks: reconstruction (left), frame prediction (middle) and self-supervised: video event completion (right) (Yu *et al.*, 2020). DNN refers to a deep neural network and video event is the VCO input.

Concerning VAD, there are only two works that use self-supervised tasks. The first work is by Yu *et al.* (2020), where they propose a new self-supervised VAD task, called video event completion. This task is an adaptation of video cloze procedure task (Luo *et al.*, 2020). In this task, first VCO is extracted from a video clip using a pre-trained object detector and a series of appearance and motion-based operations. The VCO is referred as video event here and contains image patches with objects of interest like human. From each video event, one patch is erased randomly and a DNN (U-Net style autoencoder) predicts it using the rest of incomplete video event, known as appearance completion. The MSE between predicted and actual patch is used as an anomaly score. Similarly, motion completion is also done on the same video events: optical flow for each video patch is extracted using a pre-trained network and then similar pipeline like appearance completion is followed using the same U-Net style autoencoder. Again, MSE is used as an anomaly score here. The final anomaly score is a weighted sum of appearance and motion anomaly scores. Figure 2.16 demonstrates this self-supervised video event completion task along with a comparison with reconstruction and prediction tasks. The second work is proposed by Georgescu *et al.* (2021a), where they perform multiple self-supervised tasks using the same 3D CNN in a multi-task learning paradigm. They use VCO as input, using YOLOv3 object detector (Redmon et Farhadi, 2018). The proposed tasks are: (i) arrow of time prediction (discriminating forward and backward moving objects), (ii) motion irregularity detection (distinguishing objects captured in consecutive frames versus objects captured in intermittent frames), (iii) middle bounding box prediction (given objects in preceding and succeeding frames), (iv) model distillation: estimating normality-specific

class probabilities by distilling pre-trained classification (ImageNet (Russakovsky *et al.*, 2015)) and detection (MS COCO (Lin *et al.*, 2014)) teachers. Both these works show promising results but they rely heavily on external supervision using object detectors, which inhibits their generalizing capability as discussed in Section 2.5.1.

#### 2.5.2.4 Classification task

Despite the fact that video anomaly datasets are imbalanced and that only normal classes are present during training, there are still some methods that use a classifier to detect anomalies. Given only normal videos, they either perform one-class classification (Tran et Hogg, 2017; Sabokrou *et al.*, 2018) or do multi-class classification by adding dummy anomalies (Ionescu *et al.*, 2019; Georgescu *et al.*, 2021b).

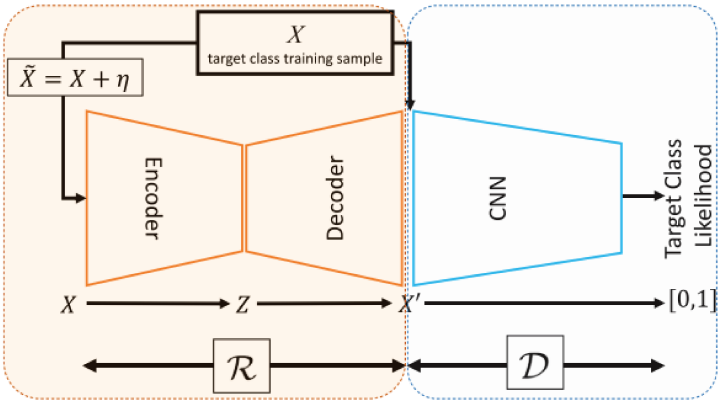



Figure 2.17: One-class classification VAD proposed by Sabokrou *et al.* (2018). $\mathcal{R}$ and $\mathcal{D}$ are generator and discriminator modules of the adversarially learned GAN. $X$, $\tilde{X}$, $Z$ and $X'$ refers to input, input with noise, latent space and reconstruction respectively, where target class 1 represents normal (non-anomaly) class.

Figure 2.17 presents an example of one-class classification approach for VAD. The input of this approach are patches from video frames, *i.e.,* FP rather than the whole frames. It adopts a GAN where the discriminator ($\mathcal{D}$) is tasked with distinguishing original image patches from reconstructions of noisy patches obtained from a denoising auto-encoder network ($\mathcal{R}$) which plays the role of generator. Since $\mathcal{R}$ is trained only on image patches from training data, it deteriorates reconstruction of outliers and thus enables D to distinguish an anomalous image patch from its reconstruction easily.

Concerning VAD with multi-class classification, Ionescu *et al.* (2019) proposed one of the first approaches. Their input is FO, which is obtained by using a pre-trained single-shot detector (SSD) (Lin *et al.*, 2017) on each frame of the video. They train convolutional auto-encoders on appearance and gradient features of these objects to learn latent representations and then perform k-means clustering followed by training of k one-class SVMs to make binary one-versus-rest classifications. Each cluster represents a different type of normality. At test time, they simply use the inverse of the maximum of k classification scores as an anomaly score, meaning if a sample does not belong to any cluster, it is considered as anomalous. The classification task fails when the anomaly resemble too much with normality in the latent space, thus making it difficult to separate anomalies from normality.

#### 2.5.2.5 Multiple task learning

Since no task is perfect for VAD, many approaches use multiple tasks to benefit from advantages of different tasks (Zhao *et al.*, 2017; Tang *et al.*, 2020; Liu *et al.*, 2021b).

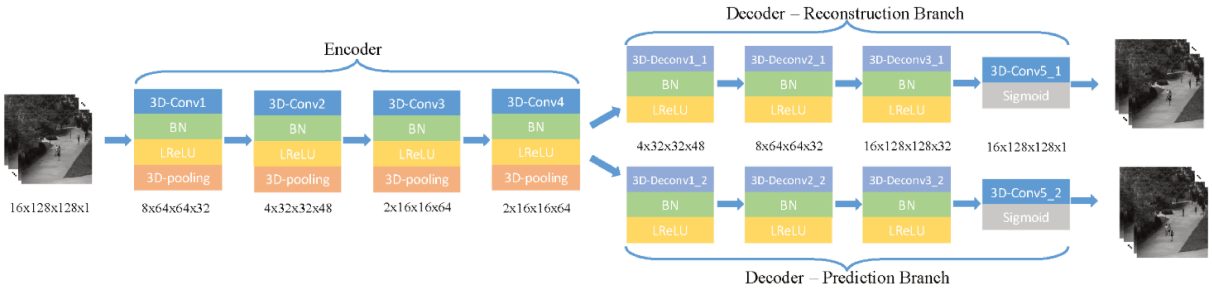



Figure 2.18: Network architecture proposed by Zhao *et al.* (2017) for reconstructing present frames and predicting future frames.

Figure 2.18 presents the work of Zhao *et al.* (2017) where they combine reconstruction and prediction tasks. The propose an encoder consisting of 3D convolutions (Baccouche *et al.*, 2011) and two decoders (one for each task) consisting of 3D deconvolutions (Zeiler *et al.*, 2010). They train on normal data to reconstruct the current video clip and predict the future frame video clip. During testing, they use only the reconstruction error of the test video clip as the anomaly measure. They demonstrated that using these tasks together boosts the detection of video anomalies. Even though multiple task learning benefits from advantages of different tasks, it is often difficult to combine various tasks and furthemore this makes the model computationally very expensive. So, the tasks must be chosen wisely with taking into consideration the gain in performance (if there is) in relation to the expense of memory and computation time.

### 2.5.3 Auxiliary components for enhancing VAD

To enhance the detection of anomalies, often many auxiliary components are used along with the VAD tasks. We describe below some of the most used components.

#### 2.5.3.1 Optical flow

Since moving objects can cause anomalies, it is important to reinforce motion analysis for VAD systems. Many motion analysis tasks employ optical flow as a fundamental basis upon which more semantic interpretation is built (Fortun *et al.*, 2015). Optical flow is defined as the estimation of a dense motion field, corresponding to the displacement of each pixel (Beauchemin et Barron, 1995). Concerning VAD, approaches use optical flow to enrich the input with motion information before feeding it to the deep neural network (Tran et Hogg, 2017; Zhao *et al.*, 2017; Ravanbakhsh *et al.*, 2017; Liu *et al.*, 2018; Nguyen et Meunier, 2019; Dong *et al.*, 2020; Yu *et al.*, 2020; Liu *et al.*, 2021b; Georgescu *et al.*, 2021b).

#### 2.5.3.2 Pre-trained feature extractor

Instead of learning features from anomaly datasets directly, many methods use a pre-trained feature extractor trained on an external dataset. Pre-trained extractors can be

used to extract features (of frames / video clips), to extract objects in the video (Ionescu *et al.*, 2019; Ouyang et Sanchez, 2021; Georgescu *et al.*, 2021a; Le et Kim, 2022), or to extract optical flow (Zhao *et al.*, 2017; Ravanbakhsh *et al.*, 2017; Liu *et al.*, 2018; Nguyen et Meunier, 2019; Dong *et al.*, 2020; Yu *et al.*, 2020; Liu *et al.*, 2021b; Georgescu *et al.*, 2021b). The advantage of features extractors for VAD is that they save computational capacity, so bigger architectures can be made, and they can add additional information like optical flow or object information. The main disadvantage is that they are biased towards the external dataset and an incorrect feature/object/optical flow estimation can lead to failure of the whole system.

#### 2.5.3.3 Data transformation

Data transformation refers to the modification of input data in various ways so that it helps in better anomaly detection. The transformed data is used in two ways: as an augmentation of data to the original input of the model, or as an auxiliary input along with the original input of the model. Some major works using data transformation in the first way are (Hasan *et al.*, 2016; Chong et Tay, 2017; Zhao *et al.*, 2017). Hasan *et al.* (2016) and Chong et Tay (2017) increase the size of their input data by generating video clips with different strides (stride-2 and stride-3) between frames. Zhao *et al.* (2017) instead augments the data by creating video clips with various transformations like random cropping, changing brightness and Gaussian blurring. The main motivation behind this type of data transformation is to train the model with a sufficiently large dataset with different input variations so that the model becomes adaptive and retain the most pertinent features of normal input.

Few important VAD works using data transformation in the second way are (Georgescu *et al.*, 2021a,b; Astrid *et al.*, 2021a,b; Park *et al.*, 2020). Georgescu *et al.* (2021a) transforms data into various ways for different tasks like reading frames backwards or not for arrow of time detection, or skipping frames in video clip for irregular/regular motion task, *etc.* Georgescu *et al.* (2021b) and Astrid *et al.* (2021a,b) transform data to form pseudo anomalies in order to use normal and pseudo-abnormal data while training.

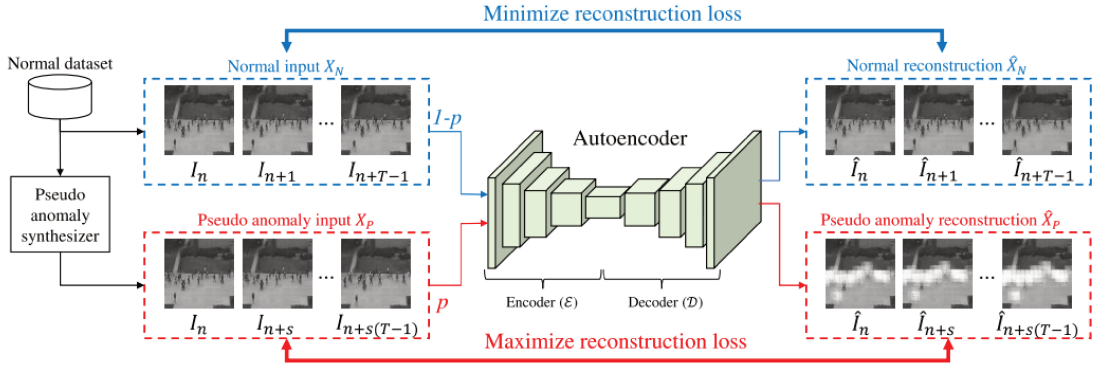



Figure 2.19: Autoencoder based VAD work of Astrid *et al.* (2021a) with normal input and pseudo anomaly input, where probability $p$ regulates quantity of pseudo anomalous input.

Figure 2.19 shows an example of VAD work that use pseudo anomalies synthesized from normal data as an auxiliary input to the AE. The pseudo anomalies are formed by temporally striding the input to create temporal incoherence in video clip, which simulates

an anomaly. The motivation is to train the AE in order to reduce the reconstruction loss for normal inputs while increasing it for pseudo anomaly inputs, thus making model sensitive to anomalies with temporally incoherent behavior.

#### 2.5.3.4 Clustering

It is the task of grouping or segmenting a collection of entities into subsets or "clusters", such that the entities within each cluster are more closely related to one another than to those assigned to other clusters (Hastie *et al.*, 2009). Concerning VAD, Chang *et al.* (2020) clusters the encoder output of an AE with reconstruction as proxy task. They propose a deep k-means clustering in encoder to obtain a more compressed data representation and extract the common factors of variation within the normal dataset. By minimizing the distance between the data representation and cluster centers, normal examples are closely mapped to the cluster center while anomalous examples are mapped away from the cluster center. The cluster centers can be deemed as a certain kind of normality within the training dataset. Since the model is only trained on normal events, the distance between cluster and abnormal representations is much higher than between the normal patterns. Reconstruction error and the cluster distance are together used as an anomaly score. With a similar strategy, Ouyang et Sanchez (2021) proposes to cluster the latent manifolds of the autoencoder using expectation maximization. Different from these approaches, Ionescu *et al.* (2019) first clusters the latent space of an autoencoder into different types of normality, and then train a binary classifier following the one-versus-rest scheme to separate normality clusters from one another. In the inference phase, a test sample is labeled as abnormal if the highest classification score is negative, *i.e.,* the sample is not attributed to any normality cluster.

#### 2.5.3.5 Memory, attention and other units

One of the main challenges with CNN based networks like autoencoder or GAN is that they have very powerful representational capacity which can even hinder to distinguish between normality and abnormality, if used naively. For example, reconstruction-based VAD methods using AE may even reconstruct the abnormal frames well (Zong *et al.*, 2018; Liu *et al.*, 2018). Hence, the assumption that reconstruction error is comparatively high for abnormal test frames might be violated.

One popular approach to handle this issue is to use memory units with the network. The prototypical patterns of normal data are recorded into the memory during training, whereas while testing on a test input, the most relevant elements from the memory are retrieved to perform the given task (reconstruction, prediction, *etc.*). Figure 2.20 shows one such approach, namely MemAE (Gong *et al.*, 2019). MemAE or memory-augmented autoencoder is a reconstruction task-based VAD method, which learns and updates the memory contents during training, to store the prototypical elements of the normal data. In the test phase, the memory is fixed, and reconstruction is performed using items selected from the memory. Taking this work forward, Park *et al.* (2020) propose a similar strategy to enhance AE for VAD by an improved memory module. They separate stored memory items explicitly using feature compactness and separateness losses, which enables using a small number of memory items compared to MemAE (10 vs 2,000 for MemAE). Furthermore, they also update the memory at test time, while discriminating anomalies simultaneously, suggesting their model memorizes normal patterns of test data. Overall, their model better records diverse and discriminative normal patterns for VAD. Finally, a

recent work by Liu *et al.* (2021b) uses multiple memory units attached in different parts of the AE and they propose an ideal arrangement of these memory units along with skip connections for the VAD task.

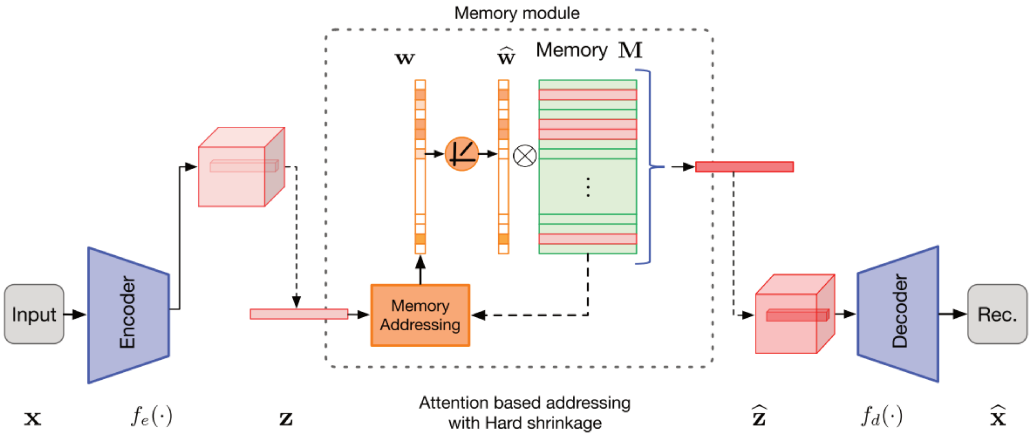



Figure 2.20: Proposed schema of MemAE (Gong *et al.*, 2019). The encoder takes the input $\mathbf{x}$ to produce encoding $\mathbf{z}$, which is taken as a query by the memory addressing unit to obtain the soft addressing weights $\mathbf{w}$. The memory slots can be used to model the whole encoding or the features on one pixel (as shown in the figure). The updated encoding $\widehat{\mathbf{z}}$ passes via decoder to produce the reconstruction $\widehat{\mathbf{x}}$.

Another important approach to enhance neural network is to use attention modules (Zagoruyko et Komodakis, 2016; Vaswani *et al.*, 2017; Wang *et al.*, 2019). In simple words, attention is a technique that imitates human cognitive attention, enhancing a part of input, such as an object and neglecting the remaining parts (Zhou *et al.*, 2019). Concerning VAD, Le et Kim (2022) proposed an attention-based residual autoencoder with future frame prediction as the proxy task. To exploit channel dependency of features, they propose a channel attention module made of two convolutional layers and it is applied in each layer of the decoder. Similarly, Chang *et al.* (2020) proposed a cluster driven autoencoder with future frame prediction task, where a variance-based attention module is designed to assign an importance to moving part of video clips, thus improving detection of anomalies with large temporal movements like person running, jumping, *etc.* Different from these approaches, Lv *et al.* (2021) proposed an AE-based VAD method with a dynamic prototype unit. They learn diverse patterns of the normal data in the form of prototypes. Prototype is a compact representation of pertinent normal data. A novel attention operation on the AE encoding map assigns a normalcy weight to each pixel location to form a normalcy map. Then, prototypes are obtained as an ensemble of the local encoding vectors under the guidance of normalcy weights. Multiple parallel attention operations are applied to generate a pool of prototypes, which represent diverse and compact dynamics of the normal patterns. Finally, the AE encoding map is aggregated with the normalcy encoding reconstructed by prototypes for latter frame prediction.

Recently, some other types of units have been proposed to tackle with the issue of powerful representational capacity of CNN based VAD networks. Szymanowicz *et al.* (2022) proposed a vector quantized autoencoder for detecting video anomalies using the future frame prediction task. They store features of the encoder in a learnable codebook using a vector quantization module. The vector quantization leads to discretization of high-level features into embeddings, which are used by decoder to predict the future frame. Since decoder has no direct access to input features but only the discrete embeddings of

codebook, it is unable to correctly predict the anomalous frames, thus improving the video anomaly detection. Similarly, Cho *et al.* (2022) proposed normal density estimation (NDE) unit which estimate the density of normality in their AE based VAD method.

| Method | Input | Task | Principal compo. | Auxiliary compo. |
|---|---|---|---|---|
| Hasan *et al.* (2016) | VC | Recon | AE | DT |
| Chong et Tay (2017) | VC | Recon | AE, CLSTM | DT |
| Zhao *et al.* (2017) | VC | Pred, Recon | AE | OF, pre., DT |
| Ravanbakhsh *et al.* (2017) | FF | Recon | Adv | OF, pre. |
| Tran et Hogg (2017) | VCP | Classification | AE, SVM | OF |
| Lee *et al.* (2018) | VC | Pred | AE, CLSTM, Adv | ✗ |
| Sabokrou *et al.* (2018) | FP | Classification | Adv | ✗ |
| Liu *et al.* (2018) | VC | Pred | AE, Adv | OF, pre. |
| Nguyen et Meunier (2019) | FF | Recon | AE, Adv | OF, pre. |
| Ye *et al.* (2019) | FF | Recon | AE, Adv | ✗ |
| Gong *et al.* (2019) | VC | Recon | AE | mem. |
| Ionescu *et al.* (2019) | FO | Classification | AE, SVM | clust., pre. |
| Dong *et al.* (2020) | VC | Pred | AE, Adv | OF, pre. |
| Tang *et al.* (2020) | VC | Pred, Recon | AE, Adv | ✗ |
| Yu *et al.* (2020) | VCO | SST | AE | OF, pre. |
| Chang *et al.* (2020) | VC | Recon | AE | clust., att. |
| Park *et al.* (2020) | VC | Pred | AE | mem. |
| Ouyang et Sanchez (2021) | VCO | Recon | AE, GMM | pre., clust. |
| Liu *et al.* (2021b) | VCO | Pred, Recon | AE | OF, pre., mem. |
| Georgescu *et al.* (2021a) | VCO | SST | AE, FCN | pre., DT |
| Georgescu *et al.* (2021b) | FO | Classification | AE, Adv | OF, pre., DT |
| Astrid *et al.* (2021a) | VC | Recon | AE | DT |
| Astrid *et al.* (2021b) | VC | Recon | AE | DT |
| Lv *et al.* (2021) | VC | Pred | AE | att. |
| Le et Kim (2022) | VC | Pred | AE | att., pre. |
| Cho *et al.* (2022) | VC | Recon | AE | NDE |
| Szymanowicz *et al.* (2022) | VC | Pred | AE | codebook |
| Park *et al.* (2022) | VCP | Pred | AE | DT |

Table 2.1: Review of major video anomaly detection methods.

Table 2.1 presents a review summary of various VAD methods. The first column refers

to the publication associated with the method and the second column shows various input data (refer Section 2.5.1). In third column, the proxy tasks are mentioned where Pred, Recon and SST refer to prediction, reconstruction, and self-supervised task, respectively. The fourth column refers to main learning component of the method, where the undeclared terms Adv, CLSTM, GMM and FCN refer to adversial unit, convolutional LSTM, gaussian mixture modeling and fully connected network. Final column is of the auxiliary component, where the abbreviated terms are: DT - data transformation, OF - optical flow, pre. - pre-trained feature extractor, clust. - clustering, mem. - memory unit, att. - attention unit and NDE - normal density estimation. We can observe the following from Table 2.1. Most of the methods use video clip based inputs like VC, VCP and VCO and only a very few methods use the frame based input. This signifies that rather than using just the spatial data through a single frame, spatio-temporal data via video clips is essential for VAD. Regarding proxy tasks, reconstruction task is the most preferred, followed by the prediction task, and they are often combined together. The classification task has been used only four times and it uses patch or object type inputs, requiring pre-processing through an external detector or data transformation. This signifies that it is not easy to perform classification task directly on the raw frame or video clip. The SST task has been recently used for VAD and there are only two methods that use them. Both of them use VCO as input, *i.e.,* they need an external object detector to first detect objects and then perform the task. They obtain excellent performances as they do not suffer from scene background and associated noise. However, one must note that in real life, we cannot always detect all the possible anomalous object and furthermore we might not even know which objects will cause anomaly (unlike offline datasets where we know which objects are potentially anomalous). The most used learning component is the autoencoder (2D CAE, 3D CAE, *etc.*), but other components like GAN, CLSTM, *etc.,* are also used. The choice of learning component depends on the chosen task, input and architecture design. For auxiliary components, pre-trained detector, optical flow and data transformations are mostly used. These components ensure the input data needed for the methods concern. Rest of the auxiliary components like memory, attention, clustering, codebook, *etc.,* are used to reinforce the learning component. This signifies that most learning components do require this auxiliary help to better detect anomalies. To conclude, reconstruction, prediction, SST, or their combinations should be used for VAD, with minimum or no auxiliary component. Ideally, video clip should be used as input, since it is the spatio-temporal data and does not require any processing or external supervision of any sort, thus it is an unbiased input.

## 2.6 Perimeter intrusion detection (PID) methods

In this section, we review the existing perimeter intrusion detection methods. It must be noted that only a few methods tackle the complete PID task. Unlike VAD, the PID task has not gained enough scientific attention in recent years, even though private enterprises kept on improving their intrusion detection systems, but they did not share their works publicly. One simple evidence to support this claim is that there is only one public dataset available for this task, unlike the VAD with many datasets. Nevertheless, we review the few existing public methods for this task.

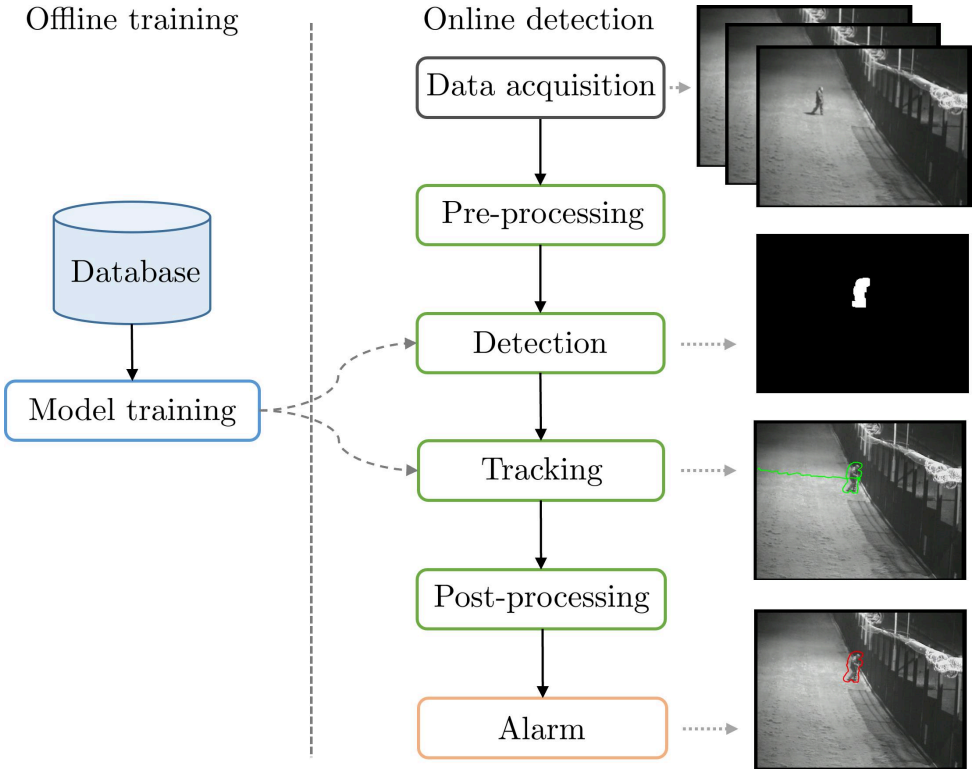



Figure 2.21: Typical pipeline of a PIDS, composed of an optional offline training (left part) and an online detection (middle part). An illustrative example of different steps in online detection is shown in right part of the figure (best viewed in color).

PID is closely related to other computer vision tasks like motion detection, object detection, tracking, *etc.* (Valera et Velastin, 2005). Indeed, these tasks are part of the typical PIDS pipeline as shown in Fig. 2.21. In order to examine PID methods, we must review methods that tackled one or several of these tasks with the aim to improve the PIDS pipeline. We describe below various parts of the pipeline, along with some major methods. It should be noted that here we do not describe data acquisition schemes since it has been already explored in Section 1.2.

### 2.6.1 Pre-Processing

In the case of protecting a perimeter, the cameras are usually CCTV and therefore they are not necessarily of the highest quality. It is because there are several cameras for each site and it is economically not viable for most people to invest too much in expensive cameras. Also, the task is just to detect the presence of an intrusion and not necessarily do profound analysis like face detection of an intruder (which would require high quality cameras). This low-quality camera sensor and adverse environmental conditions such as snow, fog, rain, extreme sunshine, *etc.*, may produce highly noisy video streams. We cannot directly feed this noisy data into the detection algorithm. Therefore, video enhancement is needed to remove noise and improve the visual appearance of the video. We can classify the existing video enhancement methods into two broad categories (Agaian *et al.*, 2007; Rao et Chen, 2012): spatial domain enhancement and transform domain enhancement.

Spatial domain video enhancement deals directly with pixels, *i.e.,* it makes a direct manipulation of pixels in video frames. It is conceptually simple and has a low time complexity, which favors real-time implementation but lacks robustness. Some surveys

on this method can be found in (Bennett et McMillan, 2005; Mittal *et al.*, 2006; Rao *et al.*, 2011). In most PIDS, some standard spatial enhancement is carried out on raw frames (Buch et Velastin, 2008, 2014; Cermeño *et al.*, 2018), such as image resizing, image normalization, mean centering, color space conversion (RGB to grayscale or vice versa), histogram equalization, *etc.*

Contrary to spatial domain, the transform domain video enhancement operates on the transform coefficients of the video frame, such as Fourier transform, discrete wavelet transform and discrete cosine transform (Ali, 2007; Rao *et al.*, 2011). The video quality is enhanced by manipulating these coefficients. This category of methods has a low computational complexity with the ease of manipulating the frequency composition of the video frame. Some major examples of PIDS using these techniques are (Buch et Velastin, 2008, 2014). They first use fast Fourier transform (FFT) on video frame patches and then decrease noise by removing very low or high frequencies from the FFT.

Apart from video enhancement, some other pre-processing can be conducted depending on the PIDS. Buch et Velastin (2014) first define patches of 16-pixel squares in each video frame. Then, they designate two regions per frame: grass and fence area for segregating the scene into an authorized and unauthorized zone. Their pre-processing is demonstrated in Figure 2.22. Similarly, another common pre-processing is to have a fixed spatial perimeter in each frame of the video (Cermeño *et al.*, 2018; Kim *et al.*, 2018b; Nayak *et al.*, 2019). This helps the PIDS to focus only on this region of the scene and to ignore activities outside this perimeter.

Figure 2.22: Pre-processing proposed by Buch et Velastin (2014), applied on the i-LIDS dataset. After normalization of video frame, patches of 16 × 16 pixels with 20% overlap are constructed on the region of interest, *i.e.,* ground and fence. These patches are used as input for their PIDS instead of the whole video frame.

### 2.6.2 Detection

This is one of the most important steps of the pipeline since the goal of a PIDS is to detect certain categories of objects that might cause an intrusion. Detection in video can belong to one of the two families: (i) detection of blobs, analyzing the pixel motion, and (ii) detection of objects, analyzing the image appearance with the localization and classification of objects. We describe below various detection strategies and associated

PID methods.

#### 2.6.2.1 Motion Detection

Intrusion is caused by a moving object in a protected perimeter during an unauthorized time. Therefore, motion detection is essential in a PIDS. The principal approaches can be classified into three categories.

**a) Optical Flow** It literally refers to the displacements of intensity patterns. It is an approximation of image motion defined as the projection of velocities of 3D surface points onto the imaging plane of a visual sensor (Beauchemin et Barron, 1995). Optical-flow-based methods use partial derivatives with respect to the spatial and temporal coordinates to calculate the motion between video frames. One key benefit of using optical flow is that it is robust to multiple and simultaneous cameras and camera shakes. However, most optical flow methods are computationally complex, very sensitive to noise and tough to implement in real-time settings. Some surveys on optical flow approaches are Beauchemin et Barron (1995); Stiller et Konrad (1999). Concerning PIDS, Kim *et al.* (2018b) implemented an optical flow based detection to compare it with their proposed detection method.

**b) Temporal Differencing** It uses pixel-wise differences among consecutive video frames to extract moving regions. This technique is adaptive to dynamic environments and has a low computational complexity. However, it can fail to extract all of the relevant pixels and can leave holes in regions. Some important studies can be found in (Sehairi *et al.*, 2017; Dimitriou *et al.*, 2017; Tsakanikas et Dagiuklas, 2018; Cermeño *et al.*, 2018). The works of Buch et Velastin (2008, 2014) use simple inter-frame difference followed by some morphological operations for motion detection in their PIDS.

**c) Background Subtraction** This is one of the most popular and key techniques for detecting moving objects in video. Background subtraction detects moving regions by taking the difference between the current frame and the reference frame, often referred to as the 'background model'. The detection ability depends on the adaptiveness of the background model. Some popular background subtraction methods are: running Gaussian average (RGA) (Wren *et al.*, 1997), Gaussian mixture model (GMM) (Stauffer et Grimson, 1999), kernel density estimator (KDE) (Elgammal *et al.*, 2000) and visual background extractor (ViBe) (Barnich et Van Droogenbroeck, 2009). Background subtraction mainly suffers from illumination changes, dynamic background, shadows, camouflage, video noise, *etc.* (Bouwmans, 2011). These effects can present a background object as a false foreground moving object or vice-versa. Most comprehensive surveys on background subtraction-based methods are (Hu *et al.*, 2004; Bouwmans, 2011; Xu *et al.*, 2016).

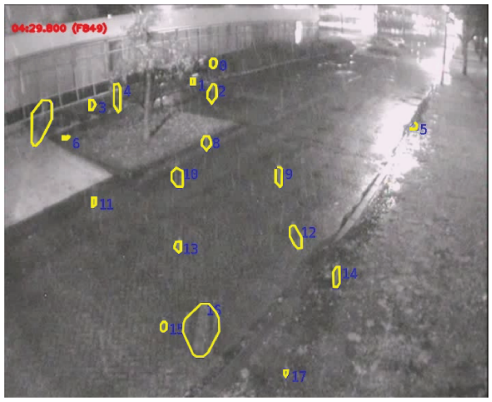
Concerning PIDS, Buch et Velastin (2008, 2014) use Gaussian background modeling to discriminate people (intruder) from background. The study of Vijverberg *et al.* (2014) uses background subtraction to extract object blobs from the video frame, which are later used for tracking. The work of Kim *et al.* (2018b) detects moving objects by comparing a background model with an input video frame in real-time. Figure 2.23 illustrates the performance of their method. Finally, Cermeño *et al.* (2018) detect the potential intruder object using the RGA method.

Figure 2.23: Moving object detection using background subtraction proposed by Kim *et al.* (2018b). The first column represents frames from a thermal and color camera respectively, with objects of interest in white bounding boxes. The second column demonstrates the output after background subtraction and image binarization, where white regions represent detected moving objects.

As already stated, background subtraction also has some drawbacks. In Figure 2.24, we demonstrate this using two examples proposed by Vijverberg *et al.* (2013). The first example, depicted in Figure 2.24a, shows false detections due to snowfall. Since the background subtraction models scene background and detects any moving object, it produces false alarms for these moving snow flakes. These are termed as false alarms due to the dynamic background. In Figure 2.24b, the second example shows split-object detections due to camouflage effect as object appearance matches with background (both have similar white texture).

#### 2.6.2.2 Object Detection

In the last decade, object detection, *i.e.*, object localization and classification, has been a field of intensive research (Zou *et al.*, 2019). Intrusion detection is closely related to it as intruders belong to certain categories of objects, such as people, car, bike, *etc.*, to be detected in a protected area. Even though intrusion can be caused by vehicles, animals, *etc.*, the state of art mostly focuses on detecting people as intruders.



(a) False detections due to snowfall in a private dataset. Yellow blobs show detections along with their index number in blue.

(b) Object split into extra detections in i-LIDS dataset. Blue blobs show split-object detections along with their index number.

Figure 2.24: Examples of incorrect detections using background subtraction method proposed by Vijverberg *et al.* (2013).

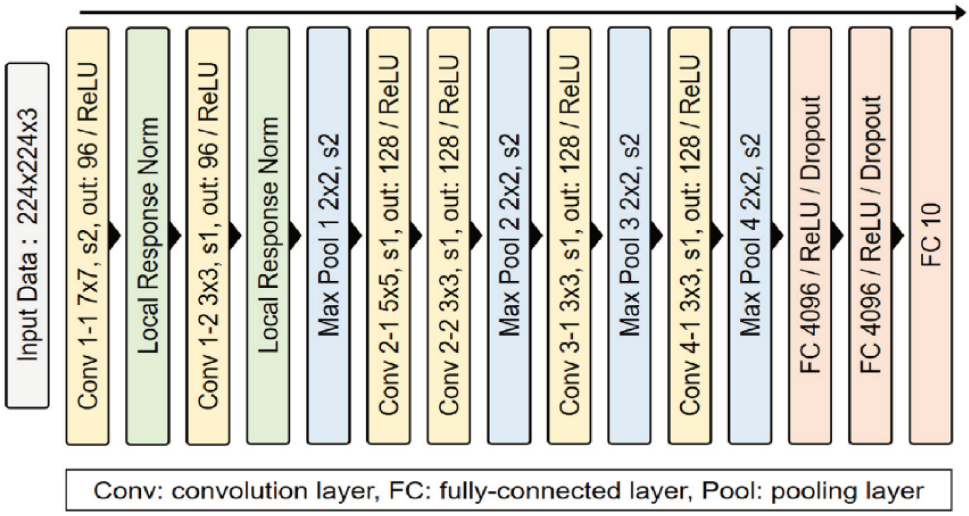



Figure 2.25: Moving object classification architecture proposed by Kim *et al.* (2018b). Input frame of size $224 \times 224 \times 3$ goes through a series of 2D convolutions, local response normalization and max-pooling units, followed by fully connected layers to finally give predictions of length ten, where each unit depicts probability of a particular class from a list of human (intruder) and wild animal classes.

Object detection methods can be classified into traditional and deep-learning-based detectors (Zou *et al.*, 2019). Some traditional object detectors are the Viola–Jones detector (Viola et Jones, 2001), histograms of oriented gradients (HOG) detector (Dalal et Triggs, 2005) and deformable part-based model (DPM) (Felzenszwalb *et al.*, 2008). Deep learning based methods such as Faster R-CNN (Ren *et al.*, 2015) and YOLO (Redmon et Farhadi, 2017), have achieved remarkable performances in object detection domain. Still human detection can be challenging, especially in scenes with an atypical human pose, such as crawling or creeping, occluded scenes (Patino *et al.*, 2021) and scenes with low luminosity, such as during night. In regard to PIDS, Nayak *et al.* (2019) uses a pre-trained

YOLO network for intruder object detection. Kim *et al.* (2018b) first detects moving objects with background modeling and then classifies them into one of the ten classes, containing wild animals and human (intruder). For this object classification, they propose a 2D CNN-based supervised classifier as shown in Figure 2.25.

### 2.6.3 Tracking

It is essential for a PIDS to keep track of a detected intruder object, otherwise it will be re-detected (or not) and may cause unnecessary alarms. Furthermore, perimeter protection solutions may use this information to impose detection constraints. For example, leaving an area can be allowed, but not entering it. We can also think of raising an intrusion alarm only if an object is inside the area for a specified amount of time.

Buch et Velastin (2008) proposed to use the Kalman filter on the texture of objects with a motion mask to build object tracks. Particle filters have also been used to track intruders (Cermeño *et al.*, 2018; Kim *et al.*, 2018b). In (Buch et Velastin, 2014), an intruder is tracked by logging positions of foreground objects over time. The work of (Vijverberg *et al.*, 2013) proposes a tracking algorithm based on tracklet clustering. Finally, Nayak *et al.* (2019) uses the simple on-line and real-time tracking (SORT) algorithm (Bewley *et al.*, 2016) for tracking intruders.

### 2.6.4 Joint Detection and Tracking

As video stream has two components, spatial and temporal, it is usually analyzed in two steps. The first step captures spatial patterns by using detection on each frame (see Section 2.6.2), whereas the second step uses tracking to apprehend temporal coherence (see Section 2.6.3). This approach creates the hypothesis that spatial and temporal dimensions are independent and can be processed sequentially.

Recent approaches jointly model spatial and temporal dimensions using 3D convolutions and improve results in video analysis (Tran *et al.*, 2015; Jiao *et al.*, 2021). Like anomaly detection, an autoencoder or GAN based on 3D convolutions/deconvolutions can lead to an implicit joint detection and tracking (Zhao *et al.*, 2017; Sun *et al.*, 2020).

### 2.6.5 Post-Processing

A major failure for the PIDS is when it misses to detect intrusions. In fact, ideally a system should always try to detect as much as possible, even at the cost of some false alarms (Cermeño *et al.*, 2018). These false alarms need to be filtered, which is why we need some form of post-processing. Even though this step is crucial in a PIDS, there are few publications on this topic because companies prefer to keep their post-processing confidential. However, despite this, we can list several post-processing techniques.

One of the most common post-processing is to filter objects of interest outside the chosen perimeter (Cermeño *et al.*, 2018; Kim *et al.*, 2018b; Nayak *et al.*, 2019). Sometimes, blobs are inconsistent across time, such as rain drops, and a filter can check the coherence of the blob trajectory. Detected objects can also be filtered with a minimum threshold on the blob size. For example, Cermeño *et al.* (2018) filters all the objects with a size less than four pixels. Since foreground objects are bigger than background ones, perspective calibration learns the dimension of object of interest as a function of its position in the scene. This allows to filter objects with a size smaller than the expected size of the object

of interest at the same position in the scene (AXIS Communications, 2020; FOXSTREAM Smart Video Analytics, 2021).

### 2.6.6 Alarm

In a perimeter intrusion detection system, it is essential to send an alarm signal to the security personnel to indicate the potential occurrence of an intrusion. This alarm signal alerts the client to verify if there is an intrusion in the site and then to take appropriate action against it. This alarm may also indicate a false alert in case there is no intrusion in the site. It is also very important to have this alarm as soon as the intrusion enters the surveilled place. The reason for this is that if the intrusion has already been inside the perimeter for long time, it would have already done the damage to the protected site. For all these reasons, in a PIDS, it is essential to send alarms at right time intervals (at the beginning of intrusions). The detected and tracked objects are transformed into alarms by using some sensibility thresholds to set the omissions–false alarms trade-off (Buch et Velastin, 2008; Kim *et al.*, 2018b; Nayak *et al.*, 2019; Vijverberg *et al.*, 2014). Generally, these thresholds are manually tuned during the actual deployment of the PIDS. Furthermore, some high-level rules can be applied to trigger the alarm. In the work of Buch et Velastin (2014), the alarm is triggered if the intruder moves towards the target for a minimum time of 2 seconds. In (Kim *et al.*, 2018b; Nayak *et al.*, 2019), the alarm is generated as long as the intruder is inside the protected perimeter. Figure 2.26 demonstrates this for the work of Nayak *et al.* (2019). Taking this forward, Cermeño *et al.* (2018) adds an extra constraint that the object must be tracked for at least three frames to give an alarm.

### 2.6.7 System Deployment

The deployment of a perimeter intrusion detection system is realized in three stages: an optional offline model training, then an online initialization of the system, and, finally, online execution.

#### 2.6.7.1 Model Training

There are some PIDS which require an offline training on part of the dataset for their detection or tracking steps (Vijverberg *et al.*, 2014; Kim *et al.*, 2018b; Nayak *et al.*, 2019). This training can be supervised, requiring labeled videos (tagging intrusion frames or events); or, it can be unsupervised, under the assumption that there is no annotated data. The supervised training can also be done on an external dataset and then the trained detectors are used for detection or tracking in PIDS. Vijverberg *et al.* (2014) proposed a classifier model trained as a multiple instance learning problem by employing image-based features to distinguish intruder objects from moving vegetation and other distractions. The works of (Kim *et al.*, 2018b; Nayak *et al.*, 2019) use supervised object detectors for detection step. So far, we do not have any work which use the unsupervised learning for perimeter intrusion detection.

#### 2.6.7.2 System Initialization

When a PIDS is installed in a new site, it may need several seconds to set the internal state of the system in alignment with the new scene. For example, to initialize the mean

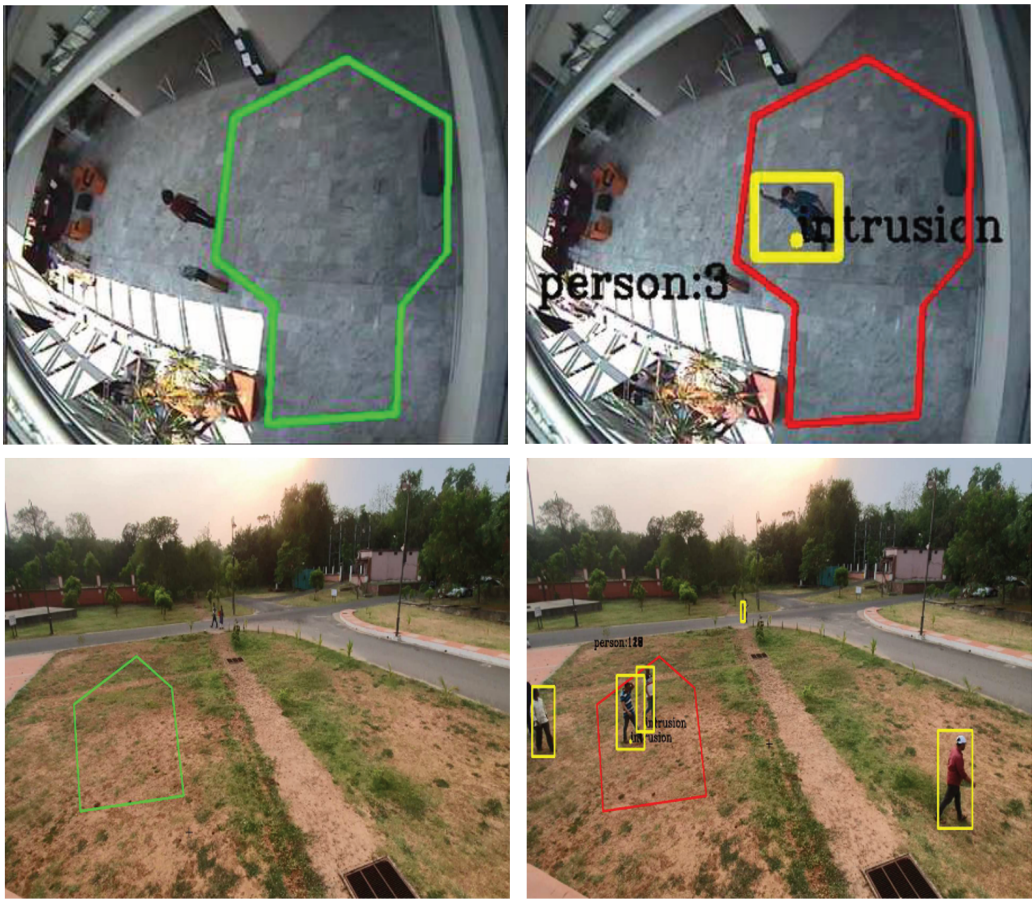



Figure 2.26: Working illustration of the PIDS proposed by Nayak *et al.* (2019) on CAVIAR (Crowley *et al.*, 2005) (first row) and a private dataset (Nayak *et al.*, 2019) (second row). The first column presents video frames from two different sites with green zones depicting the areas to protect. In the second column, potential intruder objects detected by YOLO detector are in yellow bounding boxes and since some of them are inside the protected zone, the alarm is raised (shown with red zone).

and standard deviation of a GMM (Buch et Velastin, 2014; Cermeño *et al.*, 2018). This is called the system initialization. This online initialization must be well distinguished from the offline model training. Furthermore, a PIDS can have sensibility thresholds, which are manually tuned by the installer during the deployment.

#### 2.6.7.3 System Execution

The final stage is the online execution of the PIDS. It includes all the steps of the pipeline as illustrated in the right part of Figure 2.21. To provide a reliable protection, most of the systems work between 5 and 25 frames per second (Buch et Velastin, 2014; Cermeño *et al.*, 2018).

Table 2.2 summarizes major PIDS works with various methods used in different steps of the pipeline. It can be observed that most systems use the visual camera for data acquisition and only one system is multi-camera based. Most traditional systems use background modeling for detection, whereas recent deep-learning-based models use 2D CNN or YOLO detector. Concerning tracking, the Kalman filter, particle filter, tracklet-

based tracking and SORT algorithm are used. For alarms, most systems have their own method-specific rules. Three systems use supervised training and the rest do not have a training step.

| **Publication** | **Data Acquisition** | **Pre-processing** | **Detection** | **Tracking** | **Post-processing** | **Alarm** | **Model training** |
|---|---|---|---|---|---|---|---|
| Buch et Velastin (2008) | Visual camera | frame patches, FFT | inter-frame differencing | Kalman filter | ✗ | rule-based | none |
| Vijverberg *et al.* (2013) | Visual camera | ✓ | background subtraction | tracklet tracking | ✗ | ✗ | none |
| Buch et Velastin (2014) | Visual camera | frame patches, FFT | Gaussian background modeling | Kalman filter | ✗ | rule-based | none |
| Vijverberg *et al.* (2014) | Visual camera | ✓ | background subtraction | tracklet tracking | ✓ | rule-based | supervised |
| Kim *et al.* (2018b) | Multi-camera: visual and thermal | resize, calibration, perimeter | 2D CNN | particle filter | outside perimeter | rule-based | supervised |
| Cermeño *et al.* (2018) | Visual camera | perimeter | RGA | particle filter | object size rule | rule-based | none |
| Nayak *et al.* (2019) | Visual camera | perimeter | YOLO v2 | SORT | ✗ | rule-based | supervised |

Table 2.2: PIDS reviewed in chronological order, where columns represent steps of the pipeline and model training needs. ✗ denotes unavailability of the step, whereas ✓ denotes that the step is available but not detailed.

## 2.7 Conclusion

We showed that concerning spatio-temporal feature learning, there are broadly four approaches. The first type of approach uses 2D CNN for learning spatial features, followed by an RNN-type network for learning temporal features. The second approach has two network streams to learn different and independent features like appearance and motion, which are finally fused together to obtain the overall space-time features. The third approach uses 3D convolution based networks and jointly learns spatio-temporal features. Finally, the fourth approach is a hybrid approach, which include a combination of above approaches. In this thesis, we want to learn spatio-temporal features jointly in video, with minimum supervision possible. Therefore, the 3D convolution-based approaches are ideal for us, with new variants being also fast and low on computational complexity. We do not want to rely on external supervision for optical flow, *etc.,* thus two-stream or hybrid approaches are not suitable.

To move from these supervised spatio-temporal approaches towards unsupervised approaches, three principal modeling approaches can be used, *i.e.,* reconstructive, predictive and generative. Reconstruction models principally use auto-encoder with the aim to re-

construct the input video stream. Predictive models try to predict the future frames or sequences, given observed sequence. They use LSTM-based models and autoencoder variants. Finally, the generative models principally use generative adversarial networks to model the likelihood of video samples in an unlabeled dataset. We found all three approaches are extensively used in the literature and none of them is more suitable than others. The choice of approach depends on the video task being tackled, its requirements, and the spatio-temporal feature learner (like a 2D CNN, 3DCNN, etc.) being used. Our chosen 3D convolution-based approach can be used as 3D CAE in unsupervised learning. It can be used as a reconstructive model, predictive model and even as the generator of a generative model. In this thesis, we explore the first two approaches and also combine them. The generative model requires extra computation on top of the generator, via the discriminator, therefore we avoided that approach to be computationally light.

Concerning video anomaly detection, we found that most methods use video clip based inputs like VC, VCP and VCO as input, thus signifying the importance of spatio-temporal input data. Majority of methods use the reconstruction or prediction as proxy task and they accept any type of input. The classification and SST tasks are used less and they require the help of external object detector or data transformation for obtaining the input to their model. This external supervision is not only biased to the external dataset but also in real life, we cannot detect all the possible anomalous objects beforehand (unlike offline datasets where we know which objects are potentially anomalous). The most used learning component is the autoencoder but the choice of learning component depends on the chosen task, input and architecture design. To better detect anomalies, the learning components often require reinforcement from auxiliary components like memory, attention, clustering, codebook, *etc.* To conclude, reconstruction, prediction, SST, or their combinations should be used for VAD, with minimum or no auxiliary component. Ideally, video clip should be used as input, since it is the spatio-temporal data and does not require any processing or external supervision of any sort.

Like VAD, the task of perimeter intrusion detection concerns with detection of rare and unknown event, *i.e.,* intrusion. In real life PIDS, we do not have labeled data for these intrusions and thus we need to use an unsupervised approach. Irrespective of this, none of the methods in state of the art uses unsupervised learning. The simple answer to this is the lack of public dataset, *i.e.,* there is only one dataset called i-LIDS (i-LIDS Team, 2006). This dataset was released in 2006, so most traditional methods used non-deep learning based approaches. Furthermore, the i-LIDS dataset provide annotation in training set for developing supervised PIDS. This is potentially because during its launch time (2006), deep learning was not explored much, even less unsupervised deep learning.

In our PIDS review, we found that majority of works use their private datasets and mostly the visual camera is used for data acquisition and only one system is multi-camera based. Most traditional systems use background modeling for detection, whereas recent deep-learning-based models use 2D CNN or YOLO detector. Concerning tracking, the Kalman filter, particle filter, tracklet-based tracking and SORT algorithm are used. For alarms, most systems have their own method-specific rules. Three systems use supervised training and the rest do not have a deep learning based training step. Since no two systems use the same dataset or same part of the dataset and many use different evaluation strategies, it is impossible to quantitatively compare the existing methods. Overall, we observe that even though there are only human intruders in the i-LIDS dataset, it still has the potential because of various distractions, weather conditions and intruder approach manners. Therefore, still an unsupervised deep learning based approach can be evaluated

on this public dataset.

Since in this thesis, we explore for unsupervised deep learning approaches for both VAD and PID tasks. We will use 3D CAE as our initial point, as stated above. In Chapter 4, we propose a 3D CAE for PID with reconstruction as the proxy task. It has an adaptive thresholding mechanism to adapt with changing scene dynamics like weather, light conditions, *etc.*, and it will be discussed later in that chapter. In Chapter 5, we enrich our 3D CAE with different unsupervised and self-supervised tasks. This new model shows promising results for both VAD and PID tasks, without need of external supervision or memory modules.

# Chapter 3

# Defining tasks and their evaluations

In this chapter, we are concerned with formally defining the two tasks that we tackle in this thesis, *i.e.,* video anomaly definition and perimeter intrusion definition. We mathematically describe these two tasks in the first two sections. Then, we explain various evaluation protocols for these tasks. Since the existing protocols are not fully suitable for the PID task, we propose a new edge-level evaluation protocol for it. Next, we describe various evaluation metrics used for these tasks, with a proper explanation concerning which metrics should be used for each of the tasks. Part of this work was published in the special issue "Unusual Behavior Detection Based on Machine Learning" of the Sensors 2022 journal (Lohani *et al.*, 2022b).

## Contents

## 3.1 Introduction

In this thesis, we are focused principally on video anomaly detection (VAD) and perimeter intrusion detection (PID) task. Since past few years, both these tasks have attained a major attention in the computer vision research (Chandola *et al.*, 2009; Saligrama *et al.*, 2010; Kiran *et al.*, 2018; Aravamuthan *et al.*, 2020; Nayak *et al.*, 2021; Ramachandra *et al.*, 2022). Even after all this attention, their definitions still lack clarity, since they are not defined formally (Vijverberg *et al.*, 2014; Kim *et al.*, 2018b; Kiran *et al.*, 2018; Aravamuthan *et al.*, 2020; Ramachandra *et al.*, 2022). It is essential to mathematically define these tasks as it has a direct impact on the evaluation. For example, when a PID system detects an intrusion, it sends a short video to the security post, where a human operator validates the alarm as a true intrusion or otherwise. This short video, composed of several frames before and after the suspected intrusion, must contain the intruder so that the operator can decide to act on it. The end of the intrusion event is generally not relevant for this application. In practice, we would like to detect intrusion as soon as it occurs; thus, we must obey time constraints. This requires a suitable evaluation protocol that takes these constraints into account. Thus, in this chapter, we first mathematically define the two tasks. Then we describe the existing evaluation protocols and metrics, with a correct explanation of which protocols / metrics to be used for the two tasks. We also propose a new evaluation protocol for the PID task that conforms correctly with its definition and requirements.

## 3.2 Defining video anomaly detection

Video anomaly detection is the task of localizing anomalies in space and/or time in a video. In literature, anomalies are described simply as activities that are out of the ordinary (Chandola *et al.*, 2009; Saligrama *et al.*, 2010; Sodemann *et al.*, 2012). Anomalies are also known as abnormalities, novelties, and outliers among other similar terms. Examples of studied anomalies range from fighting and falling detection, to unattended baggage at airports, to a person loitering outside a building. According to Saligrama *et al.* (2010), video anomalies can be thought of as the occurrence of unusual appearance or motion attributes or the occurrence of usual appearance or motion attributes in unusual locations or times.

One simple implication of this definition is that video anomalies are context dependent. This signifies that an activity considered anomalous in one context or scene may be normal in another and vice-versa. For example, in one scene, skating is considered a normal activity, while in another, it is anomalous. Therefore, real-world anomalous events are complicated and diverse, and it is difficult to list all the possible anomalous events. It is indeed desirable that the anomaly detection algorithm does not rely on a big prior information about the events, as it can mislead the algorithm to detect/ignore certain types of normal/abnormal events.

Overall, the most fundamental question here is to determine what are the events which can be designated as anomalous. Anomaly definition has always been debated in the literature due to its subjective nature and the complexity of human behaviors (Jiang *et al.*, 2009; Saligrama *et al.*, 2010; Sodemann *et al.*, 2012; Kiran *et al.*, 2018). In general, an event is considered abnormal if it deviates from observed or learned ordinary events (*i.e.,* the event with low occurrence or statistical representation in the learned model) or

the event is not known, or it is outstanding. As aforementioned, another important aspect about the definition of anomalies is the context or environment of the scene. Besides, the difficulty of defining anomalous events increases as the semantics of events grow. In other words, the greater the semantics of an event, more complex situations must be described to determine the anomalous events out of the normal.

To determine whether an event is anomalous, there are two main scenarios: (i) the expected anomalies are known, (ii) the expected anomalies are unknown. In the first scenario, we know some information of anomalies in the data, *i.e.,* we have some sort of labels for anomalies available during training. It is exceptional to have this since anomalies are rarely labeled beforehand. In the second scenario, we just have the data without any anomaly information. It is assumed that this data is only normal, *i.e.,* without anomalies. This is a more plausible setting since anomalies are rare and it is not possible to have all potential normal and abnormal samples available during training. The main challenge here is to understand pertinent features of normal events from a diversity of scenes and contexts, which would lead to distinguish anomalous events from the normal events (Sodemann *et al.*, 2012; Kiran *et al.*, 2018).

Please note that throughout the manuscript anomaly and abnormality denotes the same thing, *i.e.,* they are synonyms for each other. Similarly, their adjective forms anomalous and abnormal are used interchangeably.

We define below the two cases concerning video anomaly detection.

### 3.2.1 VAD with anomaly information

It refers to video anomaly detection when normal examples and at-least some abnormal examples are known. We can also say that in this case we have at-least some supervision via anomalous and normal examples during training.

#### 3.2.1.1 Fully Supervised VAD

Conceptually, the most direct case is fully supervised VAD having well-defined explanation of normality and abnormality, *i.e.,* training data have labels for both normal and abnormal class. This case is particularly common for some VAD sub-tasks like fall detection (Vishwakarma *et al.*, 2007; Kong *et al.*, 2019), fighting detection (Esen *et al.*, 2013) and traffic violation detection (Fu *et al.*, 2005; Zen et Ricci, 2011). Concerning VAD, since most datasets do not have a training set with anomalies, some works try to rearrange the datasets to add anomalies in the training set and provide their annotations. For example, He *et al.* (2018) re-organized UCSD Ped datasets (Li *et al.*, 2013) and provided annotations. Similarly, Landi *et al.* (2019) and Liu et Ma (2019) provided bounding box annotations in UCF-Crime dataset. Given both normal and abnormal classes during training, a supervised binary classifier can efficiently make a distinction between normal and abnormal videos during testing. Some works on supervised video anomaly detection are (Huo *et al.*, 2012; He *et al.*, 2018; Liu et Ma, 2019; Landi *et al.*, 2019).

Even though fully supervised VAD can produce good results for a dataset, its outcome is not generalizable to other datasets. Furthermore, the big assumption of having labeled data is rarely feasible in real life, especially for mid to large length datasets, which would require extensive manual or semi-automated annotations. Since the portion of abnormal samples is extremely small in comparison to the normal samples (as anomaly is a rare event), there is a huge imbalance of classes in VAD, and this makes it even more difficult for a supervised classifier to perform well (Kim *et al.*, 2020). Moreover, the big diversity

of anomalies disturbs the proper training procedure and practically makes it infeasible to consider all anomaly types. Thus, supervised VAD has a very limited applicability in real-life applications.

#### 3.2.1.2 Semi-Supervised VAD

Since anomalies are very diverse and rare, it is implausible and computationally expensive to access the complete range of anomalous and normal events for training. However, using both normal and abnormal samples during training provides a better anomaly understanding to the model. Therefore, some works propose semi-supervised VAD, where they use a very small portion of abnormal samples along with the usual training dataset (with no anomalies) (Akcay *et al.*, 2018; Ruff *et al.*, 2019; Liu *et al.*, 2019).

#### 3.2.1.3 Weakly Supervised VAD

Another, more realistic form of supervision in VAD is weak supervision. It comprises of video-level labels (as opposed to video clip or frame-level labels), *i.e.*, a label indicating whether a video is normal or contains anomaly somewhere, but we do not know where (Sultani *et al.*, 2018). UCF-Crime dataset contains these kinds of labels by default (Sultani *et al.*, 2018). Researchers also re-arranged the well-known ShanghaiTech (Luo *et al.*, 2017a) dataset to have weakly labeled anomalous videos during training. It is known as re-organized ShanghaiTech (Zhong *et al.*, 2019) dataset. Some notable works on weakly supervised video anomaly detection are (Sultani *et al.*, 2018; Zhong *et al.*, 2019; Majhi *et al.*, 2021; Feng *et al.*, 2021; Tian *et al.*, 2021; Li *et al.*, 2022).

#### 3.2.1.4 Formalism: VAD with anomaly information

We assume $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively as the input and the output of a VAD system. $\mathcal{X}$ and $\mathcal{Y}$ represent respectively the input and output space. In the case of video anomaly detection, $x$ can be either a video clip, a video frame or something else like object in frames, depending on the method (refer Section 2.5.1 for more details), and $y \in \{0, 1\}$ is the associated label where 1 indicates anomalous class and 0 otherwise.
We assume a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$ composed of $N$ training samples. The goal is to learn a mapping function that can correctly predict the label $y$ given the input $x$. *In VAD with supervision, the anomalies are simply defined by labels given in the labeled dataset.*

In general, a discriminative model, represented by function $f$ and composed of parameters $\theta$ is supervisely trained as:

$$\theta^* = \arg\min_{\theta} \ \mathbb{E}_{x \sim \mathcal{X}} \left[ \ \mathcal{L}_{\text{dis}}(f_\theta(x), y) \ \right] \ , \tag{3.1}$$

where $\mathcal{L}_{\text{dis}}$ is a discriminative loss function. This can be easily used for the VAD task with a dedicated loss function.

Like the usual supervised binary classification setup, during testing phase, an input sample $x$ can produce a two-dimensional output (or a single dimensional output) $\hat{y} = f_\theta(x)$, where each dimension corresponds to the probability of normal and abnormal classes in $[0, 1]$. The class with higher probability is predicted.

### 3.2.2 VAD without anomaly information

In this case, we have training data with no labels. The main assumption is that the training data has none to very negligible amount of anomalies, which is true since anomalies are rare. In other words, the training data principally have normal events, *i.e.,* only one class. Based on this, a popular paradigm for anomaly detection is one-class classification, *i.e.,* to encode the usual pattern with only normal training samples (Kiran *et al.*, 2018; Ramachandra *et al.*, 2022). Then, the distinctive encoded patterns are detected as anomalies. In this technique, anomalies are detected solely based on intrinsic properties of normal data instances. The main idea is: "Something that has not being seen before is considered anomalous". Since in unsupervised VAD, we do not have strict assumptions about the training data, it leads to generalizability, especially for applications in which there is not a precise definition for anomalous events (Mohammadi *et al.*, 2021).

#### 3.2.2.1 Formalism

The formalism proposed here is inspired by probably approximately correct (PAC) learning framework (Valiant, 1984). In unsupervised VAD, the training set $\mathcal{D}$ does not contain labels, *i.e.*, $\mathcal{D} = \{x_i\}_{i=1}^{N} \subset \mathcal{X}$ with $N$ training samples. Usually, it is equivalent to assume that all videos of $\mathcal{D}$ are normal videos, *i.e.*, their labels $y = 0$. Hence, video clips follow a distribution $\mathcal{N}$, usually called normality.
A reconstructive model (Kiran *et al.*, 2018), like an autoencoder, is trained as:

$$\theta^* = \arg\min_{\theta} \ \mathbb{E}_{x\sim\mathcal{N}} [\ \mathcal{L}_{\text{rec}}(f_\theta(x), x)\ ] \ , \tag{3.2}$$

where $f$ is a reconstruction function composed of parameters $\theta$ and $\mathcal{L}_{\text{rec}}$ is a reconstructive loss function. During testing, a reconstruction error $\hat{y}$ is calculated for an input sample $x$ as:

$$\hat{y} = R(\ \mathcal{L}_{\text{rec}}(f_\theta(x), x)\ ) \ , \tag{3.3}$$

where $R$ is a common re-scaling function to transform the error into a probability in $[0, 1]$. Then, when the reconstruction error is low, *i.e.,* it is inferior to a threshold $\epsilon << 1$, the input sample is classified as normal with a high confidence, denoted $1 - \delta$ where $\delta << 1$:

$$p(\ x \sim \mathcal{N} \mid \hat{y} \leq \epsilon\ ) \geq 1 - \delta \ . \tag{3.4}$$

Conversely, when the reconstruction error is high, *i.e.,* it is superior to the threshold $1-\epsilon$, the input sample is classified as abnormal with a high confidence:

$$p(\ x \sim \mathcal{A} \mid \hat{y} \geq 1 - \epsilon\ ) \geq 1 - \delta \ , \tag{3.5}$$

where $\mathcal{A}$ is the distribution followed by abnormal videos.

Finally, we can define *abnormality as the complement of the distribution of training set*: $\mathcal{A} = \mathcal{X}\backslash\mathcal{N}$. It must be well noted that this representation takes care of complicated or debated cases for VAD using the uncertainty parameter $\delta << 1$. It denotes in Equations (3.4 and 3.5) that we can never be completely sure that if the reconstruction error is small, it is normal and if it is large, it is an anomaly. It other words, we cannot be sure that all representations far from distribution followed by training data should be anomaly. For example, in ShanghaiTech dataset, the normal activities like picking up stones and eating ice-creams are present only in the testing set and not the training set. Therefore,

even if the reconstruction error is high for these activities, we cannot be fully certain that they are anomaly due to the uncertainty parameter $\delta$. Like reconstructive models, this definition of anomalies can be easily extended to generative models also.

## 3.3 Defining perimeter intrusion detection

In simple words, the task of perimeter intrusion detection (PID) aims to detect the presence of an unauthorized moving object in a protected site during a certain time. As described before, PID is a specific type of VAD task. Furthermore, intrusions are a particular type of anomalies, classified as point and contextual anomalies (Chandola *et al.*, 2009). But contrary to VAD, we can more concretely define this task as it has certain unique attributes like perimeter, intruder object type, movement constraints and site protection time.

The PID task has been defined in various ways in the state-of-the-art. Vijverberg *et al.* (2014) defined a perimeter intrusion detection system (PIDS) as a monitoring system that identifies the presence of humans or devices in a pre-defined field of view. In (Kim *et al.*, 2018b), it is defined as a system that detects physical intrusions on a site having a protective barrier to isolate it from outside. Aravamuthan *et al.* (2020) described it as a system that detects the movements of intruders attempting to breach a security wall or region and alert security. However, all these definitions lack clarity and formalization; for example, the following questions need to be addressed: "what are intruders?", "does moving intruder cause intrusion?" and "is a perimeter necessary?". To answer all these questions concretely, we must mathematically define a PIDS. Before defining a PIDS, we need a definition of intrusion. Since objects cause intrusion, we first define an object in the video.

### 3.3.1 Object in the Video

We define a video $\mathcal{V}$ acquired for $n$ frames during the interval $\mathcal{T} = [1, n]$ as:

$$\mathcal{V} = \left\{ I_t \in \mathbb{R}^{H \times W \times D} \right\}_{t \in \mathcal{T}} \ , \tag{3.6}$$

where $I_t$ denotes the frame at the time instant $t$, with height $H$, width $W$ and number of channels $D$. To define an object in the video, we must first specify the object definition at frame-level. An object in a frame or image is defined with a spatial specification and a class that distinguishes one family of objects from another (such as humans, animals, or cars). The spatial specification can be either on pixel-level by allocating each pixel to an object or background, or on area-level by encapsulating the object in a bounding box. We choose the bounding box specification as it has been used in the literature extensively (Zou *et al.*, 2019). It should be noted that the choice of spatial specification (bounding box or otherwise) cannot have an impact on the intrusion definition. Thus, we define an object at frame-level with a class and a bounding box. To define an object in the video, we consider all of the frames where it is present. Therefore, an object $o_i$ in the video is defined as:

$$o_i = \left( \{ b_{i,t} \}_{t \in \mathcal{T}}, c_i \in \mathcal{C} \right) \ , \tag{3.7}$$

where $c_i$ is the class of the object from the set of object classes $\mathcal{C}$, and $b_{i,t}$ is its bounding box at time instant $t$, which is defined as:

$$b_{i,t} = \{ g_{i,t}, w_{i,t}, h_{i,t} \} \ , \tag{3.8}$$

where $g_{i,t}$, $w_{i,t}$ and $h_{i,t}$ are the center, width and height of the bounding box, respectively. The center is defined by its coordinates as $g_{i,t} = (x_{i,t}, y_{i,t}) \in I_t$. Note that, instead of the bounding box center, it is also possible to choose other points, such as the bounding box bottom, as reference. We illustrate these definitions in Figure 3.1.

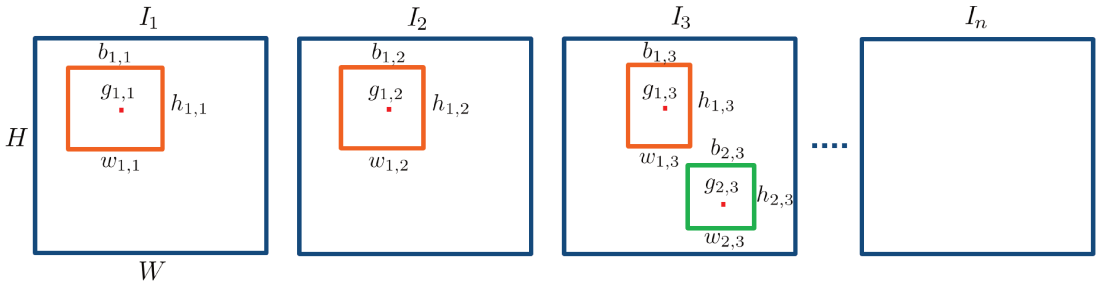



Figure 3.1: Video with $n$ frames of height $H$, width $W$ and channels $D = 1$. Two objects, shown in orange and green bounding boxes, are defined as $o_1 = (\{b_{1,1}, b_{1,2}, b_{1,3}\}, c_1)$ and $o_2 = (\{b_{2,3}\}, c_2)$, where $c_1, c_2 \in \mathcal{C}$ are the object classes. Here, $\{b_{1,1}, b_{1,2}, b_{1,3}\}$ are bounding boxes of object 1 on first three frames and $\{b_{2,3}\}$ represents bounding box of object 2 at frame 3.

### 3.3.2 Intrusion event

For the protection of a site, some parameters must be defined to qualify objects as non-authorized ($na$), *i.e.*, intruders.

- $\mathcal{S}_{na} \subseteq \mathbb{R}^{H \times W}$ : the subset of the frame/image, defining the surface to protect.
- $\mathcal{T}_{na} \subseteq \mathcal{T}$ : the time interval during which the surface must be protected (*e.g.*, protection only during night).
- $\mathcal{C}_{na} \subseteq \mathcal{C}$ : the set of non-authorized classes, such as person, car, truck, *etc.* These classes of objects are considered as possible intruders and can be different from one site to another.

Since $\mathcal{C}_{na}$ is a non-finite set (it is impossible to make the exhaustive list of non-authorized objects, exposing the system to omissions), it is easier to explicitly define a short list of authorized objects $\mathcal{C}_a$ (such as small animals), which leads to $\mathcal{C}_{na} = \mathcal{C} \backslash \mathcal{C}_a$.

An object causes an intrusion event if it belongs to a non-authorized class and is moving in a protected area during a prohibited time interval. Our hypothesis is that the object should be moving and not stationary to cause an intrusion event. It is to tackle the case where the unauthorized object enters and stays for a long period of time in the site. For example, if a car enters the site, it causes an intrusion event (assuming car belongs to non-authorized class) but later if the car stays stationary in the site for many days, it is not necessarily causing an intrusion event. We are interested in knowing if the intruder has entered the site, thus our hypothesis of moving object. We define the intrusion event caused by an object $o_i$ as:

$$\mathcal{IE}(o_i) = \{\, I_t \text{ s.t. } c_i \in \mathcal{C}_{na} \text{ and } t \in \mathcal{T}_{na} \text{ and } \|\text{grad } \vec{g}_{i,t}\| > 0 \text{ and } g_{i,t} \in \mathcal{S}_{na} \}_{t \in \mathcal{T}} \quad , \tag{3.9}$$

where $\|\text{grad } \vec{g}_{i,t}\|$ is the gradient of object $o_i$ at instant $t$ and it being non-zero signifies that the object is in motion. Thus, the intrusion event caused by object $o_i$ is a collection

of all the frames $I_t$ such that $t \in \mathcal{T}_{na}$, object class $c_i \in \mathcal{C}_{na}$, the gradient is non-zero and the bounding box center lies in the protected area.

Figure 3.2 illustrates the intrusion event caused by an object. The surface to protect $\mathcal{S}_{na}$ is depicted with a yellow trapezoid in each frame, and we assume that we want to protect it during the entire video. One object is present in the video, and it is shown with a rectangular bounding box plus a center. The object is in motion from the second frame until the eighth frame. While in motion, the center of object lies in $\mathcal{S}_{na}$ from fourth to seventh frame, causing an intrusion event. Thus, this object triggers an intrusion event for four frames.

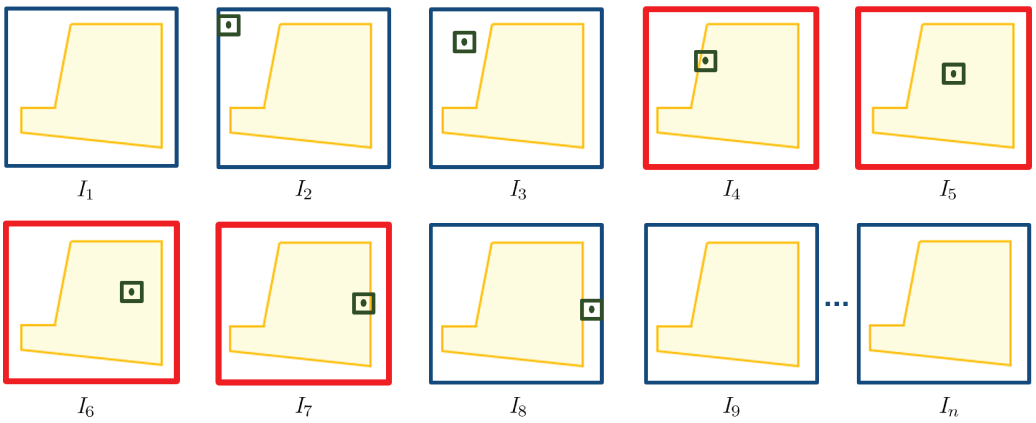



Figure 3.2: Illustration of an intrusion event caused by a single object. Video with $n$ frames, where $\mathcal{S}_{na}$ is shown with yellow surface and object with a green bounding box plus center. The object causes an intrusion event for four frames from frame $I_4$ to $I_7$, colored in red.

Since a video can have more than one object causing intrusion events, we define the intrusion events of the whole video containing $j$ objects as:

$$\mathcal{IE}(\mathcal{V}) = \bigcup_{i=1}^{j} \mathcal{IE}(o_i) \ . \tag{3.10}$$

### 3.3.3 Intrusion interval

Figure 3.3 shows three intrusion events caused by three objects in the video. We can observe that intrusion events of objects 1 and 2 overlap in time for two frames. It means that for those two frames, there were two objects causing intrusion events simultaneously. The intrusion events of this video is a collection of all the intrusion frames, marked by 1 (see Figure 3.3). In the context of video surveillance, we are concerned with whether there is an intrusion event, regardless of whether one object or many objects are causing it. Therefore, we are interested in an interval of a contiguous sequence of intrusion frames. We term this as an intrusion interval, and the task of intrusion detection is focused on detecting them. Formally, an intrusion interval $\mathcal{II} \subseteq \mathcal{IE}(\mathcal{V})$ is defined on a closed interval as:

$$\mathcal{II} = \{I_t \in \mathcal{IE}(\mathcal{V}) \text{ with } t \in [t_{start}, t_{end}] \text{ s.t. } I_{t_{start}-1} \notin \mathcal{IE}(\mathcal{V}) \text{ and } I_{t_{end}+1} \notin \mathcal{IE}(\mathcal{V})\} \ , \tag{3.11}$$

where $t_{start}$ and $t_{end}$ denote the first and last frames of an intrusion interval. In other words, an intrusion interval is a contiguous sequence of frames of maximal size derived from $\mathcal{IE}(\mathcal{V})$. Figure 3.3 depicts two intrusion intervals of the video.

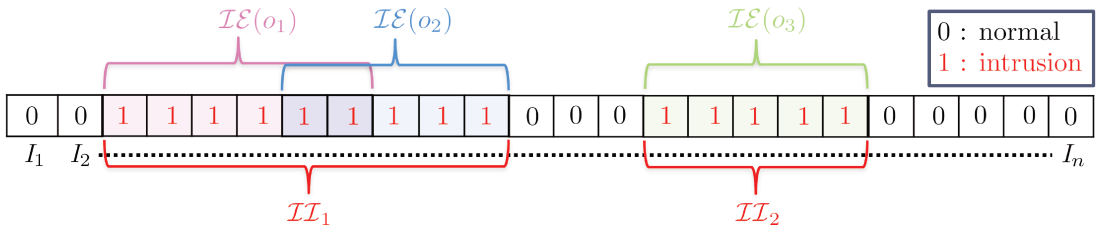



Figure 3.3: Illustration of intrusion events of the video and intrusion intervals. Objects $o_1$, $o_2$ and $o_3$ cause intrusion events $\mathcal{IE}(o_1)$, $\mathcal{IE}(o_2)$ and $\mathcal{IE}(o_3)$, marked with value 1. $\mathcal{IE}(\mathcal{V})$ is collection of all the frames with value 1. Two intrusion intervals $\mathcal{II}_1$ and $\mathcal{II}_2$ are shown in red intervals.

### 3.3.4 PIDS

Given a precise definition of intrusion, we can now define a perimeter intrusion detection system (PIDS). Given a video $\mathcal{V}$ and intrusion parameters $(\mathcal{S}_{na}, \mathcal{T}_{na}, \mathcal{C}_{na})$, the prediction of a PIDS can be defined as:

$$\mathcal{P}(\mathcal{V}, \mathcal{S}_{na}, \mathcal{T}_{na}, \mathcal{C}_{na}) = \{\hat{p}_t \in \{0,1\}\}_{t \in \mathcal{T}_{na}} \ , \tag{3.12}$$

where $\hat{p}_t$ is a binary prediction for each frame $t$ of video $\mathcal{V}$ for time $\mathcal{T}_{na}$, with 1 denoting a frame predicted as an intrusion, and 0 otherwise. Therefore, a PIDS classifies each frame into an intrusion frame or otherwise. This type of output is useful when we want to evaluate a PIDS at frame-level (refer Section 3.4.1). In a real-life surveillance system, the system sends an alarm signal to surveillance personnel as soon as there is a transition from a normal to intrusion state (i-LIDS Team, 2006). The output of the PIDS $\mathcal{O}(\mathcal{P})$ can be derived from $\mathcal{P}$ as follows:

$$\mathcal{O}(\mathcal{P}) = \{\ \hat{p}_t \in \mathcal{P} \ \text{ s.t. } \ \hat{p}_{t-1} \in \mathcal{P} \text{ with } \hat{p}_{t-1} = 0 \text{ and } \hat{p}_t = 1\ \}_{t \in \mathcal{T}_{na}} \ . \tag{3.13}$$

This is a set of intrusion alarms created by the system, marked by the rising edge, *i.e.*, the transition of the system state from non-alarm to alarm. These alarms alert the surveillance personnel about a suspicious activity. For each alarm, a mini clip is sent containing some frames before the alarm and some frames after the alarm. The surveillance personnel visually analyze this mini-clip and decide whether it is an actual intrusion activity or a false alarm. Therefore, we need an evaluation scheme that takes into account this real-life constraint.

## 3.4 Evaluation protocols

Given a video or set of videos and the requested task, the system can detect intrusions or anomalies. To evaluate the performance, we need to compare the predicted output with ground truth annotations. The way this evaluation is carried out is essential as each evaluation scheme quantifies different aspects of the system and depending on the task, one should wisely choose the correct evaluation methodology. The following subsections present different evaluation protocols. It must be noted that only frame-level evaluation is used in both the tasks and other evaluation protocols are only used for the PID task.

### 3.4.1 Frame-level evaluation

As the name suggests, in this type of evaluation, we are interested in checking whether each frame of the video is correctly classified or not.

#### 3.4.1.1 VAD task

Irrespective of the VAD types and formalisms defined in Section 3.2, every system outputs whether a frame is normal or abnormal. Since per-frame anomaly ground truth is normally available in VAD datasets (refer Section 1.4.1), the predictions can be easily compared with it. It is simply the binary classification evaluation of each frame of the video (Sokolova *et al.*, 2006). First, the elements related to the confusion matrix, *i.e.*, true positive (TP), false negative (FN), false positive (FP) and true negative (TN), are computed with anomaly as the objective class. Then, the performance of VAD is evaluated using the metrics, such as the precision, recall, AUROC, AUPR, *etc.*, as defined in Section 3.5.

In frame level evaluation, the anomalies are evaluated at the lowest level, *i.e.,* frames. This means that here we loose the notion of anomalous events. For example, if we have a good frame level score, it does not necessarily mean that all anomalous events are well detected and maybe a few long anomalous events are contributing a large portion of the overall score while many small anomalous events are missed or incorrectly detected. In short, a good frame level anomaly score just gives an overall idea for a VAD performance and does not guarantee if most anomalous events are correctly detected. Even after these drawbacks, the VAD community still mostly use the frame-level evaluation protocol (Popoola et Wang, 2012; Kiran *et al.*, 2018; Nayak *et al.*, 2021; Ramachandra *et al.*, 2022). In this work, we continue with this evaluation so that we can fairly compare with other VAD works.

#### 3.4.1.2 PID task

For a video $\mathcal{V}$ and given intrusion parameters $(\mathcal{S}_{na}, \mathcal{T}_{na}, \mathcal{C}_{na})$, the frame-level ground truth is defined as:

$$\mathcal{G}(\mathcal{V}, \mathcal{S}_{na}, \mathcal{T}_{na}, \mathcal{C}_{na}) = \{\ p_t \in \{0,1\} \text{ s.t. } p_t = 1 \text{ if } I_t \in \mathcal{IE}(\mathcal{V})\ \}_{t \in \mathcal{T}_{na}}\ , \tag{3.14}$$

where $p_t$ is the ground truth label for each frame $t$ of the video $\mathcal{V}$ at time $\mathcal{T}_{na}$; value 1 denotes an intrusion class, and 0 otherwise.

Given ground truth $\mathcal{G}$ and prediction $\mathcal{P}$ (see Equation (3.12)), the frame-level intrusion evaluation is computed like frame-level evaluation of VAD task by computing confusion matrix elements and using metrics like precision, recall, $F_1$ score, *etc.*

In this type of evaluation, each frame contributes equally to the overall score. Thus, it can provide the same overall score for an algorithm that fails to detect multiple intrusion events altogether, *i.e.,* detecting no frames in those events, versus an algorithm that detects all intrusion events but does not detect some intrusion frames in multiple intrusion events. This is an undesirable evaluation in the case of intrusion detection because we cannot afford to have omissions of intrusion events. In reality, we are more interested in knowing if the system is able to detect the intrusion events correctly as a whole. This demands an event-level evaluation. In other words, we want to detect intrusion intervals ($\mathcal{II}$s) from the video. More specifically, we are interested in evaluating whether the beginnings of these intrusion intervals are detected correctly. This is because, if an intrusion event is detected too late, then that detection is not very useful (i-LIDS Team,

2006; Buch et Velastin, 2014). The idea is to detect each intrusion interval as soon as it occurs and, thus, we need an evaluation scheme that takes this into account.

### 3.4.2 i-LIDS evaluation

To correctly evaluate PID systems, the i-LIDS dataset is provided with an evaluation protocol (i-LIDS Team, 2006). It focuses on evaluating intrusion at the event-level rather than frame-level. To be precise, an intrusion is considered correctly detected if there is at least one system alarm within 10 seconds from the start of the intrusion event.
For an $\mathcal{II}$ of the video and alarms $\mathcal{O}(\mathcal{P})$, the rules of the i-LIDS evaluation protocol are as follows:

1. TP: if there is at least one alarm within 10 seconds from the beginning of the $\mathcal{II}$. If there are multiple alarm candidates, the first one is taken, and the rest are ignored.

2. FN: if there is no alarm within 10 seconds from the beginning of the $\mathcal{II}$.

3. FP: if there is an alarm but not within 10 seconds from the beginning of the $\mathcal{II}$. If there are consecutive FPs within a 5-seconds gap among them, only the first one is considered and the rest are ignored.

Apart from these, one rule is specific to the i-LIDS-dataset: all $\mathcal{II}$s and alarms that start within 5 minutes from the beginning of the video are ignored. This means that they wanted to give a preparation time to the system. This evaluation scheme is not generic and has several drawbacks, as illustrated in Figure 3.4. It penalizes an alarm as an FP after 10 seconds from the beginning of an $\mathcal{II}$ without considering the duration of intrusion. If the $\mathcal{II}$ has a long duration (such as an hour) and we have an alarm at the $11^{\text{th}}$ second, it is not ideal to mark it as an FP. From a practical point of view, the surveillance personnel will receive a mini clip as soon as the alarm is triggered and, if the intrusion is present, then it is not sensible to mark this as an FP. Instead, this alarm should be ignored as it is not detected within 10 seconds. Similarly, each alarm after 10 seconds but within $\mathcal{II}$ is considered as an FP, and this strongly penalizes the system precision. Instead, these extra alarms should be counted without assigning them as an FP.

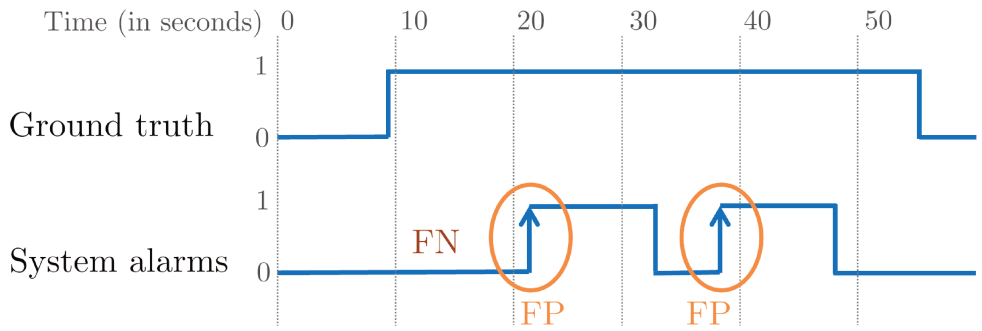



Figure 3.4: Illustration of i-LIDS evaluation protocol, highlighting its drawback on an intrusion example starting at $9^{\text{th}}$ second. Since no alarm has been raised in the first 10 seconds of the intrusion, a false negative (FN) is counted. Following alarms, at $22^{\text{nd}}$ and $37^{\text{th}}$ second, are marked as false positive (FP) because they do not occur within 10 seconds from the beginning of the intrusion.

### 3.4.3 Edge-level evaluation

To appropriately evaluate a PIDS while considering the real-world aspects, we propose a new evaluation protocol. An intrusion event begins with a transition from a non-intrusion to intrusion state, *i.e.*, we have a rising edge as shown in Figure 3.5. Similarly, an intrusion event stops by a reverse transition, *i.e.*, a falling edge. We are interested in detecting intrusion within a few frames from the rising edge. Since we focus on this rising edge, we call this the edge-level evaluation. In other words, we emphasize detecting the beginning of intrusion intervals. We first define the following terms from an intrusion interval of the video (see Figure 3.5).

The intrusion interval neighborhood IN is an expanded interval defined by $m_{\text{pre}}$ frames before and $m_{\text{post}}$ frames after the $\mathcal{II}$:

$$\text{IN}(\mathcal{II}, m_{\text{pre}}, m_{\text{post}}) = [t_{\text{pre}}, t_{\text{post}}] \text{ s.t. } t_{\text{pre}} = t_{\text{start}}(\mathcal{II}) - m_{\text{pre}} \text{ and } t_{\text{post}} = t_{\text{end}}(\mathcal{II}) + m_{\text{post}} .$$

These $m_{\text{pre}}$ and $m_{\text{post}}$ frames are in the range of one to five (less than 1/5 seconds for a video at 25 FPS) and are added in order to take into account the error of annotation. This error is because it is difficult to mark the exact frame at which the intrusion starts or ends. This tolerance further permits not strictly penalizing the system when an intrusion event is detected a few frames before the actual event or when the system detects a few more intrusion frames after the actual event is finished. These cases arise often when the intrusion object is in the scene but not inside the surface to protect. Therefore, IN is an interval where the actual intrusion activity takes place, and an alarm given by a PIDS in this interval can be counted as either TP or ignored. An alarm given outside IN must be a false alarm and should be counted as an FP.

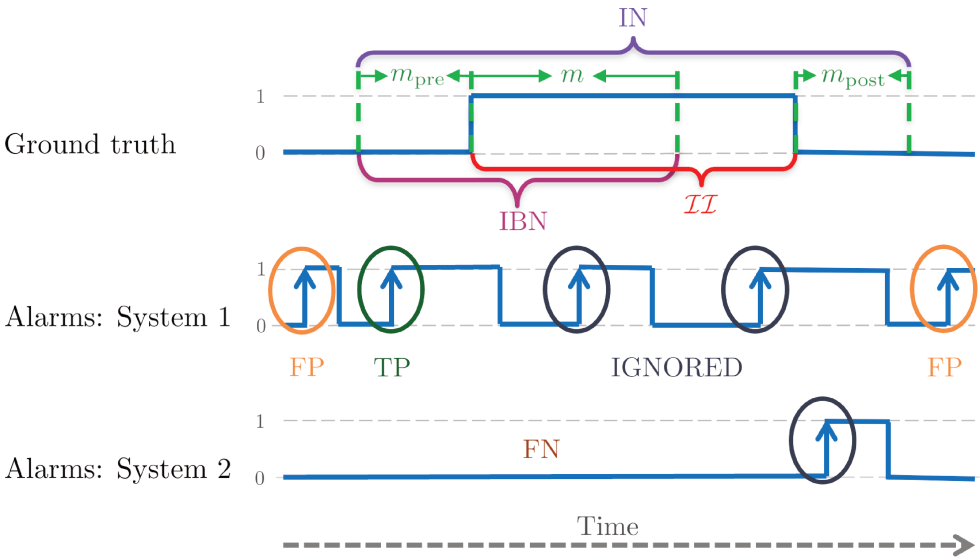



Figure 3.5: The top subfigure is an illustration of the definitions of edge-level evaluation terms on ground truth, with time in abscissa and non-intrusion (0) and intrusion (1) class for each frame of the video in ordinate. The next two subfigures represent examples of alarms and possible outcomes (TP, FP, FN) for two different PIDSs evaluated by the edge-level protocol.

The intrusion beginning neighborhood IBN is an interval comprising $m_{\text{pre}}$ frames before

and $m$ frames from the beginning of $\mathcal{II}$:

$$\text{IBN}(\mathcal{II}, m_{\text{pre}}, m) = [t_{\text{pre}}, t_{\text{m}}] \text{ s.t. } t_{\text{pre}} = t_{\text{start}}(\mathcal{II}) - m_{\text{pre}} \text{ and } t_{\text{m}} = t_{\text{start}}(\mathcal{II}) + m \ .$$

This interval highlights the importance of the initial frames of an $\mathcal{II}$, where an intruder has just entered the protected area, and it is in this interval where we ideally want the PIDS to raise an alarm. An alarm raised in IBN must be a TP.

For an $\mathcal{II}$ and alarms $\mathcal{O}(\mathcal{P})$, the possible outcomes at edge-level are defined as:

1. TP: if there is at least one alarm in IBN. For multiple alarms in IBN, only the first one is considered, and the rest are ignored.
2. FN: if there is no alarm in IBN.
3. FP: if an alarm is outside IN. Each alarm outside of IN is counted as an FP.

In this evaluation scheme, alarms lying outside IBN but inside IN are ignored. This means that we neither adversely penalize these alarms as an FP nor count them as a TP. In event-level evaluation, whether i-LIDS or this scheme, we do not define a true negative (TN). A TN is when a normal (non-intrusion) event is detected as such; in other words, how well we are classifying a normal event as normal. However, this is not the aim of intrusion detection; indeed, it is the opposite. Furthermore, the calculation of TN is ambiguous. We cannot generalize what length of the non-intrusion video should be considered as a TN. For example, a non-intrusion video clip of 5 minutes cannot be considered as similar to a non-intrusion video clip of 5 days.
These rules are for individual $\mathcal{II}$s, but how we deal with scenarios where the intrusion neighborhoods are so close that they intersect one another is another matter. If INs of two or more $\mathcal{II}$ intersect one another, then we merge them into a single IN. The new IN consists of $m_{\text{pre}}$ frames of the first $\mathcal{II}$ and $m_{\text{post}}$ frames of the last $\mathcal{II}$, and all the frames in between are merged as an $\mathcal{II}$. Algorithm 1 summarizes the protocol to evaluate a video at edge-level.

**Algorithm 1:** Edge-Level Evaluation of a PIDS

1 Initialize variables $m$, $m_{\text{pre}}$ and $m_{\text{post}}$.
2 Calculate IN for all $\mathcal{II}$s of the video.
3 If two or more INs intersect, merge them into a single expanded IN.
4 Calculate intrusion beginning neighbourhood IBN for each $\mathcal{II}$.
5 Obtain alarms $\mathcal{O}(\mathcal{P})$ from the PIDS.
6 Calculate TP, FN and FP.
7 Calculate precision, recall and other metrics.

## 3.5 Evaluation metrics

Since both PID and VAD tasks are finally a binary classification task, *i.e.,* to classify a frame or event as an intrusion / anomaly (positive) or otherwise (negative), we can naturally apply common metrics to obtain a quantitative performance measure. We describe below some widely used metrics in the literature. They are classified into two categories based on the threshold dependency.

### 3.5.1 Threshold dependent metrics

Most of the binary classification methods irrespective of application to VAD or PID task, produce a continuous output value within a pre-defined range. As an example, reconstruction-based VAD methods produce an output in $[0, 1]$ range, indicating the probability for an input to be anomalous (refer Section (2.5.2.1 and 3.2). To convert this continuous output value into discrete predictions $\{0, 1\}$, we need to choose a threshold. This threshold simply creates a boundary and values above and below it are converted to 1 and 0 respectively.

Given ground truth values, with these binary predictions, we can compute various metrics as defined below. They also work for methods where the output is not continuous but discrete values. When any system (PID or VAD) is deployed in a site, we need to set the threshold and therefore, these metrics are essential in a real-life online scenario. Concerning their use, the PID community always use these metrics for evaluation (Buch et Velastin, 2014; Vijverberg *et al.*, 2014; Kim *et al.*, 2018b), while in video anomaly detection, these are rarely used and instead the offline threshold independent metrics are used.

#### 3.5.1.1 Precision

The precision is the percentage of correctly predicted output out of the total predictions, as defined in the Equation (3.15). It is particularly useful when we want to measure how false alarms are affecting a VAD or PID system. A high value of precision denotes that we have very low false alarms.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \ . \tag{3.15}$$

#### 3.5.1.2 Recall

It is also known as sensitivity or true positive rate (TPR). It is the percentage of correct predictions out of the total predictions, as defined in the Equation (3.16). It is useful when the cost of the false negatives is high, *i.e.*, when we cannot afford to have omissions of intrusions or anomalies. Usually, in any system, the preference is to have the minimum omissions possible. This means that we need to ideally maximize recall.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \ . \tag{3.16}$$

#### 3.5.1.3 False positive rate (FPR)

It is also known as the false alarm rate. As shown in the Equation (3.17), it is the ratio between false alarms or false positives and total ground truth negatives (non-intrusion or non-anomaly in our case). True negatives (TN) have a major impact in FPR.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \ . \tag{3.17}$$

Concerning VAD and PID, true negatives refer to the case where normal frames or events (without anomaly or intrusion) are correctly detected. Both these tasks usually have abundance of normal frames since anomaly or intrusions are rare events. Therefore,

this metric is not very useful while evaluating systems for these tasks. However, popular VAD datasets like UCSD Pedestrian, CUHK Avenue, *etc.* have almost balanced testing set (see Section 1.4.1) . Thus, FPR is computed for them and it is used to calculate AUROC score (more details in Section 3.5.2.1).

#### 3.5.1.4 $F_\beta$ Score

To take into account both omissions (FN) and false alarms (FP), we need a way to combine precision and recall. The $F_\beta$ score combines them with a bias parameter $\beta$, which provides more or less importance to recall or precision.

$$F_\beta = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \ . \tag{3.18}$$

The most common values used for $\beta$ are 0.5, 1 and 2. With $\beta = 1$, we obtain the $F_1$ score. The $F_1$ score is the harmonic mean of precision and recall and it is widely used (Vijverberg *et al.*, 2013, 2014; Sehairi *et al.*, 2017). The choice of $\beta$ depends on the precision versus recall demand. For the i-LIDS dataset, there are two system roles with different bias values (i-LIDS Team, 2006). The roles are called 'Operational Alert' and 'Event Recording', with $\beta$ as 0.81 and 0.87, respectively (in (i-LIDS Team, 2006), $\alpha$ is used instead of $\beta$, where $\alpha = \beta^2$). The former role is designed for real-time intrusion detection and, therefore, has a lower $\beta$ value to give more importance to precision, as false alarms are essential here. The latter role is for non-real-time systems, where videos are recorded and analyzed on an offline basis. It has a higher $\beta$ value, as we cannot afford omissions in this case. Most PID systems (Buch et Velastin, 2008, 2014; Cermeño *et al.*, 2018) use this $F_\beta$ metric for evaluating on the i-LIDS dataset.

### 3.5.2 Threshold independent metrics

Often in threshold dependent metrics, it is not clear how the threshold is chosen (Saito et Rehmsmeier, 2015). Therefore, systems tend to choose a threshold to maximize the final score, *e.g.*, Buch et Velastin (2014) chose a high detection threshold to eliminate false alarms, as the metric used is $F_\beta$ with $\beta = 0.81$, which favors precision.

Alternatively, a system can be evaluated over a range of all possible thresholds, thus avoiding bias towards the choice of a fixed single threshold. We present below these types of threshold independent metrics. The main requirement of these metrics is that the system output should be continuous, and they are not adapted for discrete output systems. Contrary to threshold dependent metrics, these are not used in online system deployment but in offline system evaluation instead. These metrics are rarely used in PID community but are very prevalent for the video anomaly detection task (Kiran *et al.*, 2018; Nayak *et al.*, 2021; Ramachandra *et al.*, 2022).

#### 3.5.2.1 AUROC

It stands for area under the receiver operating characteristics (ROC) curve. The ROC curve shows the trade-off between sensitivity and specificity (Fawcett, 2006). It is created by plotting the TPR against the FPR, where each point corresponds to a threshold from the list of possible thresholds. This is depicted in Figure 3.6. In ROC curve, a classifier with random performance is shown via a straight diagonal line from (0, 0) to (1, 1), and this line is the baseline of ROC, with AUROC value of 0.5. The best possible value of

AUROC is 1, which signifies perfect classification. Since this score contains FPR, it is influenced by true negatives (refer Section 3.5.1.3). Therefore, AUROC is not suitable for imbalanced datasets (Saito et Rehmsmeier, 2015; Sokolova *et al.*, 2006). Since anomalies or intrusions are rare events, there are often very few anomalies or intrusions compared to the normal events. Therefore, theoretically AUROC should not be used for these tasks.

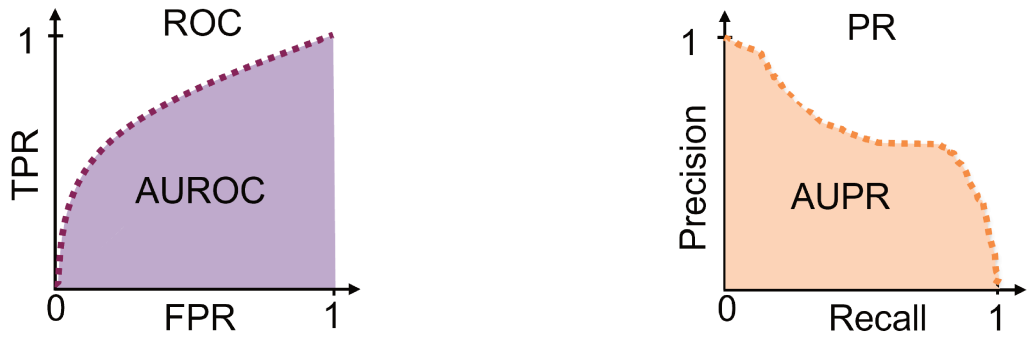



Figure 3.6: Illustration of ROC (left sub figure) and PR curve (right sub figure). Each point in dotted lines represent values ( (TPR, FPR) for ROC and (Precision, Recall) for PR) corresponding to a threshold. The area under the ROC or PR curve, *i.e.,* AUROC or AUPR is depicted with colored areas.

However, recent VAD datasets have balanced dataset for testing phase and therefore AUROC is primarily used there (Kiran *et al.*, 2018; Nayak *et al.*, 2021). Given the VAD test set, the ROC curve is obtained by varying the thresholds on the frame level predictions across the whole test set. It is called as "AUROC on all videos" or micro-averaged AUROC. However, some VAD works compute a "AUROC per video" and report the average, also called macro-averaged AUC (Georgescu *et al.*, 2021a,b; Ristea *et al.*, 2022). In this metric, the succession of thresholds to estimate the ROC curve is not common to all test videos. Since thresholds are adapted to each video, ROC curve is in risk to be overfitted, providing overly optimistic performances. Consequently, AUROC should always be measured as "AUROC on all videos", computed on the whole test set with thresholds common to all test videos concatenated together (Fawcett, 2006).

#### 3.5.2.2 AUPR

It refers to area under the precision-recall (PR) curve. The precision-recall plot shows precision values for corresponding sensitivity (recall) values. Like the ROC curve, it is model-wide evaluation with complete set of possible thresholds, as shown in Figure 3.6. The AUPR values range from 0 to 1, where 0 and 1 are for the worst and perfect classification, respectively. It is particularly useful when the datasets are highly imbalanced, such as in PID or VAD (Saito et Rehmsmeier, 2015). This is because it uses precision and recall which are focused on predictions of correct class (intrusion or anomaly), rather than that of incorrect class (normal class), *i.e.,* no dependency on true negatives unlike ROC. Concerning VAD, it has been used in some works (Kiran *et al.*, 2018) and it must be used if future datasets are imbalanced (which theoretically they should be).

## 3.6 Conclusion

In this chapter, we defined the two tasks that we tackle in this thesis, *i.e.,* the video anomaly detection and the perimeter intrusion detection. We provided proper formalism to both these tasks. This clear mathematical formulation is essential for properly evaluating works in these tasks. We explained the usual evaluation protocols and since they were not suitable according to the definition and task requirements, we provided a new evaluation protocol. We also studied various evaluation metrics and explained which ones are suitable for which tasks.

Concerning video anomaly detection, we are interested in detecting the occurrence of unusual appearance or motion attributes in unusual locations in the video. There are two principal ways to define an anomaly: either with some labeled anomaly examples during model training, or with only normal (non-anomaly) examples. The anomalous examples can provide weak, semi or full supervision. This first case of VAD with supervision is not practical for real-life use. The two main problems with it are: (1) since anomaly occur so rarely, it is practically impossible to have annotations for all possible future anomalous activities in any natural scene, and (2) even if all possible anomalous activities were available for supervision, the task itself would reduce to binary video classification where the anomalous class is "known". This vanquish the spirit of video anomaly detection where the ultimate goal in practice is to detect any deviation from normality. In contrary to this, in the second case, the normality is learned from the data. We have data with only normal scenes in the case. Hence, the main idea is to understand normality from this data and then the distribution outside this normality is defined as the anomaly. We propose a formalism for this using the PAC learning framework, which takes into account the complicated cases where normality and abnormality are conceptually very close to each other. For VAD evaluation, frame-level evaluation is followed, therefore the concept of anomalous events is neglected. Furthermore, the threshold-independent metrics are used instead of online metrics. In short, frame-level AUROC metric is used as an evaluation measure in most VAD systems.

Perimeter intrusion detection is a special sub-task of VAD with several specificities like perimeter, intruder object type, movement constraints, *etc.* It has never been systematically defined before. We first mathematically define how an object can cause an intrusion event, followed by the definition of intrusion intervals in the video. We proposed that the task of perimeter intrusion detection is basically to detect the beginning of these intrusion intervals. Since these intrusion intervals are nothing but events, so we first explained the existing i-LIDS evaluation protocol. We found several drawbacks in this evaluation scheme. Therefore, we proposed a new evaluation protocol for PIDS, called edge level evaluation. It is completely coherent with the PID definition and basically focuses on evaluating a PIDS based on the rising edge from non-intrusion to intrusion state. In other words, this protocol helps evaluating if an intrusion was detected within the first few seconds of its beginning. Finally, similar to i-LIDS evaluation protocol, the threshold-dependent metrics like precision, recall and $F_1$ score can be used with the proposed edge level evaluation.

# Chapter 4

# Unsupervised autoencoder and adaptive detection

In this chapter, we introduce an unsupervised perimeter intrusion detector. We first introduce how we learn spatio-temporal normality from videos with an end-to-end trainable neural network, followed by how intrusions are detected using an adaptive mechanism that handles long-term variations in scene dynamics. We then provide a thorough comparative analysis with other methods on i-LIDS dataset, using different evaluation protocols. Part of this work was published in IEEE International Conference on Image Processing (ICIP) 2022 (Lohani *et al.*, 2022a).

## Contents

## 4.1 Introduction

In this chapter, we introduce an unsupervised deep learning method for perimeter intruder detection. As already stated in previous chapters, intrusions are rare by nature and therefore it is very difficult to have large amounts of annotated examples for training a supervised model. We saw in the review of existing PID methods (refer to Section 2.6) that most methods do not use deep learning. The ones that use deep learning rely on externally trained supervised object detectors and thus their functioning is biased on the external dataset, and furthermore they assume that intruder object class is known beforehand. Therefore, there is a need of methods which learn from the data itself, without requiring need of annotation or external dataset. We propose such an approach in this chapter.

Inspired from the methods of video anomaly detection, this work is based on video clip reconstruction proxy task for PID. The same principle as VAD is followed here: learning normality while training and detecting intrusions as deviations from normality. One major challenge for any PIDS is to adapt to changing weather, light, and environmental conditions. Standard reconstruction based methods are sensitive to these dynamic conditions (Chalapathy et Chawla, 2019; Kiran *et al.*, 2018). To address this limitation, we propose an adaptive strategy so that the presented unsupervised PIDS works well irrespective of scene dynamics.

In the rest of the chapter, we present our proposed method and provide an extensive comparative analysis of it with other methods.

## 4.2 Unsupervised and Adaptive Perimeter Intrusion Detector

In this section, we present our proposed AE-Adapt (AutoEncoder with Adaptive thresholding) method. It is the first unsupervised deep learning approach for perimeter intrusion detection. It has two main steps: learning spatio-temporal normality using an autoencoder and detecting intrusions using this learned representation via an adaptive mechanism. These two steps are described in following subsections.

Before presenting them, we must define the input to our method. We use video clip as the input unit. We mathematically define it as follows. Given a video $\mathcal{V}$ with $n$ frames $\{I_1, I_2, \ldots, I_n\}$, a video clip $V_{l,s}$ of length $l$ and temporal gap $s$ between frames is defined as:

$$V_{l,s} = \left\{I_1, I_{1+s}, \ldots, I_{1+(l-1)s}\right\} = \{I_{1+xs}\}_{0 \leq x < l} \ , \tag{4.1}$$

where for simplicity, it is assumed that the clip starts from the $1^{st}$ frame.

### 4.2.1 Learning spatio-temporal normality

We propose a strided 3D convolutional autoencoder (S3DCAE) as autoencoder for our PIDS as shown in Figure 4.1. Given a video, we use a sliding window to extract video clips. Each video clip is fed to the S3DCAE, which reconstructs it. The idea is to train the autoencoder with only normal videos (without intrusions) and minimize the error between input and reconstructed video clips. The intuition is that the trained autoencoder should correctly reconstruct normal video clips and badly reconstruct the video clips with intrusions. In other words, it will learn the normality and reconstruction error between the input and reconstructed clip will help to distinguish between non-intrusion and intrusion.

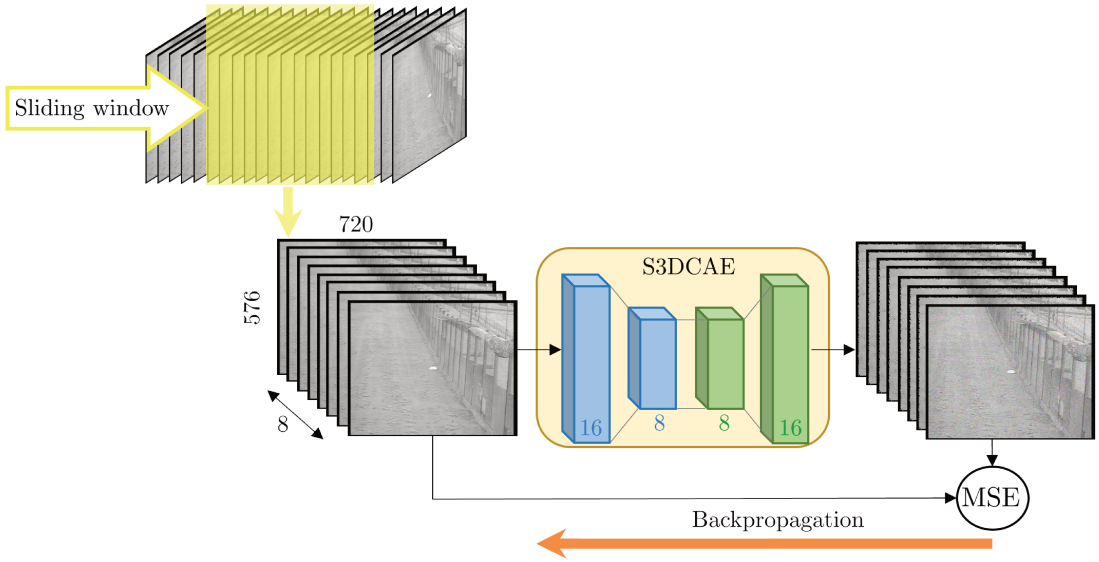



Figure 4.1: Learning normality: A sliding window (in yellow) extracts video clips of fixed length. Each video clip is fed to S3DCAE, which reconstructs it. The mean squared error (MSE) between input and reconstructed clip is backpropagated to update learnable parameters of the autoencoder.

#### 4.2.1.1 S3DCAE architecture

The main components of this architecture are 3D convolutions and 3D deconvolutions. They were used since they jointly model spatio-temporal information and are better suited for feature learning from videos than 2D convolutions, 2D convolutions with LSTM, *etc.* (Section 2.3).

Let us denote the input as $X$, where $X$ is a video clip of length $T$ and temporal gap of 1, *i.e.*, $X = V_{l=T,s=1} = \{I_1, I_2, \ldots, I_T\}$. The proposed autoencoder takes $X$ and outputs a reconstructed video clip $\hat{X}$ of same dimensions. The error between these clips is the training loss function (MSE), defined as:

$$\mathcal{L} = \frac{1}{T \times D \times H \times W} \left\| \hat{X} - X \right\|_F^2 , \tag{4.2}$$

where $D$, $H$ and $W$ denotes channels (depth), height and width of each frame and $\| \cdot \|_F$ denotes the Frobenius norm.

Figure 4.2 shows the architecture of S3DCAE. Each encoder layer consists of a strided 3D convolution while each decoder layer uses a strided 3D deconvolution (Zeiler *et al.*, 2010), with kernel size of $5 \times 3 \times 3$ (shape : temporal length, height, width). A stride of $2 \times 2 \times 2$ is used in each operation except the last 3D deconvolution where $1 \times 1 \times 1$ stride is used. This signifies that each encoder layer reduces the input volume into half, while each decoder layer (except the last) does the opposite. Encoder is composed of two layers containing (16, 8) filters, while decoder also have two layers with (8, 16) filters respectively. The ReLU activation function is used between each layer, except the final layer, which uses the hyperbolic tangent (*tanh*) activation. A dropout layer with dropout probability of 0.25 is applied after the first layer. Overall, we have a light architecture with only 15, 889 parameters (1011 MB size). Finally, this model is trained using the Adadelta optimizer (Zeiler, 2012).

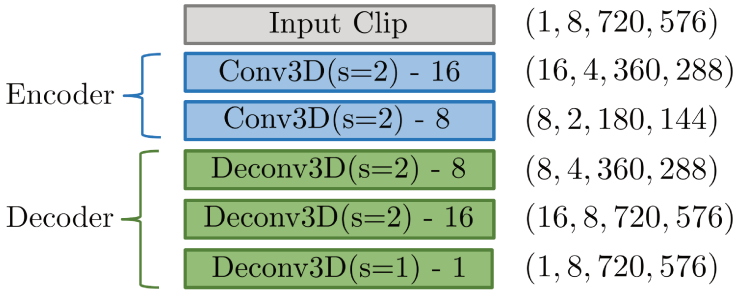



Figure 4.2: S3DCAE architecture for PID with encoder and decoder. Conv3D and Deconv3D refer to 3D convolution and 3D deconvolution respectively, where $s$ is stride, and the number that follows − denotes number of filters. The rightmost column refers to the data shape (channels, temporal length, height, width).

### 4.2.2 Detecting Intrusions

Once we have a trained model, we use it during testing phase with the hypothesis that the intrusion frames will be badly reconstructed, *i.e.*, with a high reconstruction error. Figure 4.3 illustrates how intrusions are detected in our system. A video clip is obtained using a sliding window, the learned S3DCAE takes it as input and reconstructs it. Then, the frame level reconstruction error is computed using the input and reconstructed clip. This reconstruction error is used by an adaptive strategy with moving z-score to produce an intrusion alarm. These steps are explained in detail below.

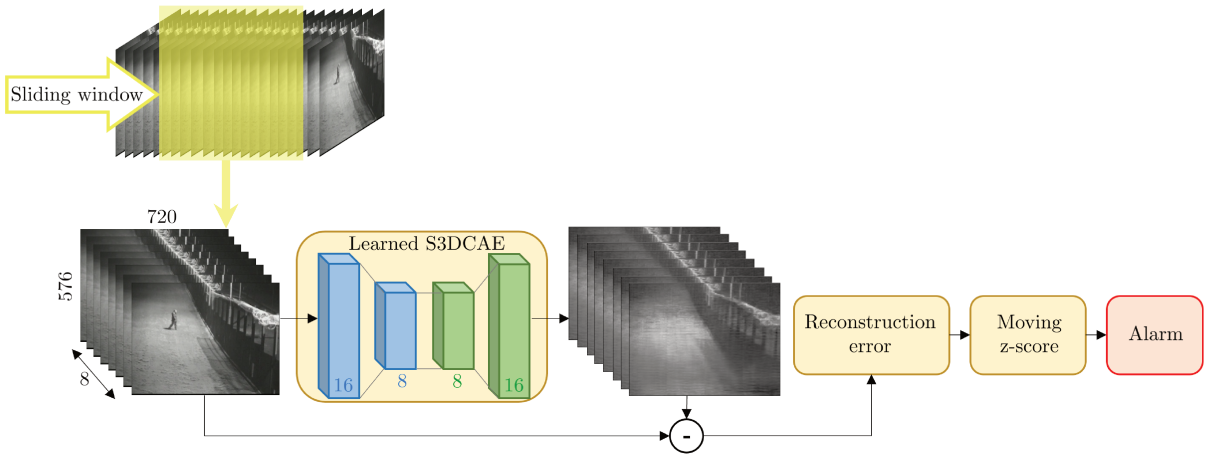



Figure 4.3: Detecting intrusion using AE-Adapt: an input video clip is reconstructed after going through the S3DCAE. The reconstruction error is calculated between these clips and fed to a moving z-score module, the output of which is thresholded to raise an intrusion alarm.

#### 4.2.2.1 Frame level reconstruction error

From a test video, we obtain video clips using the sliding window like training. A test video clip of length $T$ frames and temporal gap $s = 1$, *i.e.*, $\{I_1, I_2, \ldots, I_T\}$ goes through S3DCAE to produce a reconstructed clip $\{\hat{I}_1, \hat{I}_2, \ldots, \hat{I}_T\}$. The reconstruction error can then be calculated as:

$$r = \frac{1}{T \times D \times H \times W} \sum_{k=1}^{T} \left\| \hat{I}_k - I_k \right\|_F^2 . \tag{4.3}$$

For real-time applications, we need a reconstruction error for each frame that the system encounters. Our method is designed for video clip level and not frame level. The video clips of length $T$ are formed with temporal gap of one between each video clip. This means that if we have a video with $n$ frames $\{I_1, I_2, \ldots, I_n\}$ and $T = 8$, the first video clip is $\{I_1, I_2, \ldots, I_8\}$, the second video clip is $\{I_2, I_3, \ldots, I_9\}$, *etc.* Since each video clip provides a single scalar value of reconstruction error (see Equation 4.3). To convert this video clip level reconstruction error to frame level error, we assume that the value of each video clip reconstruction error corresponds to the last frame of the video clip. This implies that for first $T-1$ frames of a video, there are no frame level reconstruction errors, *i.e.,* $r_x$ with $x \geq T$.

#### 4.2.2.2 Adaptive thresholding using moving z-score

A PIDS installed in a site records videos for many days. This long video sequence has very high scene dynamics with changes in light, weather, temperature, scene background, etc. Since reconstruction error changes with scene dynamics, naively thresholding it can provide undesirable outcomes (Ribeiro *et al.*, 2018). To resolve this issue, most work using unsupervised autoencoder try to standardize the per-frame reconstruction error for each video (Ribeiro *et al.*, 2018; Ramachandra *et al.*, 2022). They usually use the min-max (MM) rescaling which force the error values to be in the [0,1] range, where 0 and 1 represents normal and intrusion case, respectively. Thresholding on this rescaled reconstruction error is dangerous because this standardization will force the lower values of reconstruction errors (*e.g.*, values of night) to be considered as normal (since forced to be near 0), even if there is an intrusion.

To address this issue of adjusting to scene dynamics, we propose an adaptive mechanism which follows the reconstruction error along time and trigger an alarm as soon as it deviates from the normal behavior and then continue adapting to the new values. It is also well aligned with perimeter intrusion detection definition, where the goal is to detect just the beginning of intrusions and not every intrusion frame in the video. We compute the moving z-score (MZ) of reconstruction errors to provide a temporal standardization of values (Barthélemy *et al.*, 2019). Once standardized, values can be easily compared to a fixed threshold $z_{th}$. Using frame level reconstruction errors for first $t$ frames, *i.e.,* $\{r_1, r_2, \ldots, r_t\}$, the system is initialized by computing the mean and standard deviation of them, denoted as $\mu_t$ and $\sigma_t$, respectively. For the $(t+1)^{th}$ frame, z-score of reconstruction error $r_{t+1}$ is computed as:

$$z_{t+1} = \frac{r_{t+1} - \mu_t}{\sigma_t} \ , \tag{4.4}$$

If this value is greater than a fixed chosen threshold, *i.e.,* $z_{t+1} \geq z_{th}$, the system raises an intrusion alarm. Regardless of whether an alarm is raised or not, the moving mean and standard deviation are updated as:

$$\begin{aligned} \mu_{t+1} &= \alpha \ r_{t+1} + (1-\alpha) \ \mu_t \\ \sigma_{t+1} &= \sqrt{\alpha \ \left(r_{t+1} - \mu_{t+1}\right)^2 + (1-\alpha) \ \sigma_t^2} \ , \end{aligned} \tag{4.5}$$

where $\alpha \in [0, 1]$ defines the speed of the exponential update. If $\alpha$ is close to 1, then it implies that the system adapts itself quickly with the current value of reconstruction error, without taking much into account the evolution of error. Contrary to this, when $\alpha$

is close to 0, the system relies on evolution of error historically and the current value of error plays a smaller role. This process with Equation (4.4) and Equation (4.5) is repeated for each new frame of the video.

## 4.3 Experiments and results

In this section, we first describe our experimental setup with data, methods and evaluation protocols. Next, we present various results comparing our approach different existing PID systems.

### 4.3.1 Implementation details

Here, we present the data used in experiments, the implementation details of our approach and other methods, and finally the evaluation protocols used.

#### 4.3.1.1 Data

We used the two cameras views of i-LIDS sterile zone dataset as described in Table 1.2. The frame resolution of i-LIDS dataset is 720x576. The training set contains 10 non-intrusion videos per view. The testing set contains 17 and 16 videos of view 1 and view 2, with 7 and 6 videos containing intrusions respectively (from 36 to 92 minutes in length). For more details on i-LIDS dataset, please refer to Section 1.4.2.

#### 4.3.1.2 Methods

To compare our method with others using different evaluation protocols, we need their source code. From the list of PID methods (refer Table 2.2), only one method, *i.e.,* Nayak *et al.* (2019), provides an implementation of their method. Therefore, we select it for our PIDS comparison. Since no classical PIDS (non-deep learning based) provides source code, we use GOFPID (Barthélemy, 2022), which allows to build the traditional PIDS pipeline. With GOFPID, we can have a representative method for the classical PIDS, and thus we include it in our experiments.

The method of Nayak *et al.* (2019) follows the typical PIDS pipeline (see Figure 2.21), along with some deep-learning-based components. Given a video, frames are extracted and the user is asked to draw a perimeter in the frame for protection. Then, a potential intruder object should be chosen from a list of pre-defined classes, such as person, car, *etc.* The system detects the intruder object with bounding box using the pre-trained YOLO v2 network (Redmon et Farhadi, 2017), having a detection threshold of 0.25 for class human. If the detected intruder object is inside the pre-defined perimeter, then it is considered as an intrusion. Finally, the object is tracked using simple online and real-time tracking (SORT) algorithm (Bewley *et al.*, 2016). For testing it in i-LIDS, we first drew a protection perimeter following the fences for each view. We then chose person as an intruder class. Then, frames of each video are fed into the system, and we obtained a binary intrusion/non-intrusion prediction for each frame.

GOFPID (Barthélemy, 2022) helps to define the traditional PIDS pipeline. It also provides suitable configuration for the i-LIDS dataset. We use this default i-LIDS configuration as the classical PIDS method, and it can be described as follows. First, the user is asked to draw a perimeter in the first frame of the video. Along with perimeter, the

perspective size of the intruder object (here person) at different distances from the camera is also requested. We use the same perimeter like last method and use the perspective settings provided for i-LIDS with GOFPID. Given these settings, the input video frame is first denoised by spatial blurring using a Gaussian filter. Then, the foreground detection is performed using an improved adaptive Gaussian mixture model (Zivkovic, 2004). The obtained foreground mask is then denoised by mathematical morphology. The created foreground blob is then checked for two conditions: whether it is inside the perimeter and if it obeys the minimum object size according to the perspective setting. If these conditions are met, then the blob is tracked using blob tracking. If the blob is tracked for a minimum of 3 frames and it displaces for a minimum distance (set beforehand), then an intrusion alarm is raised. Testing this method on i-LIDS dataset was straightforward and a binary intrusion/non-intrusion output is obtained for each frame.

Concerning the proposed AE-Adapt method, we trained and tested our model in each view (view 1 and view 2) of i-LIDS dataset. Test uses the same protection perimeter as defined for the last two methods, signifying that all activities (intrusion or otherwise) outside this perimeter are ignored. Non-intrusion videos from training set of each view are used for training, using one Nvidia RTX 3090 GPU, with a batch size of 32. Each frame is converted to grayscale, pre-processed using histogram equalization and pixels were rescaled to $[-1, 1]$. For each video, input video clip is constructed using 8 frames with temporal stride of one frame, leading to an input of shape $8 \times 720 \times 576 \times 1$. Adaptive thresholding is initialized with reconstruction errors of first 10 frames, moving z-score is used with $z_{th} = 4.5$ and $\alpha = 0.01$, chosen from the validation set.

#### 4.3.1.3 Evaluation Protocols

We used the three evaluation protocols studied in Section 3.4: frame-level (FL), i-LIDS and edge-level (EL) evaluations. For edge-level evaluation, we set the tolerance variables $m_{\text{pre}}$ and $m_{\text{post}}$ to 3, *i.e.*, less than 1 second of tolerance. For variable $m$, the following values were chosen: 5, 10 and 50. This signifies that we tested the methods to detect within 1, 2 and 10 seconds from the beginning of intrusion (refer Section 3.4.3). The parameters for 1 and 2 seconds were selected to satisfy the demands of a real-time PIDS, whereas the parameter for 10 second was chosen in order to be comparable with i-LIDS evaluation protocol. For each evaluation protocol, we present the results in terms of precision, recall and $F_1$ score as defined in Section 3.5.

### 4.3.2 Comparison of Evaluation Protocols for PIDS

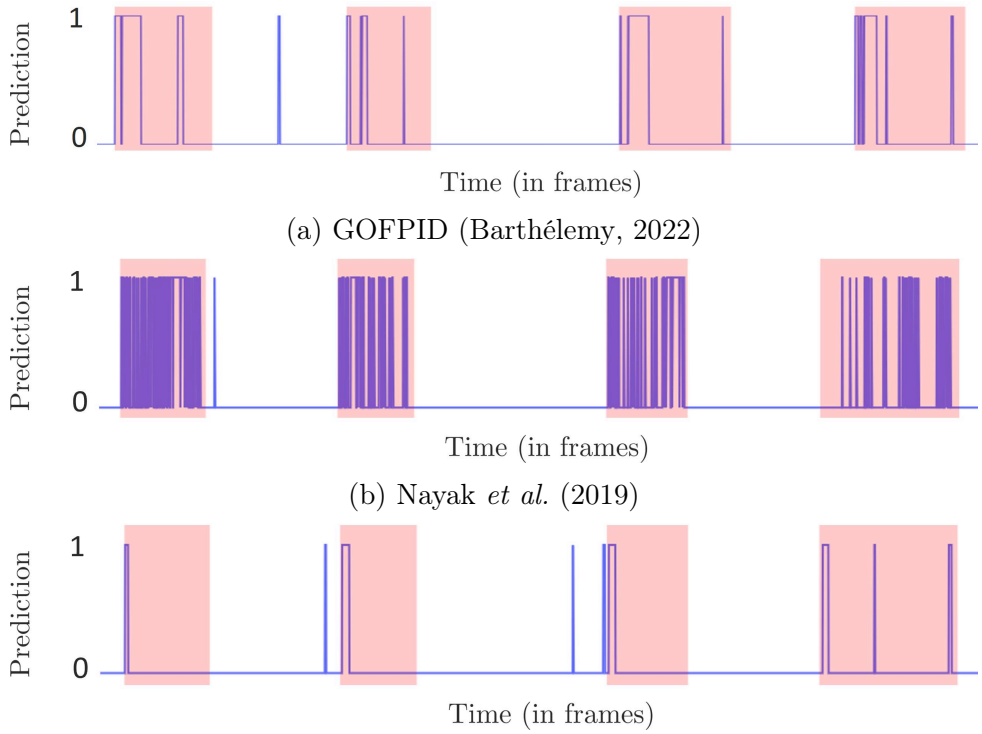
Here, we present overall results of different methods on view 1 and view 2 of i-LIDS dataset, using different evaluation protocols.

#### 4.3.2.1 Results concerning View 1

Figure 4.5 shows the results of the three methods on view 1 of the i-LIDS test dataset through different evaluation protocols. To better understand results, we must first comprehend how these methods predict intrusions. Figure 4.4 shows predictions using these methods on portions of the video taken from the i-LIDS test dataset.

Observe how each method differs in intrusion detection behavior. It can be observed that method of Nayak *et al.* (2019) has the highest number of correct frame predictions, followed by GOFPID and our method has the least frame predictions. Similarly, the

same trend follows for omissions, since Nayak *et al.* (2019) leaves least number of frames undetected and our method detects only few frames per intrusion. Both these observations explain the frame level recall scores of the three methods as seen in Figure 4.5. All the three methods have few false detections as seen in Figure 4.4, but the ratio of correctly predicted frames to false detections is highest in Nayak *et al.* (2019), followed by GOFPID and finally our proposed method. This explain the frame level precision scores of the three methods, with Nayak *et al.* (2019) having the best score. Overall, this leads to the frame level $F_1$ scores, as shown in the left subfigure of Figure 4.5.




(a) GOFPID (Barthélemy, 2022)

(b) Nayak *et al.* (2019)

(c) Proposed method: AE-Adapt

Figure 4.4: Example of per-frame predictions of the three methods on portions of video taken from i-LIDS test dataset. The intrusion intervals (ground truth) are shown in light red strips, the abscissa represents frames and ordinate shows prediction, where 1 signifies intrusion and 0 otherwise.

The i-LIDS evaluation protocol functions on event level and concerns with intrusion alarms rather than frame-level binary prediction. To recall, an intrusion alarm is raised as soon as a system changes its state from 0 to 1. For the i-LIDS evaluation protocol, if there is at least one alarm in the first 10 seconds of intrusion (ground truth), it is counted as a true positive (TP). However, an alarm after 10 seconds is counted as a false positive (FP), as already explained in Section 3.4.2. It can be observed that Nayak *et al.* (2019) has a high number of intrusion alarms from the beginning to end of each intrusion interval. These alarms at the end are considered as false detections, even though they happen during the intrusion. This explains their poor precision score in the i-LIDS evaluation protocol as shown in Figure 4.5. Since this method rarely misses producing alarms within the first 10 seconds, it has a very low number of false negatives (FNs), and this is depicted with a high recall score. For GOFPID, we can observe in Figure 4.4 that it produces alarms in beginning of almost all intrusion intervals, this leads to its excellent

recall score in i-LIDS evaluation protocol (see Figure 4.5). But this method also have alarms at the end of intrusion intervals and even outside them (see Figure 4.4), and these are counted as FP, which is aligned with their poor precision score. Our method has a lower number of alarms, and most of them occur at the beginning of intrusions as shown in Figure 4.4. This leads to a smaller FP due to fewer late predictions, and we have a good precision value as seen in Figure 4.5. We also have few omissions within the first 10 seconds; therefore, we observe a good recall value in terms of i-LIDS evaluation.

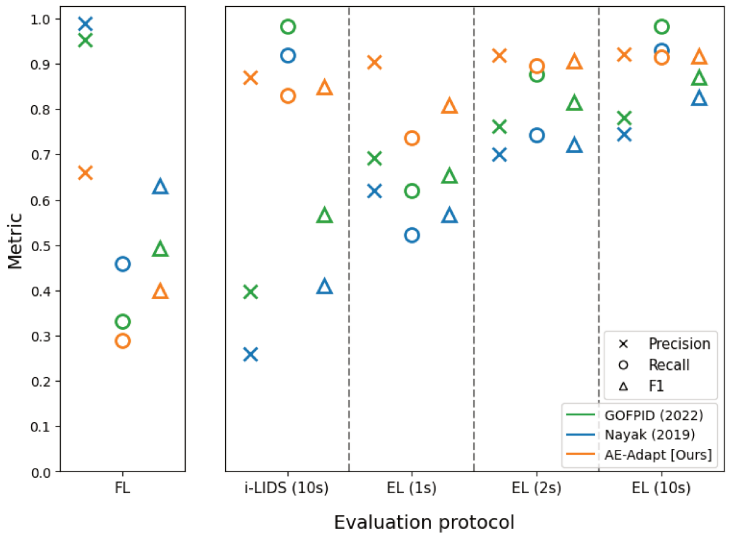



Figure 4.5: The results on view 1 of i-LIDS test set for three methods: GOFPID (Barthélemy, 2022) (in green), Nayak *et al.* (2019) (in blue) and our method (in orange). The abscissa represents three evaluation protocols: frame-level (FL) (on left subfigure), i-LIDS and edge-level (EL) (on right subfigure). i-LIDS is evaluated by default for 10 s whereas, for EL, we show results for 1, 2 and 10 s. The ordinate represents values of three metrics: precision, recall and $F_1$ score.

In the edge-level evaluation protocol (Figure 4.5), if an alarm is not raised within the first $m$ seconds of an intrusion, then it is considered as an omission or false negative. But alarms raised within the intrusion, even though after $m$ seconds from beginning are not counted as FP like i-LIDS evaluation protocol. Since it is a difficult task to raise the alarm in $1^{st}$ second of an intrusion in comparison to $2^{nd}$ or $10^{th}$ second, we observe an increase in recall for all methods from EL (1 s) to EL (10 s). Nayak *et al.* (2019) and GOFPID have a lower recall than our method in EL (1 s) and EL (2 s) because they often raise alarms later than 2 seconds. Regarding precision, here, the excessive late alarms of these two methods are not penalized as FPs. Therefore, in contrast to i-LIDS evaluation, we observe a better precision value for Nayak *et al.* (2019) and GOFPID in edge-level evaluation. Our method raises fewer false alarms than both of them, thus, has a better precision in edge-level evaluation.

Overall, we observe that predictions of Nayak *et al.* (2019) fluctuates frequently between non-intrusion and intrusion prediction, thus producing a high number of alarms. This is because it is focused on human detection and it often loses track of the human from one frame to another, and then re-detects it, causing more intrusion alarms. Furthermore, it has very few predictions outside intrusions. That is why it has a good precision and recall value at frame-level. However, for i-LIDS and edge-level, we observe that their method has a low precision value. This is because, from this high number of alarms, only few are

relevant, *i.e.,* the ones at the beginning of intrusions. Other alarms are either treated as false alarms or discarded depending on the protocol. The aim of intrusion detection is to detect the intrusion as soon as possible and not just to classify each frame; therefore, it is important to use the right evaluation protocol that emphasizes these characteristics.

Unlike the method of Nayak *et al.* (2019), GOFPID does not have depend on person detection for PIDS. It first detects blobs. The blobs are then filtered out with the perspective size of intruder objects. The resultant blobs are tracked irrespective of the blob information, *i.e.*, whether it is an intruder blob (human for example), or a non-intruder blob (camera noise or insects). An intrusion alarm is raised as soon as tracked blobs have minimum displacement. When an actual intruder enters the scene in i-LIDS dataset, it is moving and thus the alarm is raised by this system. After some time, either the intrusion stops (as there is a fence) or blob is too small, thus leading to no alarm state. Finally, often the intrusion starts moving again (like for going on other side of fence), which causes another alarm as seen in Figure 4.4. There are a few alarms which are raised not for intrusion, for example, when there is sudden change in light in the scene, or due to climatic conditions like snow. This explains the high precision value and low recall value in frame level. But in reality, the intrusion alarms were raised in beginning of intrusions and both i-LIDS and edge-level protocols captured it with high recall values. The i-LIDS evaluation highly penalized alarms end of intrusion alarms (leading to poor precision score), while edge-level evaluation addressed this issue. Thus, edge-level protocol helped in correctly evaluating this system

For our method, it can be observed that we have a lower number of alarms. It focuses more on the beginning of intrusion events and, due to the z-score, it quickly adapts itself with the scene dynamics. Since only beginning of intrusion are detected, we have a very poor frame-level recall. Our method is sensitive to flickering light or moving light conditions, therefore it has some false alarms. The relation between few correctly predicted frames and some false alarm frames leads to poor frame level precision. Therefore, we have a poorer frame level score. However, in both i-LIDS and at edge-level, we obtain a good score because the proposed method raises the alarm as soon as the intrusion occurs, as it is expected from a PIDS.

#### 4.3.2.2 Results concerning View 2

The results of view 2 are shown in Figure 4.6. The GOFPID method follows has similar results like in View 1. The recall values are slightly decreased in all protocols because few difficult intrusions were not detected like person rolling far away from camera with same clothes as background. The precision values have slightly increases in event level metrics because there is no camera switch from color to monochrome (since only monochrome videos) and there are less distractions like animals *etc.* Therefore, the GOFPID method detects lesser number of irrelevant blobs as intrusions here. Thus, it obtains overall better results than View 1. We can observe that in all evaluation protocols, our method has a similar trend of results as in view 1, but with a slight decrease in metric values. This slight decrease in values is due to an increase in FPs and FNs, which is linked to more instances of light fluctuation and intrusions at a far distance. Nayak *et al.* (2019) obtain similar results to view 1 in frame-level and i-LIDS-level evaluation. In edge-level evaluation, we observe a perfect precision score of 1. This is because this method did not predict any false detection outside the intrusion interval in view 2. This perfect precision score augments their overall score of in edge-level evaluation. This important detail of no false detections was not captured by the i-LIDS-level protocol, and we observe a poor value of precision

for them.

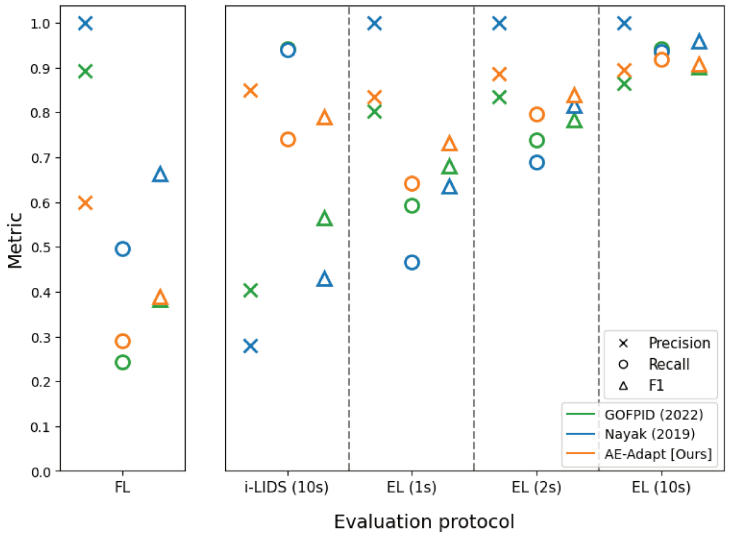



Figure 4.6: The results on view 2 of i-LIDS test set for three methods: GOFPID (Barthélemy, 2022) (in green), Nayak *et al.* (2019) (in blue) and our method (in orange). The abscissa represents three evaluation protocols: frame-level (FL) (on left subfigure), i-LIDS and edge-level (EL) (on right subfigure). i-LIDS is evaluated by default for 10 s whereas, for EL, we show results for 1, 2 and 10 s. The ordinate represents values of three metrics: precision, recall and $F_1$ score.

Overall from this subsection, we can conclude that the edge-level protocol is the most suited for evaluating any PIDS. It can be seen as an evolved version of i-LIDS evaluation protocol and thus adheres to real-life constraints, like detecting in beginning of intrusion event. Furthermore, using its parameters, we can evaluate at different lengths from the beginning of intrusions, like 1 second, 2 seconds, *etc.* Concerning compared methods, we found that their performances were very close to one another in edge-level evaluation. The GOFPID method is representative of the classical PIDS. After extracting blob, it filters out objects with irregular perspective size and tracks the resultant blobs for atleast three frames and checks whether the blob moves a minimum distance. These well formed post-processing operations allows it to filter many irrelevant blobs of objects. Since this dataset has only person as intruder without too much distractions, this method performs very well to detect intruders with a high precision. It only fails in complicated cases where the object is too far and too similar with the background. Some false alarms are caused due to fast moving light in the scene or due to snow. The method of Nayak *et al.* (2019) uses pre-trained person detector. Since this dataset has only person as intruder, this method works very well. It rarely detects noise due to snow or lighting conditions as intrusions, thus it has a very good precision value in edge-level evaluation. When intruder enters the scene, it detects it but then looses it in tracking and have re-detections. Also, it has problem in detecting when the person in crawling or log-rolling in the far away distance. It detects most easily when the person is standing. Due to this, it often have intrusion alarms late, around the end of intrusion, and this leads to a lower recall value for this method. Finally, our unsupervised and adaptive PID system performed equally well in comparison to last two methods. The false alarms are mainly caused due to changing light conditions and insects on camera. But these are very few in number, thus we have an overall good precision score in edge-level evaluation. Our method detects intrusions

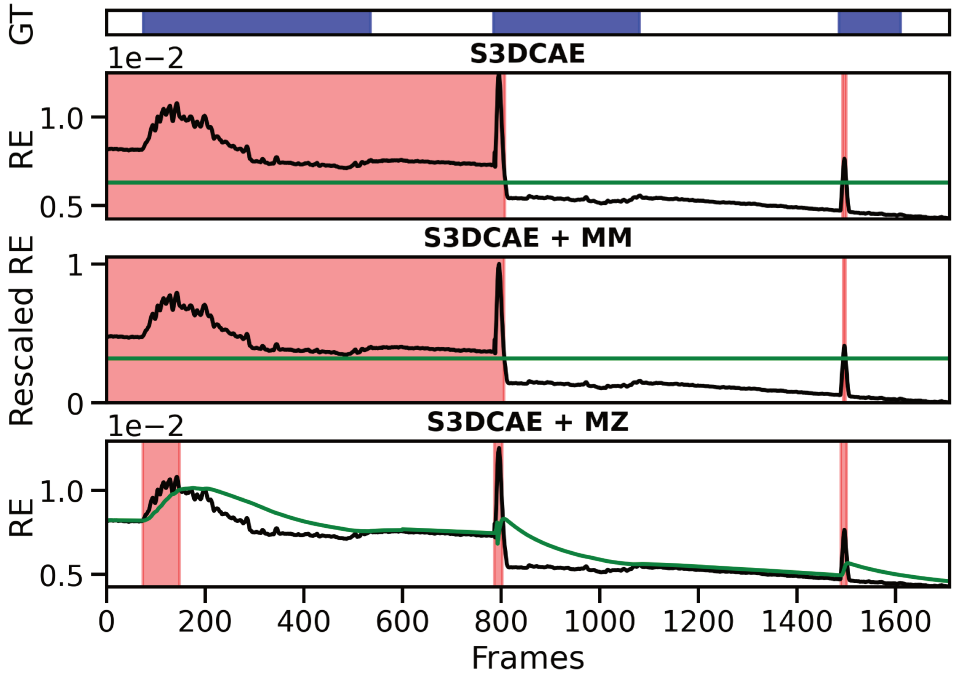



Figure 4.7: Qualitative comparison of thresholding strategies on a video, with ground truth (GT) in blue, thresholds in green and alarms in red. Fixed threshold on reconstruction error of S3DCAE (top); fixed threshold on min-max (MM) rescaled reconstruction error (middle); adaptive threshold by moving z-score (MZ) of reconstruction error (bottom).

and adapts itself, we rarely miss an intrusion event, therefore we have a very good recall value. The missed intrusions include cases where intruders are very far from the camera, having clothes similar to scene background. Our method has still a room for improvement and we will present a new method in the next chapter. Since i-LIDS dataset is not very difficult, having only person as intruder, we also test methods on a private dataset which is detailed in annex Section A.

### 4.3.3 Effect of adaptive thresholding

In this subsection, we explore the impact of adaptive thresholding in the proposed method. Figure 4.7 shows the impact of various thresholding strategies. We can observe that the reconstruction error varies a lot from one intrusion to another. Therefore, trying to select an optimal threshold on it can lead to unsatisfactory results, *e.g.,* false alarms before and after $1^{st}$ intrusion. After min-max rescaling, we obtain a similar conclusion as only the range of values have changed to $[0, 1]$, without any significant difference in threshold choosing strategy. We can see that the moving z-score adjusts itself with the reconstruction error. It detects the beginning of each intrusion and then updates itself.

In Table 4.1, we quantitatively compare the thresholding strategies on proposed S3DCAE. We observe that the direct outcome from the autoencoder leads to a poor performance. As expected, it is difficult to choose a threshold when the error varies due to scene dynamics. Rescaled reconstruction error using min-max (MM) scheme improves the overall result by only 5%. Since i-LIDS test-set videos are very long, the rescaling scheme failed to handle changes in reconstruction error caused by changing scene dynamics. We can

| Methods | Precision | Recall | $F_1$ |
|---|---|---|---|
| S3DCAE | 0.54 | 0.44 | 0.49 |
| S3DCAE + MM | 0.49 | 0.59 | 0.54 |
| S3DCAE + MZ | **0.92** | **0.91** | **0.92** |

Table 4.1: Quantitative comparison of thresholding strategies, on View 1 of i-LIDS dataset. S3DCAE stands for proposed autoencoder with frame reconstruction error, MM for min-max scaling, and MZ for moving z-score. EL (10 s) is used as evaluation protocol.

clearly observe that adding an adaptive thresholding with moving z-score (MZ) almost doubles the overall performance from $F_1$ of 0.49 to 0.92. These results strongly support our proposition that an adaptive component is necessary for the deployment of a PIDS in real-life scenes.

### 4.3.4 Working illustration of AE-Adapt

In this subsection, we demonstrate the working of our proposed method, shown in Figure 4.8. For ease of comprehension, we continue with the example of Figure 4.7.

Since it is continuously snowing (distraction) in the scene, there are small zigzag effects in reconstruction error. We can observe in images that the intruders are at different distances from camera and the magnitude of reconstruction error depends on it. For instance, the intruders far from camera occupy a smaller area in the image and thus have smaller reconstruction error. We also observe a high local RE in the beginning of intrusions (see images in top row). This is because when an intruder enters the scene, there is a high spatio-temporal change, which for autoencoder is abnormal (normal training videos in i-LIDS dataset have extremely low number of examples for where objects move) and thus it is badly reconstructed. Two images in bottom row show that even though the intruder is present in the scene, the reconstruction error is relatively lower. The reason behind is the same, *i.e.,* spatio-temporal change. The spatio-temporal change is comparatively lower in these cases, since the intruders are hardly moving (trying to break in). Finally, we can observe via the green curve that the moving z-score just detects the beginning of these intrusions and then adapts itself with the reconstruction error. This is what we expect ideally from a PIDS.

## 4.4 Conclusion

In this chapter, we introduced a new unsupervised method called AE-Adapt, for detecting intrusions in a video stream. We proposed a strided 3D convolutional autoencoder that learns spatio-temporal normality while training only on normal videos. To detect intrusions during testing, the error between input and reconstructed video clips is used. This error is thresholded to detect intrusions. But this error depends on the changing scene dynamics like weather, temperature, *etc.* We show that naively thresholding this error, especially on long videos, performs poorly. Therefore, we proposed an adaptive thresholding strategy using moving z-score on this error. We found that this adaptive

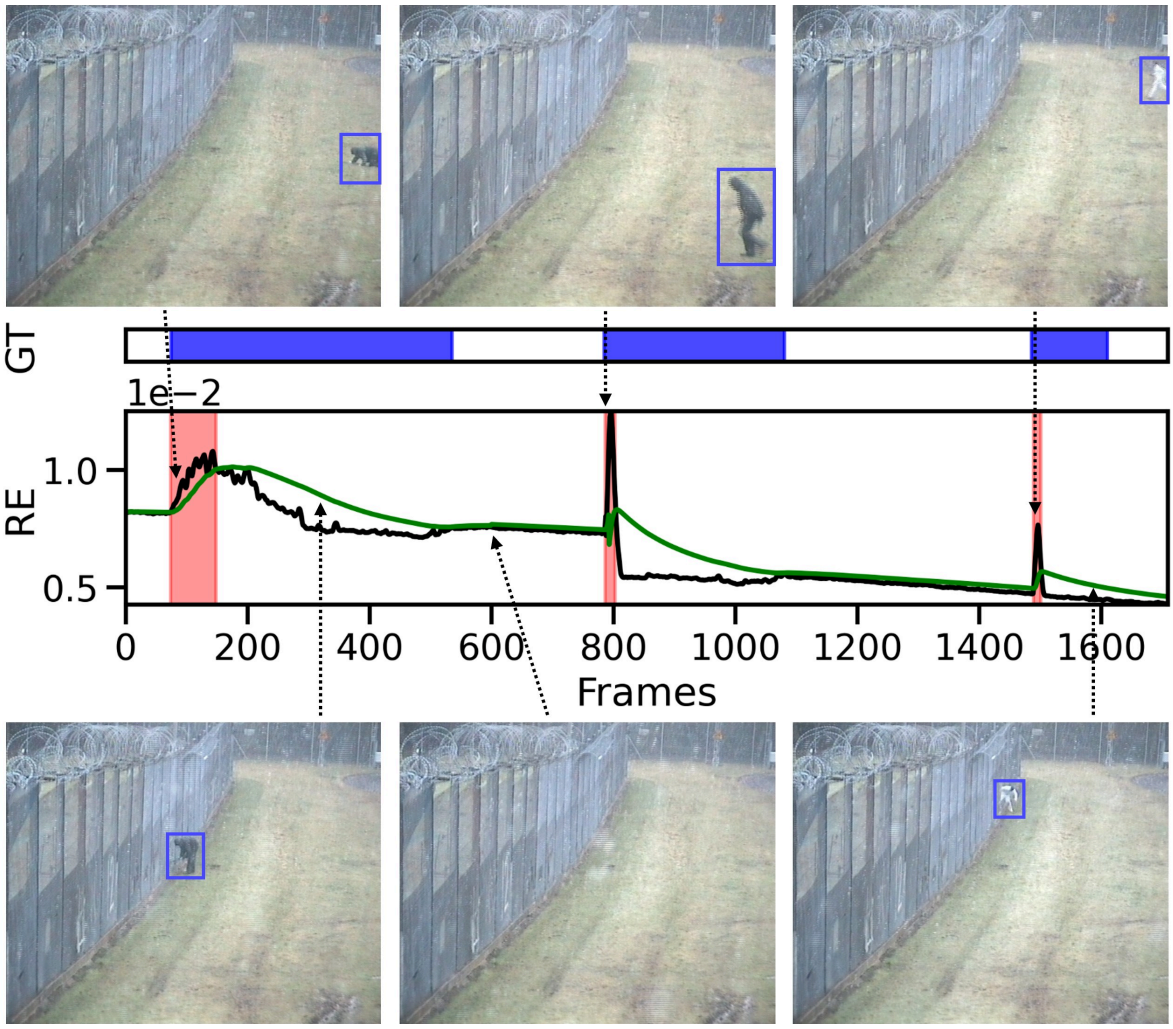



Figure 4.8: AE-Adapt working illustration on a portion of video of i-LIDS test set. In the curve, reconstruction error (RE) is shown in black, threshold in green and alarm in red. GT stands for ground truth per frame, where blue region indicates intrusion. Some illustrative frames are shown above and below the plot, where the blue bounding box shows intruder.

mechanism almost doubles the results of our proposed PIDS.

We also compared our PIDS with two existing methods, using different evaluation protocols. We showed that intrusion detection should be evaluated at a higher level than frame level, because frame level evaluation cannot tell if the beginning of intrusion events is detected or not, which is essential for a PIDS. That is why almost all methods in the state-of-the-art use i-LIDS evaluation. However, the major problem with this evaluation is that it penalizes all alarms after 10 seconds, even if they are well within the intrusion interval. It is indeed too strict in counting FPs, and this hits the overall score negatively, although the system has detected intrusion well within first 10 seconds. We proposed edge-level evaluation to address this issue, where we can parameterize the time after intrusion beginning for the evaluation. This helps in testing the system for different practical time settings and testing its robustness. More importantly, the alarms after a specified time are not counted as false positives if they are within the intrusion. We found that irrespective of method, edge level evaluation should be used for PIDS evaluation.

Concerning compared methods, we found that all the three methods have equivalent

performance on i-LIDS dataset, with edge-level evaluation. The method of Nayak *et al.* (2019) depends on a pre-trained person detector and i-LIDS have only people as intruders, it was an easy task for this method. GOFPID uses traditional PIDS pipeline and also manually drawn perspective to filter object blobs of irregular size. It performs well in this dataset, since there are very few non-intruder moving objects. Our proposed method first learned normality from training set and then detected intrusions as deviations from it. Theoretically, our method can detect intrusions caused by other objects since it does not use beforehand information about intruder class. In fact, for edge-level evalaution for 1 and 2 seconds, our method outperformed other methods. This means it is suitable for rapid real time detection as well. Nevertheless, our proposed method still has drawbacks. That is why, in the next chapter, we try to enhance the spatio-temporal comprehension of normality, to better detect intrusions. Also, since i-LIDS dataset was comparatively easy, we also test these methods on a private dataset, available in annex.

# Chapter 5

# Leveraging Unsupervised and Self-Supervised Learning

In this chapter, we introduce a new approach for spatio-temporal video representation learning. It leverages unsupervised and self-supervised learning and is applicable to both VAD and PID tasks. In the coming sections, we first explain our generic approach. Next, we reveal how it is applied to VAD and PID tasks. Then, the following two sections show experiments and results for the two tasks. Major part of this work was accepted in International Conference on Computer Vision Theory and Applications (VISAPP), 2023.

## Contents

## 5.1 Introduction

In this chapter, we propose a new approach to address the task of video anomaly detection (VAD) and perimeter intrusion detection (PID). As described in Section 2.5, one of the most highly successful approaches to tackle VAD is to use a deep autoencoder (AE) (Hasan *et al.*, 2016; Luo *et al.*, 2017a; Ramachandra *et al.*, 2022) and it also has the least assumptions on data. We also showed in Chapter 4 that the same principle of AE for VAD can be also used for the task of PID. Therefore in this chapter, we proceed with the AE based approaches, proposing a method that leverages unsupervised and self-supervised learning on a single AE. To this end, we devise multiple tasks to enhance the normal spatio-temporal understanding of the AE by training it only on the normal data. Each task has its specific objective: (i) video clip reconstruction (VCR) to learn spatio-temporal characteristics of the normal videos; (ii) future frame prediction (FFP) to learn how normal spatio-temporal patterns propagate along the videos; (iii) playback rate prediction (PRP) to strengthen the playback speed perception of the encoder. Our method is end-to-end trainable and is jointly trained on the three tasks. While testing, the anomaly or intrusion is detected if the combined score of the three tasks is high. Most of the current methods use error measures like mean squared error (MSE) or peak signal to noise ratio (PSNR) for comparing input and reconstructed frames (Astrid *et al.*, 2021a,b; Gong *et al.*, 2019; Park *et al.*, 2020; Lv *et al.*, 2021). But these measures are prone to noise (Gudi *et al.*, 2022; Sinha et Russell, 2011) and therefore we introduce a new measure, called the blur pooled error (BPE). It is locally sensitive and keeps only relevant pixels for error calculation. Most works apply a min-max rescaling to anomaly scores per video (Liu *et al.*, 2018; Gong *et al.*, 2019; Park *et al.*, 2020; Astrid *et al.*, 2021a,b). It is sensitive to extreme values and to address this issue, we propose a robust rescaling of scores for VAD.

Overall, in this chapter, we propose a new approach for spatio-temporal learning in videos. It is generic and we applied it to VAD and PID tasks. In the following sections, we will present our method in general, then its application to VAD and PID tasks, followed by respective experiments and results.

## 5.2 Proposed approach

In this section, we present our proposed LUSS-AE (Leveraging Unsupervised and Self-Supervised AutoEncoder) method. The main idea is to learn normal spatio-temporal features in order to better discriminate anomalies or intrusions from normal patterns. To this end, we propose to train a 3D convolutional autoencoder (3D CAE) on normal videos using carefully designed tasks in a unsupervised and self-supervised manner. The video clip reconstruction task learns spatio-temporal characteristics of normal videos. The future frame prediction task is designed to learn the propagation of spatio-temporal patterns in the normal videos. Finally, the playback rate prediction task strengthens the speed understanding of the encoder. The autoencoder is jointly trained on all these tasks. During testing, each of these branches provides a score to distinguish anomalous or intrusion frames from the normal ones.

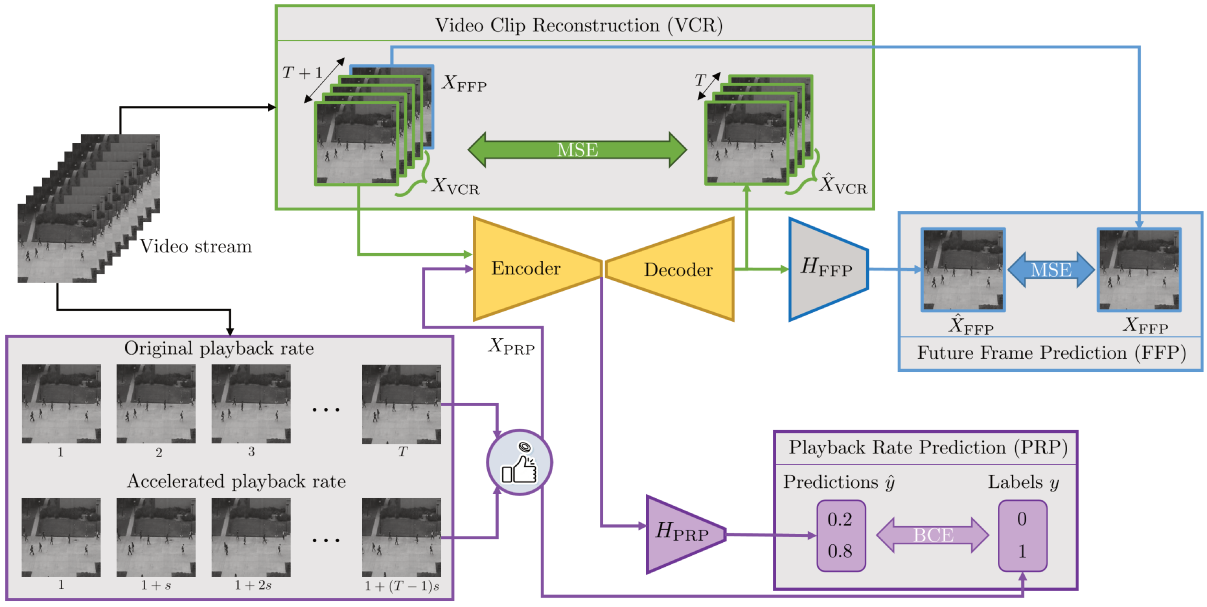



Figure 5.1: Overall schema of the proposed LUSS-AE method.

Figure 5.1 illustrates overall schema of our approach. A video clip of $T+1$ consecutive frames is extracted from the video. The first $T$ frames of this clip goes through the 3D CAE, which reconstructs it. This reconstruction, after passing through an activation function, is compared with original clip to perform the video clip reconstruction task. The same reconstruction (before activation) is fed to the $H_{\text{FFP}}$ head, which predicts a future frame. This frame is compared with the $(T+1)^{\text{th}}$ frame of the input clip, thus performing the future frame prediction task. Another video clip of $T$ frames is extracted from the video. It is either accelerated or kept at the original playback rate. Then, this video clip is fed to the encoder of the 3D CAE. This output is fed to the $H_{\text{PRP}}$ head, which produces a prediction for playback rate of the input clip. This prediction is compared with the actual playback rate of the video clip, to perform the playback rate prediction task. All these tasks are performed jointly and the weighted sum of errors is backpropagated together in the network.

In the following subsections, we first detail the role of each task and how they are trained together, followed by how these tasks act during the testing phase.

### 5.2.1 Learning normal spatio-temporal patterns using multiple tasks

In this subsection, we explain how the proposed tasks help in learning normal characteristics during training. We describe each task with its role, followed by details on how all these tasks are trained in a joint manner.

#### 5.2.1.1 Video Clip Reconstruction (VCR)

Reconstructing a video clip is one of the most popular tasks for unsupervised VAD (Astrid *et al.*, 2021a,b; Gong *et al.*, 2019; Liu *et al.*, 2021b; Zhao *et al.*, 2017). We explored its principle and some major VAD methods in Section 2.5.2.1. In the last chapter, we proposed a video clip reconstruction based PID method and we demonstrated its usefulness for the task. Concretely, the VCR task aims to reconstruct an input video clip, mostly

using an autoencoder (AE) type network. The AE is trained only on normal video clips with the learning objective of minimizing the mean-squared error (MSE) between the input and reconstructed clips. The main hypothesis is that the abnormal or intrusion clips will be badly reconstructed during testing.
Using Eq. (4.1), a non-strided video clip of length $T+1$ frames can be defined as:

$$V_{T+1,1} = \{I_1, I_2, \ldots, I_T, I_{T+1}\} \ . \tag{5.1}$$

The first $T$ frames of this clip is used for the VCR task and we denote it as $X_{\text{VCR}}$, *i.e.*, $X_{\text{VCR}} = \{I_1, I_2, \ldots, I_T\}$. This video clip goes through the autoencoder followed by an activation function to produce a reconstructed clip $\hat{X}_{\text{VCR}}$ as:

$$\hat{X}_{\text{VCR}} = \tanh(\text{Dec}(\text{Enc}(X_{\text{VCR}}))) \ , \tag{5.2}$$

where Enc and Dec stand for encoder and decoder networks, respectively, and tanh is the activation function.
The loss function can then be defined as:

$$\mathcal{L}_{\text{VCR}} = \frac{1}{T \times D \times H \times W} \left\| \hat{X}_{\text{VCR}} - X_{\text{VCR}} \right\|_F^2 \ , \tag{5.3}$$

where $D$, $H$ and $W$ denotes channels (depth), height and width of each frame and $\| \cdot \|_F$ denotes the Frobenius norm.

#### 5.2.1.2 Future Frame Prediction (FFP)

Predicting a future frame is also a well-spread task for unsupervised VAD (Liu *et al.*, 2018, 2021b; Lv *et al.*, 2021; Park *et al.*, 2020) and it was discussed thoroughly in Section 2.5.2.2. To our knowledge, it has not yet been applied for the PID task. FFP aims to predict an unseen future frame, given an input video clip. This requires comprehension of how normal spatio-temporal patterns propagate along the video clip. Like VCR task, the objective is to minimize the MSE, but between the predicted and actual future frame. Since the AE is trained only on normal videos, it should predict the anomalous or intrusion frames poorly.
This task uses the same input of VCR task, *i.e.*, $X_{\text{VCR}}$. After passing through the AE, the video clip $X_{\text{VCR}}$ goes through the prediction head $H_{\text{FFP}}$ to predict the future frame as:

$$\hat{X}_{\text{FFP}} = H_{\text{FFP}}(\text{Dec}(\text{Enc}(X_{\text{VCR}}))) \ . \tag{5.4}$$

This frame is compared with the actual future frame, *i.e.*, frame $T+1$ of $V_{T+1,1}$ (see Eq. (5.1)) denoted as $X_{\text{FFP}}$, where $X_{\text{FFP}} = I_{T+1}$. The loss function is then defined as:

$$\mathcal{L}_{\text{FFP}} = \frac{1}{D \times H \times W} \left\| \hat{X}_{\text{FFP}} - X_{\text{FFP}} \right\|_F^2 \ . \tag{5.5}$$

#### 5.2.1.3 Self-supervised Playback Rate Prediction (PRP)

Section 2.5.2.3 exhibited that self-supervised learning is very recent for the VAD task. Like FFP task, it has never been studied for the PID. We propose here a self-supervised task, *i.e.*, playback rate prediction (PRP), applicable to both VAD and PID tasks. The PRP task does exist in self-supervised representation learning and it is used as a pretext task to learn transferable semantic spatio-temporal features for downstream tasks like

action recognition (Liu *et al.*, 2021a). In other words, first PRP task is performed and later the learned model is adapted to downstream tasks. Contrary to them, we adapt the PRP task, and train it via a single AE with two other tasks, all done simultaneously in a joint and end to end manner.

The original PRP task generates speed labels for video clip sampled at different rates and aims at predicting them (Benaim *et al.*, 2020; Wang *et al.*, 2020; Yao *et al.*, 2020). Since we know that the training videos are normal in our case, we adapt this task to generate two speed-rate labels: original playback rate (implying normal behaviour) and accelerated playback rate with 2x to 5x speed (implying abnormal / intrusion behaviour). The motive is to enforce the encoder with motion comprehension of normal videos. During testing, we hypothesize that the encoder would detect anomalies or intrusions caused by irregular and abrupt motion.

Concretely, given a video clip, this task aims to predict its playback rate. The clip with default playback rate of the video is termed as the original playback rate clip and the clip formed by skipping 1 (2x), 2 (3x), 3 (4x) or 4 (5x) frames is designated as an accelerated playback rate clip. The input $X_{\text{PRP}}$ is a video clip of length $T$, chosen between an original playback rate (class $c = 1$) and an accelerated playback rate (class $c = 2$) with equal chance (50% probability each):

$$X_{\text{PRP}} = \begin{cases} V_{T,1} & \text{when } c = 1 \\ V_{T,s\in\{2,3,4,5\}} & \text{when } c = 2 \end{cases} , \tag{5.6}$$

where the accelerated playback rate clip, when $c = 2$, is a temporally strided video clip with temporal gap $s$ randomly chosen between 2 and 5. This input $X_{\text{PRP}}$ goes through the encoder, playback rate prediction head $H_{\text{PRP}}$ and softmax activation to produce prediction as:

$$\hat{y} = \text{softmax}(H_{\text{PRP}}(\text{Enc}(X_{\text{PRP}}))) \in \mathbb{R}^2 \tag{5.7}$$

The loss function for this classification task is the binary cross-entropy (BCE), defined as:

$$\mathcal{L}_{\text{PRP}} = -\sum_{c=1,2} y[c] \ \log\left(\hat{y}[c]\right) \ , \tag{5.8}$$

where $y$ is the one-hot encoding of the ground truth classes for $X_{\text{PRP}}$.

#### 5.2.1.4 Training objective

A single autoencoder is trained with the above-mentioned tasks and the overall training loss is defined as the weighted sum of individual loss functions:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{VCR}} + \lambda_2 \mathcal{L}_{\text{FFP}} + \lambda_3 \mathcal{L}_{\text{PRP}} \ , \tag{5.9}$$

where $\lambda_1$, $\lambda_2$ and $\lambda_3$ are the weights in $(0, 1]$ that regulate the importance of each task.

### 5.2.2 Testing phase

In this subsection, we describe how the LUSS-AE is used during testing. Given a test video clip, each of the three tasks provides a score and the overall score is a weighted sum of these scores. Depending on the aimed task (VAD or PID), this score is post-processed accordingly, *e.g.,* for VAD, it is normalized. After post-processing, the high values of the score represents anomaly or intrusion. We first define below some image or video error measures and then describe how these measures help to calculate the overall score in inference phase.

#### 5.2.2.1 Image error measures

To quantitatively assess how well a future frame is predicted or how well a video clip is reconstructed, we need to compare them with the appropriate ground truth using an error measure. For this, the most widely used measure is MSE (Gong *et al.*, 2019; Liu *et al.*, 2018; Lv *et al.*, 2021; Zhao *et al.*, 2017). Given two images $J, \hat{J} \in \mathbb{R}^{H \times W \times D}$ (in our case, original and reconstructed frame), the MSE is calculated as:

$$\text{MSE}(J, \hat{J}) = \frac{1}{D \times H \times W} \left\| \hat{J} - J \right\|_F^2 . \tag{5.10}$$

Since last few years, many works use the peak signal to noise ratio (PSNR) measure (Astrid *et al.*, 2021a,b; Park *et al.*, 2022; Ye *et al.*, 2019). But PSNR also depends on MSE as can be seen in its mathematical formulation. Both MSE and PSNR integrate errors on the whole image and therefore are prone to random and incoherent noise (Gudi *et al.*, 2022; Sinha et Russell, 2011). To overcome this, a new measure called the proximally sensitive error (PSE) is proposed by Gudi *et al.* (2022). It is defined as:

$$\text{PSE}(J, \hat{J}) = \frac{1}{D \times H \times W} \left\| (\hat{J} - J) * G_{(\sigma,k)} \right\|_F^2 , \tag{5.11}$$

where $*$ is the convolution operator and $G_{(\sigma,k)}$ is a 2D Gaussian kernel with size $k$ and standard deviation $\sigma$, given by:

$$G_{(\sigma,k)}[i, j] = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} , \tag{5.12}$$

where $i$ and $j$ are the pixel coordinates centered in the kernel. Thanks to the Gaussian convolution, PSE smooths incoherent noise and is locally sensitive. However, an anomaly or intrusion generating an important error in some pixels can disappear in the noise of all other pixels of the high-dimensional image.

To address this, we take this idea one step further and introduce the blur pooled error (BPE), defined as:

$$\text{BPE}(J, \hat{J}) = \frac{1}{D \times H \times W} \left\| S_b((\hat{J} - J) * B_k) \right\|_F^2 , \tag{5.13}$$

where $B_k$ is a generic 2D low-pass filter with kernel size $k$, and $S_b$ signifies a subsampling with stride $b$ (Zhang, 2019). Using a low-pass filter smooths incoherent noise, like in PSE. Then, the subsampling keeps only the most pertinent values from it. Thus, this error has an increased sensitivity to anomalies / intrusions than PSE. All these measures can easily be extended to video clip by applying them to its constituent frames.

#### 5.2.2.2 Test score

Here, we will focus on how different tasks of our approach act during testing phase and provide scores. Concretely, we will see how these scores are calculated and how the final score is obtained. Figure 5.2 shows the schema of our method during testing phase. Given a test video, we first generate non-strided clip of length $T + 1$. The first $T$ frames of this clip, *i.e.,* $I_{1:T}$ is the input to our method. This input first goes through the encoder and the resultant latent space is input to two branches, *i.e.*, decoder and $H_{\text{PRP}}$ head. The one that goes through $H_{\text{PRP}}$ head produces predictions $\hat{y}$ which are used to compute the $A_{\text{PRP}}$

score. While the one that goes through decoder bifurcates again into two branches, *i.e.*, reconstruction and $H_{\text{FFP}}$ head. With reconstruction branch, the input clip is reconstructed as $\hat{I}_{1:T}$ and the $A_{\text{VCR}}$ score is computed. The branch of $H_{\text{FFP}}$ head produces the future frame $\hat{I}_{T+1}$, which is compared with frame $I_{T+1}$ of the input to calculate the $A_{\text{FFP}}$.

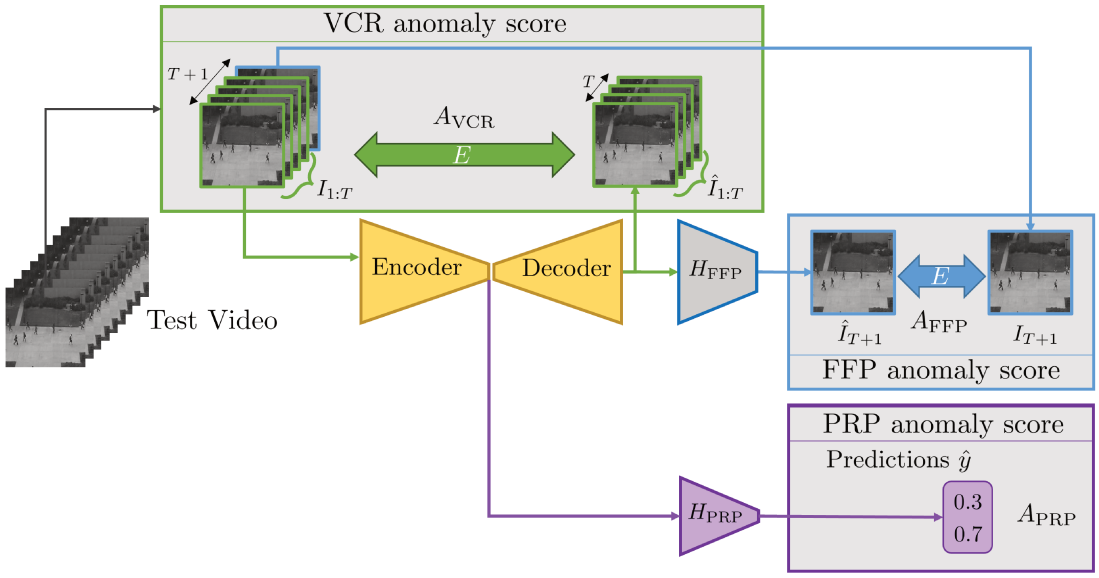



Figure 5.2: Overall schema of the proposed LUSS-AE method during testing. A window of $T+1$ consecutive frames is drawn sequentially from the test video. The first $T$ frames from this window are the input of the system and the output for each task is computed on it. The anomaly score is determined for each task, i.e., $A_{\text{VCR}}$, $A_{\text{FFP}}$ and $A_{\text{PRP}}$, and the final anomaly score is their weighted sum.

The anomaly scores for the three tasks are described below:
(i) the VCR anomaly score is defined as:

$$A_{\text{VCR}} = \frac{1}{T}\sum_{t=1}^{T} E(\hat{I}_t, I_t) \ , \tag{5.14}$$

where $I_t$, $\hat{I}_t$ are the $t^{\text{th}}$ frame and its reconstruction respectively, and $E$ can be one of three error measures, *i.e.*, MSE, PSE or BPE respectively.
(ii) the FFP anomaly score is computed as:

$$A_{\text{FFP}} = E(\hat{I}_{T+1}, I_{T+1}) \ . \tag{5.15}$$

(iii) the PRP anomaly score is defined as the probability of accelerated class ($c = 2$):

$$A_{\text{PRP}} = \hat{y}[2] \ , \tag{5.16}$$

where $\hat{y}$ is the output of PRP branch as defined in Eq. (5.8).
The final score is defined as:

$$A = \alpha_1 A_{\text{VCR}} + \alpha_2 A_{\text{FFP}} + \alpha_3 A_{\text{PRP}} \ , \tag{5.17}$$

where $\alpha_1, \alpha_2, \alpha_3$ are the weights in $[0, 1]$ for the three scores.

## 5.3 Application to tasks

So far, we introduced the main idea of LUSS-AE. It is generic and can be theoretically applied to any unsupervised video learning task. In this section, we explain concretely how it is applied to video anomaly detection and perimeter intrusion detection task, and what is the final measure that leads to detection of anomalies and intrusions.

### 5.3.1 VAD task

We recall that in this manuscript, we tackle the problem of video anomaly detection without anomaly information (refer Section 3.2.2). In other words, we have training data with no labels, and it is considered safe to assume that it contains only normal events. The idea is to understand these normal events during model training to detect anomalies as a deviation to normality during testing. We use our proposed LUSS-AE with video clip (VC) as the input unit. As explained in Section 2.5.1, video clip is an ideal input for VAD since it contains both spatial and temporal information and does not have bias towards external detectors. Our motivation is to propose a generic VAD method without extra supervision or bias. We present below the network architecture we use in LUSS-AE for VAD and the final anomaly score to detect anomalies.

#### 5.3.1.1 Network architecture

In this subsection, we detail the network architecture of different components of our method. Precisely, the components are autoencoder, the future frame prediction head and the playback rate prediction head.

**a) Autoencoder** We do not propose a new autoencoder architecture and instead use a widely popular architecture for VAD. Like this, we can study the viability of our method and understand how exactly it is better or worse in comparison to the existing works. We use the strided 3D convolutional autoencoder proposed by Gong *et al.* (2019), and it is also used by other works (Astrid *et al.*, 2021a,b). The reason for choosing this architecture is because there are 3D convolutions and 3D deconvolutions, which help in learning spatio-temporal features jointly from a video stream (refer to Sections 2.3.3 and 4.2.1.1 for more information).

In Figure 5.3, we demonstrate this architecture. It takes a video clip of 16 frames as input, where each frame is $256 \times 256$ in size. The encoder uses a series of 3D convolutions with kernel size of $3 \times 3 \times 3$ (shape : temporal length, height, width). The stride in first convolution is of shape $1 \times 2 \times 2$, signifying it reduces only the spatial dimension by half. Rest of the convolutions diminish all the three dimensions by half using $2 \times 2 \times 2$ stride. The decoder uses 3D deconvolutions or transposed convolutions with the same kernel size of $3 \times 3 \times 3$. The stride is $2 \times 2 \times 2$, except for the last deconvolution which has $1 \times 2 \times 2$ as stride. Each layer is followed by 3D batch normalization and LeakyReLU activation, except for the last layer. Finally, the output is of the same shape as the input.

**b) Future frame prediction head** Figure 5.4 shows the proposed architecture of $H_{\text{FFP}}$ head for VAD task. It takes the output of decoder as its input. This input is passed through a 3D convolution with kernel size of $3 \times 3 \times 3$ and stride $16 \times 1 \times 1$, followed by tanh activation function to produce the future frame.

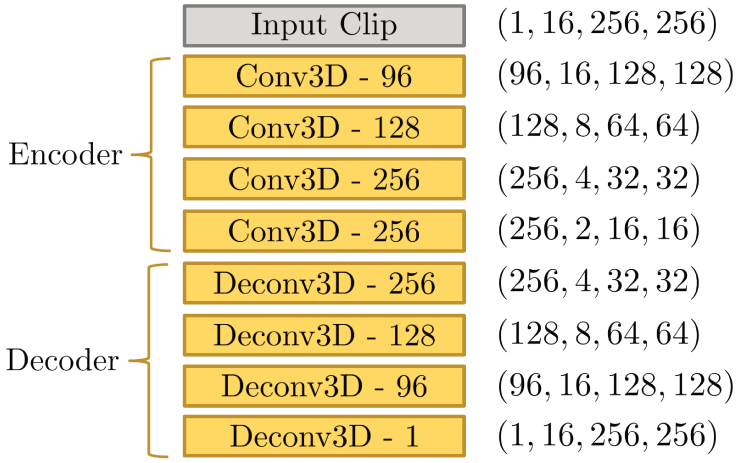



Figure 5.3: Autoencoder architecture for VAD with encoder and decoder. Conv3D and Deconv3D refers to 3D convolution and 3D deconvolution respectively, and the number that follows them, denotes amount of filters. The rightmost column refers to the data shape (channels or filters, temporal length, frame height, frame width).

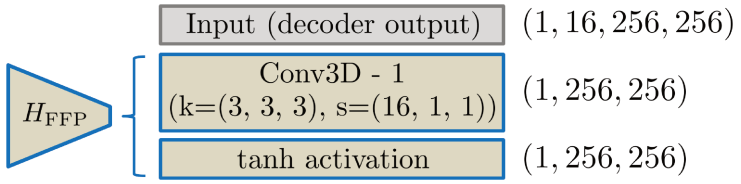



Figure 5.4: Architecture of FFP head *i.e.*, $H_{\text{FFP}}$, for VAD task. Conv3D refers to 3D convolution, where k and s are kernel size and stride respectively. The rightmost column refers to the data shape.

**c) Playback rate prediction head** The proposed architecture of $H_{\text{PRP}}$ head for VAD is shown in Figure 5.5. It takes the encoder output as its input, *i.e.*, an input of size $(256, 2, 16, 16)$. The input is reduced to a smaller dimension using a series of max-pooling and 2D convolution operations. This reduced input is then flattened to a vector of 512 dimensions. Then, three consecutive fully connected (FC) layers reduce the vector dimension to 2, where the first two FC layers are followed by ReLU activation function. Finally, this vector is converted into probabilities using the softmax activation function.

#### 5.3.1.2 Anomaly score

Most VAD works apply a min-max rescaling to scores per video (Liu *et al.*, 2018; Gong *et al.*, 2019; Park *et al.*, 2020; Astrid *et al.*, 2021a,b). This scaling bounds values to interval $[0, 1]$ where the minimum and maximum values are forced to be 0 and 1 respectively. Due to this, it is prone to outliers with extreme values. To address this issue, we propose a robust rescaling per video. For a test video with $n$ frames, the rescaled anomaly score for frame $t$ is defined as:

$$\tilde{A}_t = \frac{A_t - \text{med}\left(\{A_i\}\right)}{\text{iqr}_{1-99}\left(\{A_i\}\right)} \,, \tag{5.18}$$

where $\text{med}(\cdot)$ and $\text{iqr}_{1-99}(\cdot)$ are respectively the median and the interquantile range (between $1^{\text{st}}$ and $99^{\text{th}}$ percentiles) of scores $\{A_i\}_{i=1}^{n}$. Finally, like previous methods, the higher scores correspond to anomalies.

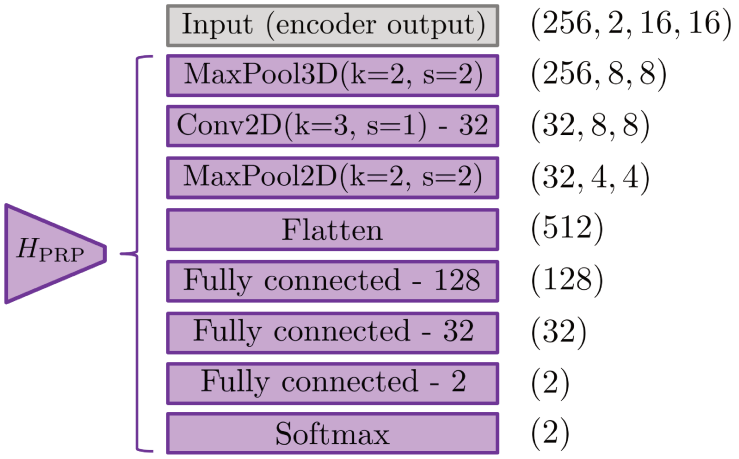



Figure 5.5: Architecture of PRP head *i.e.*, $H_{\text{PRP}}$, for VAD task. The different layers are depicted in middle column with violet color and the number that follows them, denotes amount of filters or units. Kernel size and stride are denoted by k and s, wherever applicable. The rightmost column refers to the data shape (channels, temporal length, frame height, frame width).

### 5.3.2 PID task

For PID task, we follow the same intuition as the last chapter. In other words, we train LUSS-AE on training data with no labels to understand normality, so that intrusion can be detected during testing as they would follow a deviation from normality. Like VAD and previous PID method (see Section 4.2.1.1), we use video clip as input. We present below the network architecture we propose in LUSS-AE for PID and the final intrusion score to detect intrusions.

#### 5.3.2.1 Network architecture

In this subsection, we detail the network architecture of different components of our method for PID.

**a) Autoencoder** For autoencoder, we use the same 3DCAE architecture that we proposed in last chapter. In this way, we can investigate how well or worse the LUSS-AE performs in comparison to PID method proposed in the last chapter. As shown in Figure 4.1, this autoencoder takes an input video clip of 8 frames, where each frame is $720 \times 576$ in size. The encoder uses a series of strided 3D convolutions to produce a latent space of $(8, 2, 180, 144)$. The decoder takes this latent space as input and uses a series 3D deconvolutions to produce an output with same size as that of the input video clip. Refer Section 4.2.1.1 for more details. Even though autoencoder architecture for both VAD and PID have strided 3D convolutions and deconvolutions, the architecture of PID takes only 8 frames as input instead of 16, has a much higher frame resolution than VAD, and smaller amount of filters and layers, making it computationally fast.

**b) Future frame prediction head** The proposed architecture $H_{\text{FFP}}$ for PID is shown in Figure 5.6. It takes the output of decoder as its input. A 3D convolution with kernel size of $3 \times 3 \times 3$ and stride $8 \times 1 \times 1$ is first applied to this input, and then a tanh activation function is used to produce the future frame.

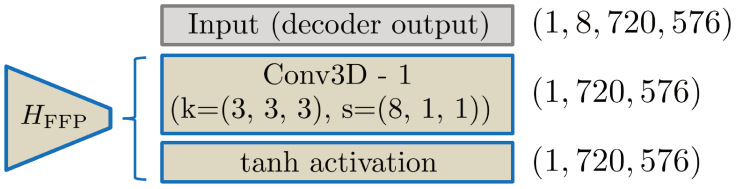



Figure 5.6: Architecture of FFP head *i.e.*, $H_{\text{FFP}}$, for perimeter intrusion detection. Conv3D refers to 3D convolution, where k and s are kernel size and stride respectively. The rightmost column refers to the data shape (channels, temporal length, frame height, frame width).

**c) Playback rate prediction head** The proposed architecture of $H_{\text{PRP}}$ head for PID is shown in Figure 5.7. The encoder output of size $(8, 2, 180, 144)$ is its input. This input is reduced to a smaller dimension using a series of max-pooling and 2D convolution operations. It is then flattened to a vector of 828 dimensions. Then, three consecutive fully connected (FC) layers reduce the vector dimension to 2, where the first two FC layers are followed by ReLU activation function. Finally, this vector is converted to probabilities using a softmax activation function.

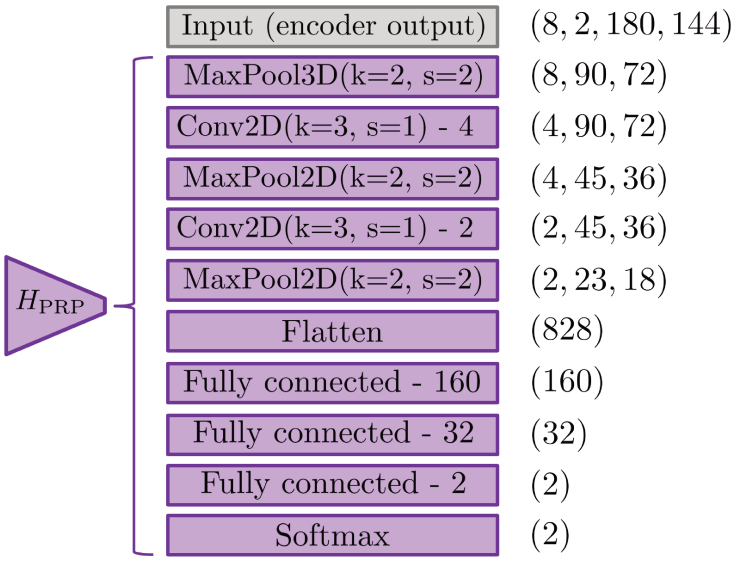



Figure 5.7: Architecture of PRP head *i.e.*, $H_{\text{PRP}}$, for perimeter intrusion detection. The different layers are depicted in middle column with violet color. Kernel size and stride are denoted by k and s, wherever applicable. The rightmost column refers to the data shape.

#### 5.3.2.2 Adaptive Intrusion score

To finally detect intrusion, we need a score that describes it. The overall testing score $A$ of Equation (5.17) can be used in three ways as an intrusion score: (i) directly using $A$ as intrusion score; (ii) using normalized $A$ (like in VAD task) as intrusion score; (iii) using adaptive mechanism of moving z-score (MZ) on $A$ (refer to Section 4.2.2.2). We concluded in the last chapter that adaptive thresholding is essential for a PID system, therefore we choose the third way for our experiments with LUSS-AE. We follow exactly the same method of adaptive thresholding as described in last chapter. We just replace $r$ values with $A$ values in Equations (4.4 and 4.5), respectively. Since we use LUSS-AE

with adaptive thresholding, we name this method as LUSS-AE-Adapt, in correspondence to the nomenclature of the last chapter.

## 5.4 VAD: Experiments and results

We perform experiments on three publicly available benchmark datasets: UCSD Ped2 (Li *et al.*, 2013), CUHK Avenue (Lu *et al.*, 2013), and ShanghaiTech (Luo *et al.*, 2017a). Each dataset has a standard training / test division, where the training set consists of only normal videos while testing set has videos with one or more anomalous events (refer to Section 1.4.1). We evaluate with the frame-level area under the receiver operating characteristic (AUROC) metric. Precisely, we use "AUROC on all videos" (micro-averaged AUROC) computed on the whole test set with thresholds common to all test videos. Refer to Section 3.5.2.1 for more details.

### 5.4.1 Implementation details

For both training and testing in each dataset, the frames are resized to $256 \times 256$ and pixels are rescaled to the range $[-1, 1]$. The autoencoder takes a video clip of 16 frames in grayscale as input, *i.e.*, $T$=16, $D$=1, $H$=256 and $W$=256 respectively. While training, the input clip for PRP task is chosen between original playback rate and accelerated playback rate with equal chance (50% probability for each). For each batch of accelerated playback rate, the value of s is chosen randomly from $(2, 3, 4, 5)$ with equal chance for the four values. The balance weights in the training objective function are set to $\lambda_1$=0.6, $\lambda_2$=0.4 and $\lambda_3$=1, respectively. The whole model is trained end-to-end using the Adam optimizer (Kingma et Ba, 2015) with a learning rate of $10^{-4}$ and a batch size of 16. While testing, a single video clip is used for the three tasks (see Figure 5.2). We use different measures like MSE, PSE and BPE for the anomaly score. For PSE and BPE, we use $\sigma$=1, $b$=2 while keeping the same kernel size of $k$=5. For blur kernel, we use a Gaussian filter. After grid search, we set the optimal weights for anomaly score $(\alpha_1, \alpha_2, \alpha_3)$ as (0.1, 0.8, 1), (0.1, 1, 0.1) and (0.2, 0.2, 0.9) for Ped2, Avenue and ShanghaiTech dataset, respectively.

### 5.4.2 Quantitative results

In this subsection, we present a quantitative comparison of our method with existing state of the art methods. Table 5.1 shows this quantitative results. The listed methods belong to the category of VAD without training labels. As explained in Section 2.5, some of the unsupervised VAD methods do use external supervision via object detectors, pre-trained feature extractors, *etc.*, but these methods should not be directly compared with fully unsupervised VAD methods and most work respect this notion (Astrid *et al.*, 2021a; Park *et al.*, 2022). The work of MNAD (Park *et al.*, 2020) and MPN (Lv *et al.*, 2021) has been re-implemented by Menon et Stephen (2021) and Lu *et al.* (2022), respectively. In both cases, the originally claimed results are not reproducible, and they are marked with * in the table.

| Method | Dataset | | |
|---|---|---|---|
| | Ped2 | Avenue | Shanghai |
| AnoPCN (Ye *et al.*, 2019) | 96.80 | 86.20 | 73.60 |
| MemAE (Gong *et al.*, 2019) | 94.10 | 83.30 | 71.20 |
| UNet-inte (Tang *et al.*, 2020) | 96.30 | 85.10 | 73.00 |
| Cluster AE (Chang *et al.*, 2020) | 96.50 | 86.00 | 73.30 |
| MNAD (Park *et al.*, 2020) * | *97.00** | *88.50** | *70.50** |
| MNAD (Menon et Stephen, 2021) | 96.33 | 87.91 | 67.81 |
| STEAL Net (Astrid *et al.*, 2021b) | 98.40 | 87.10 | 73.70 |
| LNTRA (Astrid *et al.*, 2021a) | 96.50 | 84.91 | 75.97 |
| MPN (Lv *et al.*, 2021) * | *96.90** | *89.50** | *73.80** |
| MPN (Lu *et al.*, 2022) | 96.13 | 83.90 | 73.00 |
| ITAE (Cho *et al.*, 2022) | 97.30 | 85.80 | 74.70 |
| VQU-Net (Szymanowicz *et al.*, 2022) | 89.20 | 88.30 | - |
| FastAno (Park *et al.*, 2022) | 96.30 | 85.30 | 72.20 |
| LUSS-AE [ours] | **98.52** | **89.04** | **81.21** |

Table 5.1: Quantitative comparison with the existing state of the art methods: AUROC (%) for VAD is computed on Ped2, Avenue and Shanghai datasets. Numbers in bold indicate the best performance, and * indicate non-reproducible results.

We can observe that our method outperforms all the other methods across all the datasets. The performance gain in Ped2 and Avenue datasets is less significant as in the Shanghai dataset. In fact, it has been suggested to not use the Ped2 dataset as the performance in it is almost saturated (Acsintoae *et al.*, 2022). The Shanghai dataset is considered one of the most difficult dataset and our high performance gain of 5.24% in comparison to the best method in state of the art, demonstrates the viability of our method. Furthermore, the fact that our method works on all the datasets, irrespective of the type of anomalies, shows its generalizing ability. Even though, we use the same architecture for autoencoder like many other methods (Astrid *et al.*, 2021a,b; Gong *et al.*, 2019), still our proposed method outperforms them, without using any sort of external memory or supervision. All these points demonstrate the strength and effectiveness of our LUSS-AE method.

### 5.4.3 Qualitative results

In this part, we discuss the strengths and weaknesses of our method via visual examples.

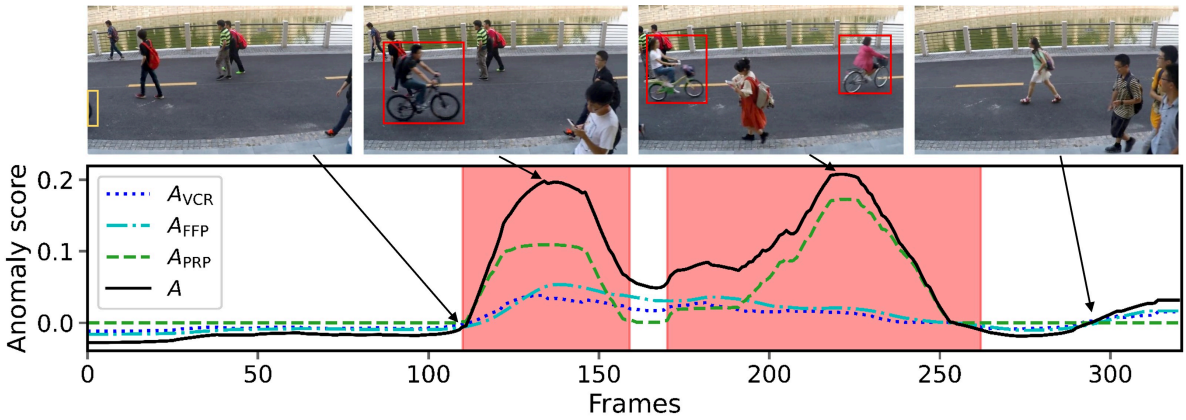



Figure 5.8: LUSS-AE working illustration on video 11_0176 of Shanghai test set. Anomaly scores ($A_{\text{VCR}}$, $A_{\text{FFP}}$, $A_{\text{PRP}}$, $A$) are plotted per video frame; red regions depict anomalous events and some illustrative frames are shown above the plot, where the yellow and red bounding boxes exhibit objects of interest and anomalies respectively.

Figure 5.8 demonstrates an example of our method tested on a video with two anomalous events, both containing movement of bikes. Here, the people move with relatively normal speed while bikes move with fast speed. Also, the number of people are relatively less in this example and bike does occupy a big area, which means its displacement causes a big spatio-temporal change. We can observe that as soon as the bike enters the scene, we have a high jump in $A_{\text{PRP}}$ and it remains high until the bike exits the scene. It jumps up again in next scene and have highest value when two bikes move in the scene. Regarding $A_{\text{VCR}}$ and $A_{\text{FFP}}$, the scores remain high when bikes are in the scene. Overall, our method detects both anomalous events in this example.

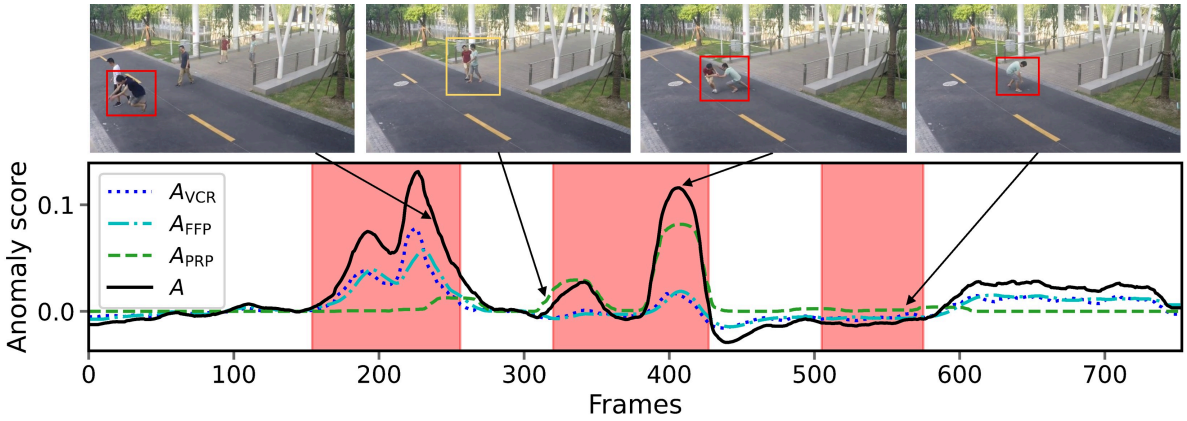



Figure 5.9: LUSS-AE working illustration on video 05_0023 of Shanghai test set. Anomaly scores ($A_{\text{VCR}}$, $A_{\text{FFP}}$, $A_{\text{PRP}}$, $A$) are plotted per video frame; red regions depict ground truth anomalous events, and some illustrative frames are shown above the plot, where the yellow and red bounding boxes exhibit objects of interest and anomalies respectively.

Figure 5.9 demonstrates another example containing three different anomalous events: person turning in wrong direction and person jumping, brawling/chasing action, and stone picking. We can observe that the problem is difficult in this example as not only the anomalies are spatio-temporal but they also depend on context. In the first anomalous

event, $A_{\text{VCR}}$ and $A_{\text{FFP}}$ have higher values than $A_{\text{PRP}}$ in the beginning. The anomaly here consists of person turning in wrong direction which is well captured by the VCR and FFP task. Later, when the person jumped, $A_{\text{PRP}}$ suddenly increases, indicating its sensitivity to abrupt motion. During the second anomalous event, we observe that $A_{\text{PRP}}$ has higher values than $A_{\text{VCR}}$ and $A_{\text{FFP}}$, thus contributing primarily to detect the anomaly. The $A_{\text{PRP}}$ starts to increase just before the start of this event because the person in red starts to suddenly approach the other person. We then observe a first peak of $A_{\text{PRP}}$ as one person pushes other to the ground. We later observe a big second peak of $A_{\text{PRP}}$ and this relates to fast movement chasing. However, the third event is very rare (picking up stones) and does not contain large spatio-temporal movement in the scene and thus our method fails to detect it. In fact, the score in later frames is slightly higher than the third event because there are three people in close proximity, trying to change directions, which is considered anomalous for Shanghai dataset. Overall, the VCR and FFP tasks work better in anomalies without abrupt motions and PRP task addresses the anomalies with sudden motions. There is still a room to improve the spatio-temporal comprehension of AE for VAD, possibly with a data augmentation technique as the datasets lack more examples of targetted scenarios.

### 5.4.4 Ablation studies

In this subsection, we study the importance of different parts of our proposed method.

#### 5.4.4.1 Are all tasks useful ?

In this ablation study, we analyze the impact of different tasks on the autoencoder for detecting anomalies. Table 5.2 shows combinations of different tasks and their respective AUROC performances on Avenue and Shanghai datasets. Concretely, for each of these combinations, we train and test AE using the chosen tasks, and use the introduced BPE for computing anomaly score during inference.

| Tasks | | | AUROC (%) | |
|---|---|---|---|---|
| VCR | FFP | PRP | Avenue | Shanghai |
| ✓ | | | 82.72 | 73.11 |
| ✓ | ✓ | | 87.95 | 76.35 |
| ✓ | | ✓ | 87.43 | 79.20 |
| ✓ | ✓ | ✓ | **89.04** | **81.21** |

Table 5.2: Influence of different tasks (VCR, FFP and FRP) used during training and testing of AE on Avenue and Shanghai datasets, in terms of AUROC (%).

We observe that when AE is trained only with the VCR task, the VAD performance is the least, *i.e.*, 82.72% and 73.11%. We consider this as the baseline since this a standard task in VAD (Kiran *et al.*, 2018) and we combine it with other tasks to assess their impact. Using VCR and FFP tasks together, boost the performance with a gain of 5.23% and 3.24% over the baseline, indicating the importance of learning propagation of normal spatio-temporal patterns via the FFP task. Similarly, when PRP task is trained together with the VCR task, we observe an increase of 4.71% and 6.09% in comparison to baseline,

validating that indeed playback speed perception enriches the comprehension of normality for anomaly detection. Finally, when all the tasked are used together, we observe a substantial yield in performance with 6.32% and 8.10% over the baseline, demonstrating the effectiveness of our proposed approach to train AE by leveraging unsupervised and self-supervised tasks.

#### 5.4.4.2 Is the autoencoder enriched by FFP and PRP?

In this work, we employ a 3D CAE used in many previous works (Astrid *et al.*, 2021a,b; Gong *et al.*, 2019). All these works used it with the VCR task. In this study, we first reproduced their work by training and testing 3D CAE with the VCR task. Then, we trained it with the proposed tasks, *i.e.*, VCR, FFP and PRP, and tested using only the VCR task. This way, we can assess the impact of these tasks on AE's comprehension of normality during training to detect anomalies during testing.

| Training tasks | Testing only with VCR | |
|---|---|---|
| | Avenue | Shanghai |
| VCR | 82.72 | 73.11 |
| VCR + FFP + PRP | **86.48** | **75.37** |

Table 5.3: Influence of tasks (VCR, FFP and PRP) used during training of autoencoder on Avenue and Shanghai datasets, when tested only with VCR, in terms of AUROC (%).

Table 5.3 exhibit this ablation study. It can be observed by training the same autoencoder with proposed tasks, a gain of 3.76% and 2.26% is obtained on Avenue and Shanghai datasets, respectively. Since the three tasks are used only during training and only VCR is used while testing, this gain in performance reveals that these tasks enriched the normality comprehension of the 3D CAE.

#### 5.4.4.3 Are error measures equivalent?

The goal of this ablation study is to see the effect of different error measures, *i.e.*, MSE, PSE and BPE, on the VAD task. Since these measures apply to VCR and FFP tasks, PRP task will not be considered here.

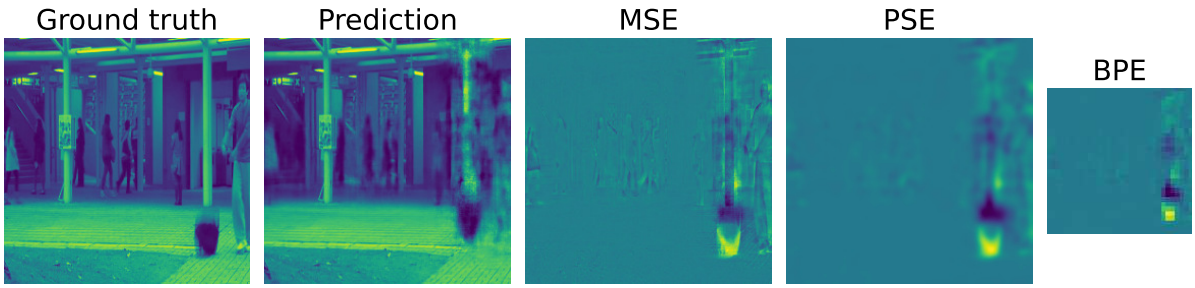



Figure 5.10: Illustration of different error measures on a test frame of the Avenue dataset. From left to right: actual frame (ground truth), predicted frame, error frame for MSE, for PSE and for BPE. (Best viewed in color)

Figure 5.10 shows an visual example using the FFP task to better understand these measures. We can observe that AE did not correctly predict this frame, where dropping bag is an anomaly. The error frame for MSE highlights this anomalous region of the image but it also captures the background noise of the frame. The PSE error frame has a more visible region of anomaly, and it smooths some background noise. Finally, the BPE error frame has a principal focus on anomalous region and has theleast amount of background noise. Furthermore, the BPE frame is smaller than other maps as we remove irrelevant pixels via subsampling.

| Error measure | Testing tasks | |
|---|---|---|
| | VCR | FFP |
| MSE | 84.95 | 85.38 |
| PSE | 85.96 | 86.61 |
| BPE | **86.48** | **87.00** |

Table 5.4: Influence of error measure (MSE, PSE and BPE) on each task (VCR and FFP) during testing of AE on Avenue dataset, in terms of AUROC (%).

Table 5.4 shows the quantitative impact of these error measures. To be precise, we train our method with the three tasks and test it with the respective tasks and measures shown in the table. We can observe that the PSE improves the performance in both tasks with 1.01% and 1.23% respectively, signifying the importance of proximity error and noise reduction. BPE provides a significant boost in results with 1.53% and 1.62% performance improvement over MSE in the two tasks. This validates the importance of our proposed BPE for the VAD task.

## 5.5 PID: Experiments and results

We perform experiments on the i-LIDS sterile zone dataset with the two cameras views as described in Table 1.2. Using only non-intrusion videos, model was trained on view 1 of i-LIDS training set and tested on view 1 of i-LIDS test set (same procedure for view 2). Test set uses the same protection perimeter as defined in the last chapter. Since, in the last chapter, we concluded the importance of edge-level (EL) protocol for PID evaluation, we use only this protocol in this chapter. Finally, we do not demonstrate the ablation studies like in VAD (refer Section 5.4.4) because we had similar observations and thus explanations would be repetitive. In the next subsections, we first describe the implementation details of LUSS-AE-Adapt for PID, followed by overall results and how it differs from PIDS proposed in the last chapter. Finally, in the last subsection we provide a summary of how our method performs in a real-world PID case.

### 5.5.1 Implementation details

Each frame is converted to grayscale, pre-processed using histogram equalization and pixels were rescaled to $[-1, 1]$. The autoencoder we use takes a video clip of 8 frames as input, *i.e.*, $T$=8, $D$=1, $H$=576 and $W$=720 respectively. Like VAD, the input clip for PRP task is chosen between original playback rate and accelerated playback rate with equal

chance. For accelerated playback rate, s is chosen randomly from $(2, 3, 4, 5)$ with equal chance for the four values. Concerning training objective function, the balance weights are experimentally set to $\lambda_1$=0.6, $\lambda_2$=0.4 and $\lambda_3$=1 respectively. The whole model is trained end-to-end using the Adam optimizer (Kingma et Ba, 2015) with a learning rate of $10^{-4}$ and a batch size of 24.

While testing, we used the BPE measure for the overall test score. For blur kernel, we use a Gaussian filter with $\sigma$=1, $b$=2 and kernel size $k$ of 5. After grid search, we set the optimal weights for final score $(\alpha_1, \alpha_2, \alpha_3)$ as (1, 1, 0.1). Finally, the adaptive thresholding is done on moving z-score of $A$. We use values of first 10 frames for initialization, moving z-score is used with $z_{th} = 4.5$ and $\alpha = 0.01$, like work in previous chapter. Following the last chapter, for edge-level evaluation, we set the tolerance variables $m_{\text{pre}}$ and $m_{\text{post}}$ to 3, *i.e.*, less than 1 second of tolerance. For variable $m$, the following values were chosen: 5, 10 and 50. This signifies that we tested the methods to detect within 1, 2 and 10 s from the beginning of intrusion.

### 5.5.2 Results

Tables 5.5 and 5.6 show the comparative results for PID in View 1 and View 2 of i-LIDS datasets respectively. The first three methods mentioned here are well defined in Section 4.3.1.2 and were used in evaluation in the last chapter. Their results shown in these tables are taken from the last chapter.

| Method | EL (1 s) | | | EL (2 s) | | | EL (10 s) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| GOFPID | 0.69 | 0.61 | 0.65 | 0.76 | 0.87 | 0.81 | 0.78 | 0.98 | 0.87 |
| Nayak *et al.* (2019) | 0.62 | 0.52 | 0.57 | 0.70 | 0.74 | 0.72 | 0.74 | 0.93 | 0.83 |
| AE-Adapt | 0.90 | 0.74 | 0.81 | 0.92 | 0.90 | 0.91 | 0.92 | 0.91 | 0.92 |
| LUSS-AE-Adapt | 0.91 | 0.78 | **0.84** | 0.93 | 0.91 | **0.92** | 0.95 | 0.93 | **0.94** |

Table 5.5: PID methods comparison based on edge-level (EL) evaluation at 1, 2 and 10 seconds for View 1 of i-LIDS dataset.

| Method | EL (1 s) | | | EL (2 s) | | | EL (10 s) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| GOFPID | 0.80 | 0.59 | 0.68 | 0.83 | 0.73 | 0.78 | 0.86 | 0.94 | 0.90 |
| Nayak *et al.* (2019) | 0.99 | 0.47 | 0.64 | 0.99 | 0.69 | 0.82 | 0.99 | 0.94 | **0.96** |
| AE-Adapt | 0.83 | 0.64 | 0.73 | 0.89 | 0.80 | 0.84 | 0.90 | 0.92 | 0.91 |
| LUSS-AE-Adapt | 0.85 | 0.70 | **0.77** | 0.90 | 0.82 | **0.86** | 0.94 | 0.93 | 0.93 |

Table 5.6: PID methods comparison based on edge-level (EL) evaluation at 1, 2 and 10 seconds for View 2 of i-LIDS dataset.

We can observe that the proposed LUSS-AE-Adapt outperforms AE-Adapt in both views. In View 1, we obtain the state-of-the-art results with a good margin in terms

of $F_1$ scores in every edge-level metric. Concerning View 2, we still outperform other methods in 1 and 2 seconds of edge-level evaluation and we are just slightly behind Nayak *et al.* (2019) in 10 seconds. This means that we detect better than other methods in online scenario, *i.e.,* 1 or 2 seconds. It can also be observed that LUSS-AE-Adapt have constantly better precision and recall scores than AE-Adapt, regardless of edge-level evalaution parameters or camera view. This signifies that indeed using our proposed LUSS-AE-Adapt for PID increases the spatio-temporal comprehension of normality, in order to better detect intrusions.

### 5.5.3 Importance of adaptiveness in proposed PID methods

Even though the same 3DCAE is used in both AE-Adapt and LUSS-AE-Adapt, we would like to explore where exactly these two methods differ in terms of PID performance. Table 5.7 shows the comparative performances of these two methods, with and without moving z-score, on View 1 of i-LIDS dataset. Please note that without moving z-score, AE-Adapt or LUSS-AE-Adapt looses the adaptiveness (Adapt) and boils down to just autoencoders.

| Method | Without moving z-score | | | With moving z-score | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| AE-Adapt | 0.54 | 0.44 | 0.49 | 0.92 | 0.91 | 0.92 |
| LUSS-AE-Adapt | 0.62 | 0.51 | **0.56** | 0.95 | 0.93 | **0.94** |

Table 5.7: Comparison (EL (10 s)) of proposed unsupervised PID methods on i-LIDS dataset, taking into account the effect of moving z-score.

We can observe that LUSS-AE-Adapt is 7% better than AE-Adapt when no moving z-score is used. This means that using carefully chosen tasks increases the intrusion detection ability of the autoencoder. When moving z-score is used, both the methods get massive gains in performances. Moreover, the gap in their performances lowers down drastically. This indicates that adaptive thresholding is essential for a PIDS when long time-length videos are present. Even though autoencoder can detect the intrusions locally (for example, if small videos are taken like in VAD datasets), thresholding it's reconstruction error globally for long-length videos leads to poor performance. This is because even though reconstruction error is high locally for intrusions in different scene dynamics like day and night, but the gap between these errors is very large and thus one cannot find a fixed optimal threshold to detect all the intrusions. Thus, the adaptive component is a must and it increases the ability of detecting intrusion in both PID systems.

### 5.5.4 Testing proposed PIDS in a more challenging real-world scenario

Until now, all experiments concerning intrusion detection were done in i-LIDS dataset. As already explained in previous chapters, even though this dataset pose some interesting challenges, it still lacks in many ways, like it only has people as intruders, the scenes are simple and most of the time if something is moving, it is an intrusion (thus lack of distractions). To address this, we performed experiments on a private industrial dataset.

This dataset has sheeps roaming in a perimeter continuously and the aim is to detect human intruder which can walk, run, crawl, *etc.* Moreover, the intruder is present at various distances from the camera and at a very far distance, it appears so small that it is very difficult to recognize. We tested our best proposed method, *i.e.,* LUSS-AE-Adapt, an industrial PIDS (an evoloved form of GOFPID) and method of Nayak *et al.* (2019) on this. We found that our method outperforms the two methods with a huge margin of 26% and 31% respectively. This proves the viability of our method which learns from the data itself in an end-to-end and unsupervised manner. The performance in this dataset validates that our approach is very suitable for real-world perimeter intrusion detection problem. Please refer to Annex A for detailed dataset description, methods analysis and results.

## 5.6 Conclusion

In this chapter, we tackled the problem of detecting video anomalies and intrusions without annotations. To address this, we proposed a novel scheme that leverages unsupervised and self-supervised learning on a single autoencoder. Our method is end-to-end trained on the normal data and jointly learns to discriminate anomalies / intrusions from normality using three chosen tasks: (i) unsupervised video clip reconstruction; (ii) unsupervised future frame prediction; (iii) self-supervised playback rate prediction. The video clip reconstruction task is meant to learn spatio-temporal characteristics of normal videos. The future frame prediction task is designed to learn the propagation of spatio-temporal patterns in the normal videos. Finally, the playback rate prediction task is for strengthening the speed understanding of the encoder. It was the first time when the PRP was adapted for VAD or PID tasks. To focus on anomalous / intrusional regions in the video, we also proposed a new error measure, called the blur pooled error (BPE) and a robust rescaling of anomaly scores.

When the proposed LUSS-AE is applied to VAD task, we outperformed the state-of-the-art methods in three major datasets. The high performance gain in comparison to previous methods on the Shanghai dataset proved the significance of our method. This is because the Shanghai dataset is considered one of most difficult VAD dataset with thirteen different scenes and various anomaly types. Our experiments demonstrate that the chosen tasks enriched the spatio-temporal comprehension of the autoencoder to better understand the normality for detecting anomalies. Our ablation study demonstrated the importance of each task and we found that the PRP task plays a major role in LUSS-AE. Furthermore, a significant boost in performance with respect to MSE, showed the significance of the BPE as it removes the background noise by keeping only the pertinent pixels.

Concerning application of LUSS-AE to PID, we found that we outperform all other methods in the i-LIDS dataset. We found through an ablation study that indeed LUSS-AE leads to a significant improvement in autoencoder performance for PID, but this effect is less visible when we use the adaptive thresholding using moving z-score. This indicates that adaptive thresholding is essential for PIDS when there are long-length videos. We further tested our proposed approach on a real-world industrial dataset having sheeps as distractors and we found that our approach outperforms existing approaches with a huge margin. All this validated the viability of our approach for detecting intrusions.

Overall, we can conclude that our proposed LUSS-AE is generic and can be successfully applied to both VAD and PID tasks and leads to superior performances in both

tasks. However, there is still a room for improvement. Concerning VAD, there are still complicated cases (like anomaly at far distance or anomaly with low spatio-temporal movement), where our method fails. We need to further improve our approach by proposing new proxy tasks, or by using modules like attention. Training network using BPE also needs to be investigated. Concerning the PID task, we have similar shortcomings. Therefore, we need to further improve our approach for obtaining an even better VAD / PID system.

# Conclusion and perspectives

In this thesis, we explored unsupervised deep learning based approaches for learning spatio-temporal representations of videos. These approaches were developed for application in video surveillance domain and in particular for video anomaly detection task and perimeter intrusion detection task. We first formally defined these tasks in chapter 3, along with suitable evaluation protocols. Then, in chapter 4, we proposed an unsupervised autoencoder based method for perimeter intrusion detection. It was coupled with an adaptive thresholding strategy to handle changing scene dynamics. Finally, we also proposed a new generic method in chapter 5. With carefully chosen tasks, it leverages unsupervised and self-supervised learning in an single autoencoder. This approach is applicable to both video anomaly detection and perimeter intrusion detection task. All the proposed approaches were compared with state of the art methods on standard datasets and obtained superior performances.

## Importance of tasks definition and suitable evaluation

Before solving any problem, it is essential to correctly understand it first and this requires defining it as precisely as possible. That is why, in chapter 3, we proposed mathematical definitions for the two main tasks that we address in this thesis along with corresponding evaluation protocols.

Concerning perimeter intrusion definition (PID), we first defined how an intrusion event is triggered by an object. An object causes an intrusion event if it belongs to a non-authorized class (like person, car, *etc.*) and is moving in a protected area during a prohibited time interval. The classes considered as non-authorized, the protected area and the prohibited time interval depend on the end user or client and can vary from one site to another. We can have multiple objects causing intrusion events and these events can also overlap. For example, if a car causes an intrusion event and then after some time, a person also triggers an intrusion event (when car is still there), then these two events overlap. In the case of PID, at any instant, we are concerned if there is an intrusion event, regardless of whether one object or many objects are causing it. Therefore, it is safe to merge intrusion events when they overlap. Consequently, the intrusion event of the whole video is just the union of its intrusion events. The constituent frames of the intrusion event of video are called the intrusion frames. The main concern in PID are the time intervals in video where intrusions occur. We define these time intervals using intrusion intervals, where an intrusion interval corresponds to a contiguous sequence of intrusion frames. The goal of PID task is to detect the beginning of intrusions intervals of the video. The beginning is essential, because if an intrusion is detected too late, then that detection is not very useful in terms of PID requirements (for client/user, the intruder is too close or inside the protected site) (i-LIDS Team, 2006; Buch et Velastin, 2014). This

real-life constraint must be considered by the evaluation protocol. The i-LIDS evaluation protocol adhered to this constraint and counted a detection as valid if it is within the first 10 seconds of actual intrusion. The main problem with this evaluation protocol is that it strongly penalizes the system. It treats detections as false alarms if they occur after 10 seconds from intrusion beginning, even if the alarms are within the intrusion. And while doing so, it does not consider the duration of intrusion. For example, if an intruder enters a site and stay there for 5 minutes and an alarm is triggered at $12^{\text{th}}$ second after the beginning of intrusion. Then, the i-LIDS evaluation protocol will count it as a false alarm, even though the alarm is triggered during the intrusion which would stay in site for almost 5 minutes. We addressed drawbacks of this protocol and introduced an edge level evaluation protocol. Our evaluation protocol adheres with real-life constraints of detecting the beginning of intrusions. Moreover, it has tolerance limits of few frames before and after the intrusion, which permits not to adversely penalize a system if an intrusion is detected a little bit in advance (like a system might detect a person as soon as it steps on the scene, while annotation is when the whole person is in the scene) or when the alarm takes some time to stop after the intrusion.

Defining video anomaly detection is not trivial since anomalies depend on context and they differ from one dataset to another. In this thesis, we defined VAD on the basis of available annotations. If annotations of both normal and abnormal events are present during training, then the VAD problem boils down to binary classification, *i.e.,* understanding the two classes (normal and anomalies) with training and then detecting anomalies during testing. However, realistically, it is very difficult to have annotations during training since anomaly is a rare event and it is practically impossible to have annotations for all possible future anomalous activities. Therefore, the second case is more realistic, where it is considered that training data primarily contains normal (non-anomalous) videos. We mathematically describe this case using the probably approximately correct (PAC) learning framework and consequently anomalies are defined as the complement of normality distribution present in the training data. This definition takes into account the anomaly context, since anomalies are defined according to the normal distribution provided with the dataset. Furthermore, our definition takes care of debated cases for VAD with the uncertainty parameter of PAC framework. For example, in ShanghaiTech dataset, normal activities like eating ice-creams are present only in the testing set and not the training set. Therefore, even though there is high probability for them to be abnormal, we cannot be fully certain that they are anomaly due to the uncertainty parameter. For evaluating VAD methods, we used the frame level binary class evaluation. This is done to be comparable with the state-of-the-art methods, all of which use this evaluation protocol.

Even though our PAC formulation addressed some complicated cases in VAD, there is still a need for a new dataset with more precise anomaly annotation (during testing). The training data should be representative of normalities and any deviation from it should define some degree of anomaliness. Concerning evaluation, anomalies are also events and like PID, we want to finally know if these events are detected or not. Frame-level evaluation does not provide much information for anomalous event detection. For example, with frame level evaluation, we cannot differentiate between a method that detected some frames of all anomalous events, with the one that detected only few events having large number of frames. Therefore, an event level evaluation should also be used in VAD. For this, datasets like Shanghai and Avenue should be manually annotated at event level. Concerning event level evaluation, we focused on detecting the beginning of event, but our proposed protocol can be extended for detecting the complete event if required. In

some cases, researchers might want to detect if the alarm is raised during the whole event. In that case, our protocol cannot work since it is dedicated to detect rising edges and not the frames of an event. Therefore, a suitable evaluation protocol should be developed in that case. Similarly, we did not penalize re-detections in our evaluation as long as it is during the event. But one can penalize them to some extent, depending on the system output requirements.

## Unsupervised spatio-temporal video understanding

In this thesis, we explored spatio-temporal learning in videos for unsupervised tasks. We first proposed a strided 3D convolutional autoencoder (S3DCAE) for this, in chapter 4. To learn spatio-temporal features in a joint manner from videos, we used only 3D convolutions and 3D deconvolutions. The encoder layers generally use pooling operation for compressing information, while decoder layers use the unpooling operation. We instead proposed to use strided convolutions and deconvolutions, therefore our architecture was light in memory and fast in execution. The proposed autoencoder learns normality during training as it is trained only on normal videos and detects intrusions as they are badly reconstructed (having high reconstruction error) during testing. Therefore, it is capable to understand spatio-temporal normality and detect intrusions (or even anomalies) in an unsupervised way.

The second method that we proposed, called LUSS-AE, leverages unsupervised and self-supervised learning using a single autoencoder for spatio-temporal feature learning. It was proposed in Chapter 5. Like the previous approach, the aim is to learn normality during training, for detecting anomalies/intrusions during testing. The main idea is to enrich the spatio-temporal comprehension of an autoencoder using different tasks. We proposed three tasks: video clip reconstruction (VCR), future frame prediction (FFP) and playback rate perception (PRP). VCR is meant to learn spatio-temporal characteristics, FFP is designed to learn the propagation of spatio-temporal patterns, and PRP is intended to strengthen the speed understanding of the encoder. We demonstrated that these tasks enhanced the spatio-temporal comprehension of the autoencoder. With successful application on both VAD and PID tasks, we showed its effectiveness and generic behavior.

Notice that one issue with LUSS-AE is that the PRP task has only two classes: original and accelerated. We group different speeds like 2x, 3x and 4x into a single class (accelerated). Semantically, these different speeds have different spatio-temporal information and therefore they should be in different classes. We can further add cases like slow speed (0.5x) and even negative speed, implying reverse playback. This will make LUSS-AE even more speed aware. Another idea could be to use speed regression instead of speed classification. In real life, often the speed cannot be discretely categorized as 2x or 3x, *etc.*, but can be represented by a continuous value. In speed classification, each speed like 2x or 3x, were treated independently but there is a semantic relation among speeds, *e.g.,* speed of 2x and 3x are more closer to each other than speed of 5x. The speed regression will permit the network to understand the semantics among different speeds and it should enrich its spatio-temporal comprehension by making it even more speed aware. Finally, more tasks can be added to our AE like backward / forward video clip prediction, masking input and predicting it, input flip or rotation prediction, *etc.* Attention modules can further enhance the spatio-temporal comprehension, but they do require extra computational / memory capacity. There is also a new family of methods that use a transformer network

(Bertasius *et al.*, 2021; Neimark *et al.*, 2021). These methods are now used extensively in spatio-temporal video learning, especially in self-supervised learning, as a pre-training step. However, they need an very large amount of GPU / CPU computational capacity and long time to train. For instance, video transformer network (Neimark *et al.*, 2021) has 114 million parameters with 4,218 GFLOPs, while our proposed LUSS-AE for VAD has 6.12 million parameters with 16.30 GFLOPs only. Even though smaller and quicker variants of this new family of methods are coming, we are still far to have a model that runs for real-time tasks like video anomaly detection and perimeter intrusion detection.

## Adaptiveness for online outdoor event detection

Since we focused on detecting intrusion or anomalies in the outdoor environment, the obvious challenge is to face changing scene dynamics like weather, temperature, lighting conditions, *etc.* Any detection system must adapt itself to these changes, otherwise it will underperform. This challenge is particularly dominant for long length videos, like we have in the i-LIDS dataset. To address this challenge, we proposed an adaptive mechanism using a moving z-score on the reconstruction error of the autoencoder. It detected intrusions as soon as they occur and then adapted itself with the scene dynamics. It is generic and brought massive gains in performance to both the network architectures we proposed in this thesis. The LUSS-AE with adaptive thresholding (LUSS-AE-Adapt) was also tested in a private dataset. This dataset is very challenging since majority of time, there are sheeps moving in the scene, which is considered a normal behavior and intrusions are caused by human intruders entering the scene in various ways like crawling, rolling, *etc.* It was filmed with thermal camera, which makes it even more difficult to distinguish intrusions with sheeps. Our adaptive LUSS-AE method outperformed an object detector based method and a private industrial method by a large margin. This validated the importance of adaptiveness.

Notice that when a VAD or PID system is deployed in real-life, it does need to face these changing scene dynamics. To develop these adaptive approaches, we need appropriate datasets. Concerning PID, the i-LIDS dataset does conform to this need, but it lacks different types of intrusions like cars, trucks, skateboard, *etc.*. Furthermore, the two views of this dataset are very similar to each other, they are like mirror images of each other. Also, the amount of distractions is very low in this dataset and thus any person detector system can work well. Therefore, we need more public PID datasets which address the above mentioned weaknesses. Even though we also tested our adaptive approach on a challenging dataset, unfortunately it cannot be made public. In regard to VAD, it is unfortunate that we do not have long length test videos in any current VAD dataset. This would have allowed the development of approaches that adapts themselves with time. Thus, for development of more adaptive approaches, we would require more realistic VAD and PID datasets. Furthermore, different types of camera sensors like thermal, depth, *etc.* and multiple viewpoints could further help both VAD and PID approaches.

## Towards more generic and deployable models

Our proposed approaches learn in offline mode from training data to detect events during testing. When these approaches are installed in a site, they need to update themselves with the newly observed data, *i.e.,* learning new normalities on the go. Naively updating model

weights with newly observed data can lead to catastrophic forgetting (Kirkpatrick *et al.*, 2017). Another solution is to gather data and re-train the models but this would quickly become infeasible due to computational and storage needs. Thus, we need to preserve the initial knowledge of models without reusing the initial data, and also acclimate the model to newly observed data. To accomplish this, continual learning can be used. Recently, it has been successfully used for VAD (Doshi et Yilmaz, 2020, 2022). The approaches we proposed are trained on each dataset separately. But how can we use a model trained on one dataset to work on another? Ideally, we would like to just use few frames of the new dataset to adapt the current model. This is precisely what few shot learning does (Lake *et al.*, 2015; Finn *et al.*, 2017). Recently, some works have used it in VAD too (Lu *et al.*, 2020; Lv *et al.*, 2021). This property is extremely desirable since then, we can use a model trained on an offline dataset to a number of sites and adapt to each one of them. As already mentioned, we can improve our model with more suitable tasks. However, the new trend in computer vision is to use transformer like model due to their immense generalization capacity. In future, smaller and faster transformer like models can be used in video surveillance.

# Annexes

## Contents

# A Testing PIDS in a real-world case

So far, we have evaluated different methods on the i-LIDS dataset. As already mentioned before, this dataset is interesting but it still has some major drawbacks, which makes it an easy task for any simple person detection. The intrusion in real-life are not so easy to detect and furthermore they can be very well non-human entity like wild animals or cars. To address this, we evaluate methods on a private dataset that depicts more challenging conditions like very dynamic background, different distances from camera, change in light and scene dynamics and hence it is more inline with recent challenges in video surveillance.

## Sheeps dataset

This dataset contains videos of a site surveilled by a thermal camera that principally contains sheep. The videos are recorded for many days and therefore present varying light conditions in the scene. Some illustrative frames are shown in Figure A.1.


Figure A.1: Example frames from the Sheeps dataset without intrusions. The images show video frames from different time of the day. The sheeps present are moving in random manner. Also, observe the sheeps at the top-right end of the third image.

The intrusions in this dataset are very rare and caused by people. In other words, when some person enters the scene, it needs to be detected. The human intruders enter in various ways like running, walking, crawling, *etc.* Furthermore, they are situated at different distances from the camera like near, middle and far. Overall, the task is hard since the PIDS method need to ignore sheeps but at the same time detect human intruders who often act like sheep. Some illustrative examples of intrusions in this dataset are shown in Figure A.2.

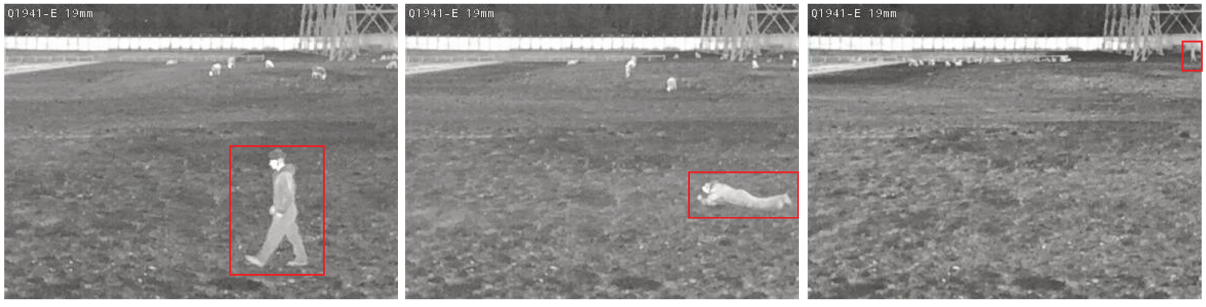



Figure A.2: Example frames from the Sheeps dataset with intrusions shown in red-bounding boxes. The images are shown in increasing level of intrusion detection difficulty.

The videos in this dataset are captured at 5 FPS with frame resolution of 384x288. Since intrusions are very rare in this dataset, the training set does not contain intrusions. The training set contains 50 videos (approx. 20,000 frames) with sheeps at different configurations and having different lighting conditions. In test set, each video contain atleast one intrusion. The intrusion can be at different distance from the camera and depicts human crawling, running, *etc.*, accompanied with varied lighting conditions and sheeps. We have in total 36 videos (11,189 frames) of intrusions (7,077 frames). Since each video has different level of difficulty for detecting intrusion, the dataset also comes with video level label from 1 to 5, where 1 is the easiest and 5 is the most difficult case.

## Methods and experimental setup

We compare three methods on this dataset: LUSS-AE-Adapt, Nayak *et al.* (2019) and an industrial PIDS. We chose LUSS-AE-Adapt because it has enhanced spatio-temporal comprehension and performs better than AE-Adapt. Instead of using GOFPID, we use a private industrial PIDS. This is because their functioning is similar (follows similar traditional pipeline) but industrial PIDS performs better than GOFPID and thus is closer to the state of the art practiced by industry. For LUSS-AE-Adapt, we use the same architecture as proposed in Chapter 5. Just the input frame size is updated to 384x288. Other settings remain the same as mentioned in Section 5.5.1. For (Nayak *et al.*, 2019) also, the same setting like before (see Section 4.3.1) as used with human as the intrusion class. The industrial PIDS was also used with appropriate settings. For evaluation, we use the proposed edge-level protocol (EL (10 s)), as we concluded in Chapter 4 that it should be used for evaluating PID methods. For better comprehension, we also show results for some levels of difficulty.

## Results

Table 5.8 presents results of the three methods on the whole Sheeps dataset. We can observe that our proposed method has outperformed the other two methods with a large margin. The $F_1$ score is almost double in comparison to Nayak *et al.* (2019) and has a huge margin **0.26** in comparison to industrial PIDS. We can also observe that our method also has the best precision and recall values. This is important because Nayak *et al.* (2019) also has a good recall value but it is at the cost of high number of false alarms, depicted by the poor precision value.

| Methods | Precision | Recall | $F_1$ |
|---|---|---|---|
| Nayak *et al.* (2019) | 0.23 | 0.78 | 0.36 |
| Industrial PIDS | 0.32 | 0.57 | 0.41 |
| LUSS-AE-Adapt | **0.57** | **0.81** | **0.67** |

Table 5.8: PID methods comparison based on edge-level (EL) evaluation at 10 seconds in the Sheeps dataset.

To better understand the results, we also provide results at different difficulty levels. For sake of simplicity, we provide the results of only three difficulties, *i.e.,* 1, 2 and 5. The other difficulty have similar results, meaning when difficulty increases performance often

decreases. The difficulty 1 and 2 correspond to easier cases, while the difficulty of level 5 corresponds to the most difficult case (shown in rightmost image of Figure A.2).

| Method | Difficulty - 1 | | | Difficulty - 2 | | | Difficulty - 5 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| Nayak *et al.* (2019) | 0.21 | 1 | 0.35 | 0.25 | 1 | 0.40 | 0.20 | 1 | 0.33 |
| Industrial PIDS | 0.45 | 0.83 | 0.58 | 0.50 | 0.77 | 0.60 | 0.10 | 0.22 | 0.14 |
| LUSS-AE-Adapt | 0.86 | 1 | **0.92** | 0.73 | 0.89 | **0.80** | 0.71 | 0.56 | **0.63** |

Table 5.9: PID methods comparison at different difficulty levels in the Sheeps dataset, where edge-level (EL) evaluation at 10 seconds is used.

Table 5.9 shows the methods comparison based on different difficulty. We observe that the method of Nayak *et al.* (2019) performs similar for the three distances. In fact, the person object detector seems to fail for distinguishing between person and sheep. That is why, there is a recall of 1 and very low precision. In other words, their system is always raising intrusion alarms (thus perfect recall), irrespective of intruder or sheep, and thus have a very high number of false alarms, depicted with very low precision value. The industrial PIDS performs better than Nayak *et al.* (2019). For difficulty 1 and 2, their precision values are better than the last method, meaning they have fewer false alarms. For difficulty 5, their method performs poorly and is not able to different between intruders and sheeps. For our proposed LUSS-AE-Adapt method, we observe that it performs better than the two methods irrespective of the difficulty. We obtain an excellent $F_1$ score of **0.92** for difficulty 1, which is very interesting for this difficult PID dataset. We also observe that the $F_1$ value decreases with increasing difficulty. The main component of this decrease in scores is due to the recall values. This means that with increase in difficulty, we detect lesser intrusions but we do not raise a lot of false alarms.

## Conclusion

In this annex, we tested our proposed PIDS with different existing methods on a private dataset. This dataset is very interesting because it contains constant movement of sheeps (distractors) in a scene, where human intruder tries to enter in various ways. The change in lighting conditions, arbitrary movement of sheeps, different distances of intruder from the camera, and various ways in which the intruder enters the scene, makes this dataset very challenging. The method of Nayak *et al.* (2019), which has excellent performance in i-LIDS dataset, performs poorly here. This is precisely because their object detector failed to detect human intruder among sheeps. To be precise, their detector mostly detected some sheep as human, thus causing massive false alarms. Therefore, here we observed the risk of using pre-trained detectors for PIDS. Concerning the private industrial PIDS, it performed a bit better than Nayak *et al.* (2019). However, it still fails to detect intrusions at difficult cases and often confuse sheeps with intruders. We found that our proposed LUSS-AE-Adapt method outperformed the two other methods by a good margin. Since our method learned normality, it was able to better differentiate between sheeps and humans. Our adaptive method has an excellent result for easier cases and even with increase in difficulty, our method does not trigger a large number of false alarms. One of the most important aspect is that our method is completely without supervision and learns

to detect intrusions just from the unlabeled data. Overall, we clearly see the viablity of our method in real-world perimeter intrusion detection task, provided we have a dataset to train the method beforehand.

# Bibliography


Andra Acsintoae, Andrei Florescu, Mariana-Iuliana Georgescu, Tudor Mare, Paul Sumedrea, Radu Tudor Ionescu, Fahad Shahbaz Khan et Mubarak Shah : Ubnormal: New benchmark for supervised open-set video anomaly detection. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20143–20153, 2022.

Amit Adam, Ehud Rivlin, Ilan Shimshoni et Daviv Reinitz : Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE transactions on pattern analysis and machine intelligence*, 30(3):555–560, 2008.

Sos S Agaian, Blair Silver et Karen A Panetta : Transform coefficient histogram-based image enhancement algorithms using contrast entropy. *IEEE Trans Image Process*, 16:741–758, 2007.

Samet Akcay, Amir Atapour-Abarghouei et Toby P Breckon : Ganomaly: Semi-supervised anomaly detection via adversarial training. *In Asian conference on computer vision*, pages 622–637. Springer, 2018.

Nijad Al-Najdawi, Helmut E. Bez, Jyoti Singhai et Eran.A. Edirisinghe : A survey of cast shadow detection algorithms. *Pattern Recognition Letters*, 33(6):752–764, 2012. ISSN 0167-8655. URL `https://www.sciencedirect.com/science/article/pii/S0167865511004326`.

Walid S Ibrahim Ali : Real time video sharpness enhancement by wavelet-based luminance transient improvement. *In ISSPA*, pages 1–4, 2007.

G Aravamuthan, P Rajasekhar, RK Verma, SV Shrikhande, S Kar et Suresh Babu : Physical intrusion detection system using stereo video analytics. *In CVIP*, pages 173–182, 2020.

M Astrid, M Z Zaheer, J-Y Lee et S-I Lee : Learning not to reconstruct anomalies. *In BMVC*, 2021a.

M Astrid, M Z Zaheer et S-I Lee : Synthetic temporal anomaly guided end-to-end video anomaly detection. *In ICCV*, pages 207–214, 2021b.

AVSS dataset : `http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_d.html`, 2007.

AXIS Communications : AXIS Perimeter Defender - User Manual M14.3, 2020.

Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia et Atilla Baskurt : Sequential deep learning for human action recognition. *In International workshop on human behavior understanding*, pages 29–39. Springer, 2011.

Dzmitry Bahdanau, Kyung Hyun Cho et Yoshua Bengio : Neural machine translation by jointly learning to align and translate. *In 3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Nicolas Ballas, Li Yao, Chris Pal et Aaron C Courville : Delving deeper into convolutional networks for learning video representations. *In ICLR (Poster)*, 2016.

Olivier Barnich et Marc Van Droogenbroeck : ViBe: a powerful random technique to estimate the background in video sequences. *In ICASSP*, pages 945–948, 2009.

Q Barthélemy, L Mayaud, D Ojeda et M Congedo : The Riemannian potato field: a tool for online signal quality index of EEG. *IEEE Trans Neural Syst Rehabil Eng*, 27:244–255, 2019.

Quentin Barthélemy : PyGOFPID is a Python package for good old fashioned perimeter intrusion detection systems for video protection. `https://github.com/qbarthelemy/PyGOFPID`, 2022.

Steven S. Beauchemin et John L. Barron : The computation of optical flow. *ACM Comput Surv*, 27(3):433–466, 1995.

S Benaim, A Ephrat, O Lang, I Mosseri, W T Freeman, M Rubinstein, M Irani et T Dekel : SpeedNet: Learning the speediness in videos. *In CVPR*, pages 9922–9931, 2020.

Eric P Bennett et Leonard McMillan : Video enhancement using per-pixel virtual exposures. *In ACM SIGGRAPH*, pages 845–852. 2005.

Gedas Bertasius, Heng Wang et Lorenzo Torresani : Is space-time attention all you need for video understanding? *In ICML*, volume 2, page 4, 2021.

Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos et Ben Upcroft : Simple online and realtime tracking. *In ICIP*, pages 3464–3468, 2016.

Christopher M Bishop et Nasser M Nasrabadi : *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Léon Bottou *et al.* : Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

Thierry Bouwmans : Recent advanced statistical background modeling for foreground detection-a systematic survey. *Recent Patents on Computer Science*, 4:147–176, 2011.

Thierry Bouwmans : Traditional and recent approaches in background modeling for foreground detection: An overview. *Computer science review*, 11:31–66, 2014.

Thierry Bouwmans, Fida El Baf et Bertrand Vachon : Background modeling using mixture of gaussians for foreground detection-a survey. *Recent patents on computer science*, 1(3):219–237, 2008.

Sebastian Brutzer, Benjamin Höferlin et Gunther Heidemann : Evaluation of background subtraction techniques for video surveillance. *In CVPR*, pages 1937–1944, 2011.

N Buch et SA Velastin : Human intrusion detection using texture classification in real-time. *Tracking Humans for the Evaluation of their Motion in Image Sequences*, page 1, 2008.

Norbert Buch et Sergio A Velastin : Local feature saliency classifier for real-time intrusion monitoring. *Opt Eng*, 53:073108, 2014.

Quin Cai et Jake K Aggarwal : Tracking human motion in structured environments using a distributed-camera system. *IEEE Trans Pattern Anal Mach Intell*, 21:1241–1247, 1999.

Joao Carreira et Andrew Zisserman : Quo vadis, action recognition? a new model and the kinetics dataset. *In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

Eduardo Cermeño, Ana Pérez et Juan Alberto Sigüenza : Intelligent video surveillance beyond robust background modeling. *Expert Syst Appl*, 91:138–149, 2018.

R Chalapathy et S Chawla : Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

Varun Chandola, Arindam Banerjee et Vipin Kumar : Anomaly detection: A survey. *ACM Comput Surv*, 41(3):1–58, 2009.

Y Chang, Z Tu, W Xie et J Yuan : Clustering driven deep autoencoder for video anomaly detection. *In ECCV*, pages 329–345, 2020.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan et Ilya Sutskever : Generative pretraining from pixels. *In International conference on machine learning*, pages 1691–1703. PMLR, 2020.

Peihao Chen, Deng Huang, Dongliang He, Xiang Long, Runhao Zeng, Shilei Wen, Mingkui Tan et Chuang Gan : Rspnet: Relative speed perception for unsupervised video representation learning. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1045–1053, 2021.

Kai-Wen Cheng, Yie-Tarng Chen et Wen-Hsien Fang : Video anomaly detection and localization using hierarchical feature representation and gaussian process regression. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2909–2917, 2015.

MyeongAh Cho, Taeoh Kim, Woo Jin Kim, Suhwan Cho et Sangyoun Lee : Unsupervised video anomaly detection via normalizing flows with implicit latent features. *Pattern Recognit*, 129:108703, 2022.

Yong Shean Chong et Yong Haur Tay : Abnormal event detection in videos using spatiotemporal autoencoder. *In International Symposium on Neural Networks*, pages 189–196. Springer, 2017.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho et Yoshua Bengio : Empirical evaluation of gated recurrent neural networks on sequence modeling. *In NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

James L Crowley, P Reignier et S Pesnel : Caviar context aware vision using image-based active recognition. Rapport technique, University of Edinburgh, 2005.

Navneet Dalal et Bill Triggs : Histograms of oriented gradients for human detection. *In CVPR*, volume 1, pages 886–893, 2005.

Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall et Luc Van Gool : Spatio-temporal channel correlation networks for action classification. *In Proceedings of the European Conference on Computer Vision (ECCV)*, pages 284–299, 2018.

Ali Diba, Mohsen Fayyaz, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousefzadeh et Luc Van Gool : Temporal 3d convnets: New architecture and transfer learning for video classification. *arXiv preprint arXiv:1711.08200*, 2017a.

Ali Diba, Vivek Sharma et Luc Van Gool : Deep temporal linear encoding networks. *In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2329–2338, 2017b.

Nikolaos Dimitriou, George Kioumourtzis, Anargyros Sideris, Georgios Stavropoulos, Evdoxia Taka, Nikolaos Zotos, George Leventakis et Dimitrios Tzovaras : An integrated framework for the timely detection of petty crimes. *In EISIC*, pages 24–31, 2017.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko et Trevor Darrell : Long-term recurrent convolutional networks for visual recognition and description. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

Fei Dong, Yu Zhang et Xiushan Nie : Dual discriminator generative adversarial network for video anomaly detection. *IEEE Access*, 8:88170–88176, 2020.

Keval Doshi et Yasin Yilmaz : Continual learning for anomaly detection in surveillance videos. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 254–255, 2020.

Keval Doshi et Yasin Yilmaz : Online anomaly detection in surveillance videos with asymptotic bound on false alarm rate. *Pattern Recognition*, 114:107865, 2021.

Keval Doshi et Yasin Yilmaz : Rethinking video anomaly detection-a continual learning approach. *In Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3961–3970, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly *et al.* : An image is worth 16x16 words: Transformers for image recognition at scale. *In International Conference on Learning Representations*, 2020.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers et Thomas Brox : Flownet: Learning optical flow with convolutional networks. *In Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

Wenbin Du, Yali Wang et Yu Qiao : Rpan: An end-to-end recurrent pose-attention network for action recognition in videos. *In Proceedings of the IEEE international conference on computer vision*, pages 3725–3734, 2017.

John Duchi, Elad Hazan et Yoram Singer : Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Vincent Dumoulin et Francesco Visin : A guide to convolution arithmetic for deep learning. *stat*, 1050:23, 2016.

Debidatta Dwibedi, Pierre Sermanet et Jonathan Tompson : Temporal reasoning in videos using convolutional gated recurrent units. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1111–1116, 2018.

Kiana Ehsani, Roozbeh Mottaghi et Ali Farhadi : Segan: Segmenting and generating the invisible. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6144–6153, 2018.

Ahmed Elgammal, David Harwood et Larry Davis : Non-parametric model for background subtraction. *In ECCV*, pages 751–767, 2000.

Ersin Esen, Mehmet Ali Arabaci et Medeni Soysal : Fight detection in surveillance videos. *In 2013 11th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 131–135. IEEE, 2013.

T Fawcett : An introduction to ROC analysis. *Pattern Recognit Lett*, 27:861–874, 2006.

William Fedus, Barret Zoph et Noam Shazeer : Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik et Kaiming He : Slowfast networks for video recognition. *In Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

Christoph Feichtenhofer, Axel Pinz et Richard P Wildes : Spatiotemporal multiplier networks for video action recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4768–4777, 2017.

Christoph Feichtenhofer, Axel Pinz et Andrew Zisserman : Convolutional two-stream network fusion for video action recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016.

Pedro Felzenszwalb, David McAllester et Deva Ramanan : A discriminatively trained, multiscale, deformable part model. *In CVPR*, pages 1–8, 2008.

J-C Feng, F-T Hong et W-S Zheng : MIST: Multiple instance self-training framework for video anomaly detection. *In CVPR*, pages 14009–14018, 2021.

Chelsea Finn, Pieter Abbeel et Sergey Levine : Model-agnostic meta-learning for fast adaptation of deep networks. *In International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Denis Fortun, Patrick Bouthemy et Charles Kervrann : Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.

FOXSTREAM Smart Video Analytics : FoxVigi 3.71.0 software user guide, 2021.

Zhouyu Fu, Weiming Hu et Tieniu Tan : Similarity based vehicle trajectory clustering and anomaly detection. *In IEEE International Conference on Image Processing 2005*, volume 2, pages II–602. IEEE, 2005.

Kunihiko Fukushima et Sei Miyake : Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. *In Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

Rikke Gade et Thomas B Moeslund : Thermal cameras and applications: a survey. *Mach Vis Appl*, 25:245–262, 2014.

Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis *et al.* : Event-based vision: A survey. *IEEE Trans Pattern Anal Mach Intell*, 44:154–180, 2020.

Hongwei Ge, Zehang Yan, Wenhao Yu et Liang Sun : An attention mechanism based convolutional lstm network for video action recognition. *Multimedia Tools and Applications*, 78(14):20533–20556, 2019.

M-I Georgescu, A Barbalau, R T Ionescu, F S Khan, M Popescu et M Shah : Anomaly detection in video via self-supervised and multi-task learning. *In CVPR*, pages 12742–12752, 2021a.

M-I Georgescu, R Ionescu, F S Khan, M Popescu et M Shah : A background-agnostic framework with adversarial training for abnormal event detection in video. *IEEE Trans Pattern Anal Mach Intell*, 2021b.

D Gong, L Liu, V Le, B Saha, M R Mansour, S Venkatesh et A van den Hengel : Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. *In ICCV*, pages 1705–1714, 2019.

Ian Goodfellow, Yoshua Bengio et Aaron Courville : Deep learning mit press (2016). *In Conference on information and communication systems (ICICS)*, pages 151–156, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville et Yoshua Bengio : Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Alex Graves, Navdeep Jaitly et Abdel-rahman Mohamed : Hybrid speech recognition with deep bidirectional lstm. *In 2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

A Gudi, F Büttner et J van Gemert : Proximally sensitive error for anomaly detection and feature learning. *arXiv preprint arXiv:2206.00506*, 2022.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu *et al.* : A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

Ismail Haritaoglu, David Harwood et Larry S. Davis : W4: real-time surveillance of people and their activities. *IEEE Trans Pattern Anal Mach Intell*, 22:809–830, 2000.

Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury et Larry S Davis : Learning temporal regularity in video sequences. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 733–742, 2016.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman et Jerome H Friedman : *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Jamie Hayes, Luca Melis, George Danezis et Emiliano De Cristofaro : Logan: Evaluating privacy leakage of generative models using generative adversarial networks. 05 2017.

Simon Haykin et N Network : A comprehensive foundation. *Neural networks*, 2 (2004):41, 2004.

Chengkun He, Jie Shao et Jiayu Sun : An anomaly-introduced learning method for abnormal event detection. *Multimedia Tools and Applications*, 77(22):29573–29588, 2018.

Ryota Hinami, Tao Mei et Shin'ichi Satoh : Joint detection and recounting of abnormal events by learning deep generic knowledge. *In Proceedings of the IEEE international conference on computer vision*, pages 3619–3627, 2017.

Geoffrey E Hinton et Richard Zemel : Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

Sepp Hochreiter et Jürgen Schmidhuber : Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Berthold KP Horn et Brian G Schunck : Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

Weiming Hu, Tieniu Tan, Liang Wang et Steve Maybank : A survey on visual surveillance of object motion and behaviors. *IEEE Trans Syst Man Cybern C Appl Rev*, 34:334–352, 2004.

Xianwei Hu, Tie Li, Zongzhi Wu, Xuan Gao et Zhiqiang Wang : Research and application of intelligent intrusion detection system with accuracy analysis methodology. *Infrared Phys Technol*, 88:245–253, 2018.

Xing Hu, Shiqiang Hu, Yingping Huang, Huanlong Zhang et Hanbing Wu : Video anomaly detection using deep incremental slow feature analysis network. *IET Computer Vision*, 10(4):258–267, 2016.

Jing Huo, Yang Gao, Wanqi Yang et Hujun Yin : Abnormal event detection via multi-instance dictionary learning. *In International conference on intelligent data engineering and automated learning*, pages 76–83. Springer, 2012.

Noureldien Hussein, Efstratios Gavves et Arnold WM Smeulders : Timeception for complex action recognition. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 254–263, 2019.

i-LIDS Team : Imagery library for intelligent detection systems (i-LIDS); a standard for testing video based detection systems. *In IET Conference on Crime and Security*, pages 445–448, 2006.

Sutrisno Warsono Ibrahim : A comprehensive review on intelligent surveillance systems. *Communications in Science and Technology*, 1, 2016.

Radu Tudor Ionescu, Fahad Shahbaz Khan, Mariana-Iuliana Georgescu et Ling Shao : Object-centric auto-encoders and dummy anomalies for abnormal event detection in video. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7842–7851, 2019.

Shuiwang Ji, Wei Xu, Ming Yang et Kai Yu : 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

Fan Jiang, Ying Wu et Aggelos K Katsaggelos : Detecting contextual anomalies of crowd motion in surveillance video. *In 2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1117–1120. IEEE, 2009.

L. Jiao, R. Zhang, F. Liu, S. Yang, B. Hou, L. Li et X. Tang : New generation deep learning for video object detection: A survey. *IEEE Trans Neural Netw Learn Syst*, Early Access:1–21, 2021.

Longlong Jing et Yingli Tian : Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058, 2020.

Michael I Jordan : Attractor dynamics and parallelism in a connectionist sequential machine. *In Artificial neural networks: concept learning*, pages 112–127. 1990.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar et Li Fei-Fei : Large-scale video classification with convolutional neural networks. *In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

Phulpreet Kaur, M Gangadharappa et Shalu Gautam : An overview of anomaly detection in video surveillance. *In 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 607–614. IEEE, 2018.

Jacob Devlin Ming-Wei Chang Kenton et Lee Kristina Toutanova : Bert: Pre-training of deep bidirectional transformers for language understanding. *In Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan et Mubarak Shah : Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

Jong Sun Kim, Dong Hae Yeom et Young Hoon Joo : Fast and robust algorithm of tracking multiple moving objects for intelligent video surveillance systems. *IEEE Trans Consum Electron*, 57:1165–1170, 2011.

Junbong Kim, Kwanghee Jeong, Hyomin Choi et Kisung Seo : Gan-based anomaly detection in imbalance problems. *In European Conference on Computer Vision*, pages 128–145. Springer, 2020.

Pil-Soo Kim, Dong-Gyu Lee et Seong-Whan Lee : Discriminative context learning with gated recurrent unit for group activity recognition. *Pattern Recognition*, 76:149–161, 2018a.

Seung Hyun Kim, Su Chang Lim *et al.* : Intelligent intrusion detection system featuring a virtual fence, active intruder detection, classification, tracking, and action recognition. *Ann Nucl Energy*, 112:845–855, 2018b.

Sung-Yeol Kim, Manbae Kim et Yo-Sung Ho : Depth image filter for mixed and noisy pixel removal in RGB-D camera systems. *IEEE Trans Consum Electron*, 59:681–689, 2013.

D P Kingma et J Ba : Adam: A method for stochastic optimization. *In ICLR*, 2015.

B R Kiran, D M Thomas et R Parakkal : An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *J Imaging*, 4:36, 2018.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj et Daniel J Inman : 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska *et al.* : Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Yongqiang Kong, Jianhui Huang, Shanshan Huang, Zhengang Wei et Shengke Wang : Learning spatiotemporal representations for human fall detection in surveillance video. *Journal of Visual Communication and Image Representation*, 59:215–230, 2019.

Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton : Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Brenden M Lake, Ruslan Salakhutdinov et Joshua B Tenenbaum : Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Zhenzhong Lan, Yi Zhu, Alexander G Hauptmann et Shawn Newsam : Deep local video feature for action recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2017.

Federico Landi, Cees GM Snoek et Rita Cucchiara : Anomaly locality in video surveillance. *arXiv preprint arXiv:1901.10364*, 2019.

Viet-Tuan Le et Yong-Guk Kim : Attention-based residual autoencoder for video anomaly detection. *Applied Intelligence*, pages 1–15, 2022.

Yann Le Cun, Leon Bottou et Yoshua Bengio : Reading checks with multilayer graph transformer networks. *In 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 151–154. IEEE, 1997.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang *et al.* : Photo-realistic single image super-resolution using a generative adversarial network. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

Sangmin Lee, Hak Gu Kim et Yong Man Ro : Stan: Spatio-temporal adversarial networks for abnormal event detection. *In ICASSP*, pages 1323–1327. IEEE, 2018.

Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng et Shuicheng Yan : Perceptual generative adversarial networks for small object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017.

Jun Li, Xianglong Liu, Mingyuan Zhang et Deqing Wang : Spatio-temporal deformable 3d convnets with attention for action recognition. *Pattern Recognition*, 98:107037, 2020.

Nanjun Li et Faliang Chang : Video anomaly detection and localization via multivariate gaussian fully convolution adversarial autoencoder. *Neurocomputing*, 369:92–105, 2019.

Qing Li, Zhaofan Qiu, Ting Yao, Tao Mei, Yong Rui et Jiebo Luo : Action recognition by learning deep multi-granular spatio-temporal video representation. *In Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 159–166, 2016.

Shuo Li, Fang Liu et Licheng Jiao : Self-training multi-sequence learning with transformer for weakly supervised video anomaly detection. *In AAAI*, 2022.

W Li, V Mahadevan et N Vasconcelos : Anomaly detection and localization in crowded scenes. *IEEE Trans Pattern Anal Mach Intell*, 36:18–32, 2013.

Xiaoli Li et Ze-min Cai : Anomaly detection techniques in surveillance videos. *In 2016 9th International congress on image and signal processing, BioMedical engineering and informatics (CISP-BMEI)*, pages 54–59. IEEE, 2016.

Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain et Cees GM Snoek : Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018.

KM Liang, HW Hon, MJ Khairunnisa, TL Choong et HZ Khairil : Real time intrusion detection system for outdoor environment. *In ISCAIE*, pages 147–152, 2012.

P Lichtsteiner, C Posch et T Delbruck : A 128x128 120 db 15 µs latency asynchronous temporal contrast vision sensor. *IEEE J Solid-State Circuits*, 43:566–576, 2008.

Ji Lin, Chuang Gan et Song Han : Tsm: Temporal shift module for efficient video understanding. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7083–7093, 2019.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie : Feature pyramid networks for object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C Lawrence Zitnick : Microsoft coco: Common objects in context. *In European conference on computer vision*, pages 740–755. Springer, 2014.

Zachary C Lipton, John Berkowitz et Charles Elkan : A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Kun Liu et Huadong Ma : Exploring background-bias for anomaly detection in surveillance videos. *In Proceedings of the 27th ACM International Conference on Multimedia*, pages 1490–1499, 2019.

W Liu, W Luo, D Lian et S Gao : Future frame prediction for anomaly detection–a new baseline. *In CVPR*, pages 6536–6545, 2018.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C Berg : Ssd: Single shot multibox detector. *In European conference on computer vision*, pages 21–37. Springer, 2016.

Wen Liu, Weixin Luo, Zhengxin Li, Peilin Zhao, Shenghua Gao *et al.* : Margin learning embedded prediction for video anomaly detection with a few anomalies. *In IJCAI*, pages 3023–3030, 2019.

X Liu, F Zhang, Z Hou, L Mian, Z Wang, J Zhang et J Tang : Self-supervised learning: Generative or contrastive. *IEEE Trans Knowl Data Eng*, 2021a.

Z Liu, Y Nie, C Long, Q Zhang et G Li : A hybrid video anomaly detection framework via memory-augmented flow reconstruction and flow-guided frame prediction. *In ICCV*, pages 13588–13597, 2021b.

Zhenbing Liu, Zeya Li, Ruili Wang, Ming Zong et Wanting Ji : Spatiotemporal saliency-based multi-stream networks with attention-aware lstm for action recognition. *Neural Computing and Applications*, 32(18):14593–14602, 2020.

Devashish Lohani, Carlos Crispim-Junior, Quentin Barthélemy, Sarah Bertrand, Lionel Robinault et Laure Tougne Rodet : Unsupervised and adaptive perimeter intrusion detector. *In 2022 IEEE International Conference on Image Processing (ICIP)*, pages 3621–3625. IEEE, 2022a.

Devashish Lohani, Carlos Crispim-Junior, Quentin Barthélemy, Sarah Bertrand, Lionel Robinault et Laure Tougne : Spatio-temporal convolutional autoencoders for perimeter intrusion detection. *In RRPR*, pages 47–65, 2021.

Devashish Lohani, Carlos Crispim-Junior, Quentin Barthélemy, Sarah Bertrand, Lionel Robinault et Laure Tougne Rodet : Perimeter intrusion detection by video surveillance: A survey. *Sensors*, 22(9), 2022b. ISSN 1424-8220. URL `https://www.mdpi.com/1424-8220/22/9/3601`.

C Lu, J Shi et J Jia : Abnormal event detection at 150 fps in Matlab. *In ICCV*, pages 2720–2727, 2013.

Yiwei Lu, Frank Yu, Mahesh Kumar Krishna Reddy et Yang Wang : Few-shot scene-adaptive anomaly detection. *In European Conference on Computer Vision*, pages 125–141. Springer, 2020.

Yue Lu, Congqi Cao, Yifan Zhang et Yanning Zhang : Learnable locality-sensitive hashing for video anomaly detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

Elena Luna, Juan Carlos San Miguel, Diego Ortego et José María Martínez : Abandoned object detection in video-surveillance: Survey and comparison. *Sensors*, 18 (12), 2018.

Dezhao Luo, Chang Liu, Yu Zhou, Dongbao Yang, Can Ma, Qixiang Ye et Weiping Wang : Video cloze procedure for self-supervised spatio-temporal learning. *In Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11701–11708, 2020.

W Luo, W Liu et S Gao : A revisit of sparse coding based anomaly detection in stacked RNN framework. *In ICCV*, pages 341–349, 2017a.

Weixin Luo, Wen Liu et Shenghua Gao : Remembering history with convolutional lstm for anomaly detection. *In 2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 439–444. IEEE, 2017b.

Weixin Luo, Wen Liu, Dongze Lian, Jinhui Tang, Lixin Duan, Xi Peng et Shenghua Gao : Video anomaly detection with sparse coding inspired deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(3):1070–1084, 2019.

H Lv, C Chen, Z Cui, C Xu, Y Li et J Yang : Learning normal dynamics in videos with meta prototype network. *In CVPR*, pages 15425–15434, 2021.

Lucia Maddalena et Alfredo Petrosino : A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Trans Image Process*, 17:1168–1177, 2008.

Mahshid Majd et Reza Safabakhsh : Correlational convolutional lstm for human action recognition. *Neurocomputing*, 396:224–229, 2020.

Snehashis Majhi, Srijan Das et François Brémond : Dam: Dissimilarity attention module for weakly-supervised video anomaly detection. *In 2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2021.

Jonathan Masci, Ueli Meier, Dan Cireşan et Jürgen Schmidhuber : Stacked convolutional auto-encoders for hierarchical feature extraction. *In International conference on artificial neural networks*, pages 52–59. Springer, 2011.

Dierck Matern, Alexandru Paul Condurache et Alfred Mertins : Automated intrusion detection for video surveillance using conditional random fields. *In MVA*, pages 298–301, 2013.

MathWorks : `https://fr.mathworks.com/discovery/lstm.html`, 2022.

Warren S McCulloch et Walter Pitts : A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Jefferson Ryan Medel et Andreas Savakis : Anomaly detection in video using predictive convolutional long short-term memory networks. *arXiv e-prints*, pages arXiv–1612, 2016.

Ramin Mehran, Alexis Oyama et Mubarak Shah : Abnormal crowd behavior detection using social force model. *In 2009 IEEE conference on computer vision and pattern recognition*, pages 935–942. IEEE, 2009.

Lili Meng, Bo Zhao, Bo Chang, Gao Huang, Wei Sun, Frederick Tung et Leonid Sigal : Interpretable spatio-temporal attention for video action recognition. *In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

V Menon et K Stephen : Re learning memory guided normality for anomaly detection. *In ML Reproducibility Challenge 2020*, 2021. URL `https://openreview.net/forum?id=vvLWTXkJ2Zv`.

Gaurav Mittal, Sushrutha Locharam, Sreela Sasi, Glenn R Shaffer et Ajith K Kumar : An efficient video enhancement method using LA*B* analysis. *In AVSS*, pages 66–66, 2006.

Bahram Mohammadi, Mahmood Fathy et Mohammad Sabokrou : Image/video deep anomaly detection: A survey. *arXiv preprint arXiv:2103.01739*, 2021.

R Nayak, U C Pati et S K Das : A comprehensive review on deep learning-based methods for video anomaly detection. *Image Vis Comput*, 106:104078, 2021.

Rashmiranjan Nayak, Mohini Mohan Behera, Umesh Chandra Pati et Santos Kumar Das : Video-based real-time intrusion detection system using deep-learning for smart city applications. *In ANTS*, pages 1–6, 2019.

Daniel Neimark, Omri Bar, Maya Zohar et Dotan Asselmann : Video transformer network. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3163–3172, 2021.

Trong-Nguyen Nguyen et Jean Meunier : Anomaly detection in video sequence with appearance-motion correspondence. *In Proceedings of the IEEE/CVF international conference on computer vision*, pages 1273–1283, 2019.

Michael Nielsen : http://neuralnetworksanddeeplearning.com, 2017. Accessed: 2022-11-20.

Hyeonwoo Noh, Seunghoon Hong et Bohyung Han : Learning deconvolution network for semantic segmentation. *In Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

Genevieve Orr, Nici Schraudolph et Fred Cummins : https://www.willamette.edu/~gorr/classes/cs449/momrate.html, 1999.

Yuqi Ouyang et Victor Sanchez : Video anomaly detection by estimating likelihood of representations. *In 2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8984–8991. IEEE, 2021.

C Park, M Cho, M Lee et S Lee : FastAno: Fast anomaly detection via spatio-temporal patch transformation. *In WACV*, pages 2249–2259, 2022.

H Park, J Noh et B Ham : Learning memory-guided normality for anomaly detection. *In CVPR*, pages 14372–14381, 2020.

Luis Patino, Jonathan Boyle, James Ferryman, Jonas Auer, Julian Pegoraro, Roman Pflugfelder, Mertcan Cokbas, Janusz Konrad, Prakash Ishwar, Giulia Slavic *et al.* : PETS2021: Through-foliage detection and tracking challenge and evaluation. *In AVSS*, pages 1–10, 2021.

Toby Perrett et Dima Damen : Ddlstm: dual-domain lstm for cross-dataset action recognition. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7852–7861, 2019.

Oluwatoyin P Popoola et Kejun Wang : Video-based abnormal human behavior recognition—a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):865–878, 2012.

Ning Qian : On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Zhaofan Qiu, Ting Yao et Tao Mei : Learning spatio-temporal representation with pseudo-3d residual networks. *In proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

B Ramachandra et M Jones : Street scene: A new dataset and evaluation protocol for video anomaly detection. *In WACV*, pages 2569–2578, 2020.

Bharathkumar Ramachandra, Michael J Jones et Ranga Raju Vatsavai : A survey of single-scene video anomaly detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(05):2293–2312, 2022.

Yunbo Rao et Leiting Chen : A survey of video enhancement techniques. *J Inf Hiding Multimedia Signal Process*, 3:71–99, 2012.

Yunbo Rao, Zhongho Chen, Ming-Ting Sun, Yu-Feng Hsu et Zhengyou Zhang : An effective night video enhancement algorithm. *In VCIP*, pages 1–4, 2011.

Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo Regazzoni et Nicu Sebe : Abnormal event detection in videos using generative adversarial nets. *In ICIP*, pages 1577–1581, 2017.

Joseph Redmon et Ali Farhadi : YOLO9000: better, faster, stronger. *In CVPR*, pages 7263–7271, 2017.

Joseph Redmon et Ali Farhadi : Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun : Faster R-CNN: towards real-time object detection with region proposal networks. *In NIPS*, pages 91–99, 2015.

M Ribeiro, AE Lazzaretti et HS Lopes : A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognit Lett*, 105:13–22, 2018.

N-C Ristea, N Madan, R T Ionescu, K Nasrollahi, F S Khan, T B Moeslund et M Shah : Self-supervised predictive convolutional attentive block for anomaly detection. *In CVPR*, pages 13576–13586, 2022.

Lionel Robinault : *Détection et suivi d'objets mobiles par caméras fixes*. Habilitation à diriger des recherches, Université Lumière Lyon 2, juin 2021. URL `https://hal.univ-lyon2.fr/tel-03274532`.

Olaf Ronneberger, Philipp Fischer et Thomas Brox : U-net: Convolutional networks for biomedical image segmentation. *In International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Frank Rosenblatt : *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller et Marius Kloft : Deep semi-supervised anomaly detection. *In International Conference on Learning Representations*, 2019.

David E Rumelhart, Geoffrey E Hinton et Ronald J Williams : Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein *et al.* : Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy et Ehsan Adeli : Adversarially learned one-class classifier for novelty detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3379–3388, 2018.

Takaya Saito et Marc Rehmsmeier : The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 10 (3):e0118432, 2015.

Haşim Sak, Andrew Senior, Kanishka Rao et Françoise Beaufays : Fast and accurate recurrent neural network acoustic models for speech recognition. *arXiv preprint arXiv:1507.06947*, 2015.

V Saligrama, J Konrad et P-M Jodoin : Video anomaly identification. *IEEE Signal Process Mag*, 27:18–33, 2010.

Kamal Sehairi, Fatima Chouireb et Jean Meunier : Comparative study of motion detection methods for video surveillance systems. *J Electron Imaging*, 26:023025, 2017.

Ling Shao, Jungong Han, Pushmeet Kohli et Zhengyou Zhang : *Computer vision and machine learning with RGB-D sensors*, volume 20. Springer, 2014.

Shikhar Sharma, Ryan Kiros et Ruslan Salakhutdinov : Action recognition using visual attention. *In International Conference on Learning Representations: Workshop*, 2016.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong et Wang-chun Woo : Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

Yemin Shi, Yonghong Tian, Yaowei Wang, Wei Zeng et Tiejun Huang : Learning long-term dependencies for action recognition with a biologically-inspired deep network. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 716–725, 2017.

Karen Simonyan et Andrew Zisserman : Two-stream convolutional networks for action recognition in videos. *In Advances in neural information processing systems*, pages 568–576, 2014.

P Sinha et R Russell : A perceptually based comparison of image similarity metrics. *Perception*, 40:1269–1281, 2011.

A A Sodemann, M P Ross et B J Borghetti : A review of anomaly detection in automated surveillance. *IEEE Trans Syst Man Cybern C Appl Rev*, 42:1257–1272, 2012.

Marina Sokolova, Nathalie Japkowicz et Stan Szpakowicz : Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. *In AJCAI*, pages 1015–1021, 2006.

Nitish Srivastava, Elman Mansimov et Ruslan Salakhudinov : Unsupervised learning of video representations using lstms. *In International conference on machine learning*, pages 843–852. PMLR, 2015.

Chris Stauffer et W Eric L Grimson : Adaptive background mixture models for real-time tracking. *In CVPR*, volume 2, pages 246–252, 1999.

Christoph Stiller et Janusz Konrad : Estimating motion in image sequences. *IEEE Signal Process Mag*, 16(4):70–91, 1999.

Swathikiran Sudhakaran, Sergio Escalera et Oswald Lanz : Lsta: Long short-term attention for egocentric action recognition. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9954–9963, 2019.

Swathikiran Sudhakaran, Sergio Escalera et Oswald Lanz : Gate-shift networks for video action recognition. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1102–1111, 2020.

Waqas Sultani, Chen Chen et Mubarak Shah : Real-world anomaly detection in surveillance videos. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488, 2018.

Che Sun, Yunde Jia, Hao Song et Yuwei Wu : Adversarial 3d convolutional auto-encoder for abnormal event detection in videos. *IEEE Transactions on Multimedia*, 23:3292–3305, 2020.

Lin Sun, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E Shi et Silvio Savarese : Lattice long short-term memory for human action recognition. *In Proceedings of the IEEE international conference on computer vision*, pages 2147–2156, 2017.

S Szymanowicz, J Charles et R Cipolla : Discrete neural representations for explainable anomaly detection. *In WACV*, pages 148–156, 2022.

Y Tang, L Zhao, S Zhang, C Gong, G Li et J Yang : Integrating prediction and reconstruction for anomaly detection. *Pattern Recognit Lett*, 129:123–130, 2020.

David Thirde, Longzhen Li et F Ferryman : Overview of the PETS2006 challenge. *In PETS*, pages 47–50, 2006.

Yu Tian, Guansong Pang, Yuanhong Chen, Rajvinder Singh, Johan W Verjans et Gustavo Carneiro : Weakly-supervised video anomaly detection with robust temporal feature magnitude learning. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4975–4986, 2021.

Tijmen Tieleman et Geoffrey Hinton : Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *COURSERA Neural Networks Mach. Learn*, 2012.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani et Manohar Paluri : Learning spatiotemporal features with 3d convolutional networks. *In Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun et Manohar Paluri : A closer look at spatiotemporal convolutions for action recognition. *In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

Hanh TM Tran et David Hogg : Anomaly detection using a convolutional winner-take-all autoencoder. *In Proceedings of the British Machine Vision Conference 2017*. BMVC, 2017.

Vassilios Tsakanikas et Tasos Dagiuklas : Video surveillance systems-current status and future trends. *Computers & Electrical Engineering*, 70:736–753, 2018.

Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad et Sung Wook Baik : Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE access*, 6:1155–1166, 2017.

Maria Valera et Sergio A Velastin : Intelligent distributed surveillance systems: a review. *IEE Proc Vis Image Sign Process*, 152:192–204, 2005.

Leslie G Valiant : A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Koen Van De Sande, Theo Gevers et Cees Snoek : Evaluating color descriptors for object and scene recognition. *IEEE Trans Pattern Anal Mach Intell*, 32:1582–1596, 2009.

Gül Varol, Ivan Laptev et Cordelia Schmid : Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40 (6):1510–1517, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser et Illia Polosukhin : Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Julien A Vijverberg, Roel TM Janssen, Remco de Zwart et Peter HN de With : Perimeter-intrusion event classification for on-line detection using multiple instance learning solving temporal ambiguities. *In ICIP*, pages 2408–2412, 2014.

Julien A Vijverberg, Cornelis J Koeleman et Peter HN de With : Towards real-time and low-latency video object tracking by linking tracklets of incomplete detections. *In AVSS*, pages 300–305, 2013.

Julien A Vijverberg, Marijn JH Loomans, Cornelis J Koeleman *et al.* : Two novel motion-based algorithms for surveillance video analysis on embedded platforms. *In Proc. SPIE 7724, Real-Time Image and Video Processing*, volume 7724, page 77240I, 2010.

Michael Villamizar, Angel Martínez-González, Olivier Canévet et Jean-Marc Odobez : WatchNet: Efficient and depth-based network for people detection in video surveillance systems. *In AVSS*, pages 1–6, 2018.

Paul Viola et Michael Jones : Rapid object detection using a boosted cascade of simple features. *In CVPR*, volume 1, pages I–I, 2001.

Vinay Vishwakarma, Chittaranjan Mandal et Shamik Sural : Automatic detection of human fall in video. *In International conference on pattern recognition and machine intelligence*, pages 616–623. Springer, 2007.

J Wang, J Jiao et Y-H Liu : Self-supervised video representation learning by pace prediction. *In ECCV*, pages 504–521, 2020.

Limin Wang, Zhe Wang, Yuanjun Xiong et Yu Qiao : Cuhk&siat submission for thumos15 action recognition challenge. *THUMOS Action Recognition challenge*, pages 1–3, 2015a.

Limin Wang, Yuanjun Xiong, Zhe Wang et Yu Qiao : Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015b.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang et Luc Van Gool : Temporal segment networks: Towards good practices for deep action recognition. *In European conference on computer vision*, pages 20–36. Springer, 2016a.

Wenguan WANG, Hongmei SONG, Shuyang ZHAO, Jianbing SHEN, Sanyuan ZHAO, Steven CH HOI et Haibin LING : Learning unsupervised video object segmentation through visual attention. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3064–3074, 2019.

Xuanhan WANG, Lianli GAO, Peng WANG, Xiaoshuai SUN et Xianglong LIU : Two-stream 3-d convnet fusion for action recognition in videos with arbitrary size and length. *IEEE Transactions on Multimedia*, 20(3):634–644, 2017.

Xue WANG, Sheng WANG et Daowei BI : Distributed visual-target-surveillance system in wireless sensor networks. *IEEE Trans Syst Man Cybern B Cybern*, 39:1134–1146, 2009.

Yasi WANG, Hongxun YAO et Sicheng ZHAO : Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016b.

WIKIPEDIA CONTRIBUTORS : Autoencoder — Wikipedia, the free encyclopedia, 2022. URL `https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=1120107812`. [Online; accessed 23-December-2022].

Christopher Richard WREN, Ali AZARBAYEJANI, Trevor DARRELL et Alex Paul PENTLAND : Pfinder: Real-time tracking of the human body. *IEEE Trans Pattern Anal Mach Intell*, 19:780–785, 1997.

Zuxuan WU, Caiming XIONG, Yu-Gang JIANG et Larry S DAVIS : Liteeval: A coarse-to-fine framework for resource efficient video recognition. *Advances in Neural Information Processing Systems*, 32, 2019a.

Zuxuan WU, Caiming XIONG, Chih-Yao MA, Richard SOCHER et Larry S DAVIS : Adaframe: Adaptive frame selection for fast video recognition. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1278–1287, 2019b.

Saining XIE, Chen SUN, Jonathan HUANG, Zhuowen TU et Kevin MURPHY : Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 1(2):5, 2017a.

Saining XIE, Chen SUN, Jonathan HUANG, Zhuowen TU et Kevin MURPHY : Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. *In Proceedings of the European conference on computer vision (ECCV)*, pages 305–321, 2018.

Xuemei XIE, Chenye WANG, Shu CHEN, Guangming SHI et Zhifu ZHAO : Real-time illegal parking detection system based on deep learning. *In ICDLT*, pages 23–27, 2017b.

Dan XU, Elisa RICCI, Yan YAN, Jingkuan SONG et Nicu SEBE : Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015a.

Kelvin XU, Jimmy BA, Ryan KIROS, Kyunghyun CHO, Aaron COURVILLE, Ruslan SALAKHUDINOV, Rich ZEMEL et Yoshua BENGIO : Show, attend and tell: Neural image caption generation with visual attention. *In International conference on machine learning*, pages 2048–2057. PMLR, 2015b.

Yong Xu, Jixiang Dong, Bob Zhang et Daoyun Xu : Background modeling methods in video analysis: A review and comparative evaluation. *CAAI Trans Intell Technol*, 1:43–60, 2016.

Y Yao, C Liu, D Luo, Y Zhou et Q Ye : Video playback rate perception for self-supervised spatio-temporal representation learning. *In CVPR*, pages 6548–6557, 2020.

Mu Ye, X Peng, W Gan, W Wu et Y Qiao : AnoPCN: Video anomaly detection via deep predictive coding network. *In ACM Int Conf Multimed*, pages 1805–1813, 2019.

W Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey *et al.* : Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Guang Yu, Siqi Wang, Zhiping Cai, En Zhu, Chuanfu Xu, Jianping Yin et Marius Kloft : Cloze test helps: Effective video anomaly detection via learning to complete video events. *In Proceedings of the 28th ACM International Conference on Multimedia*, pages 583–591, 2020.

Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga et George Toderici : Beyond short snippets: Deep networks for video classification. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

Sergey Zagoruyko et Nikos Komodakis : Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

Matthew D Zeiler : Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor et Rob Fergus : Deconvolutional networks. *In CVPR*, pages 2528–2535, 2010.

Gloria Zen et Elisa Ricci : Earth mover's prototypes: A convex learning approach for discovering activity patterns in dynamic scenes. *In CVPR 2011*, pages 3225–3232. IEEE, 2011.

Yanhong Zeng, Jianlong Fu et Hongyang Chao : Learning joint spatial-temporal transformations for video inpainting. *In European Conference on Computer Vision*, pages 528–543. Springer, 2020.

R Zhang : Making convolutional networks shift-invariant again. *In ICML*, pages 7324–7334, 2019.

Han Zhao et Xinyu Jin : Human action recognition based on improved fusion attention cnn and rnn. *In 2020 5th International Conference on Computational Intelligence and Applications (ICCIA)*, pages 108–112. IEEE, 2020.

Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu et Xian-Sheng Hua : Spatio-temporal autoencoder for video anomaly detection. *In ACM Int Conf Multimed*, pages 1933–1941, 2017.

Jia-Xing Zhong, Nannan Li, Weijie Kong, Shan Liu, Thomas H Li et Ge Li : Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1237–1246, 2019.

Bolei Zhou, Alex Andonian, Aude Oliva et Antonio Torralba : Temporal relational reasoning in videos. *In Proceedings of the European conference on computer vision (ECCV)*, pages 803–818, 2018.

Joey Tianyi Zhou, Le Zhang, Zhiwen Fang, Jiawei Du, Xi Peng et Yang Xiao : Attention-driven loss for anomaly detection in video surveillance. *IEEE transactions on circuits and systems for video technology*, 30(12):4639–4647, 2019.

Linchao Zhu, Du Tran, Laura Sevilla-Lara, Yi Yang, Matt Feiszli et Heng Wang : Faster recurrent networks for efficient video classification. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13098–13105, 2020.

Zoran Zivkovic : Improved adaptive gaussian mixture model for background subtraction. *In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31. IEEE, 2004.

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho et Haifeng Chen : Deep autoencoding gaussian mixture model for unsupervised anomaly detection. *In International conference on learning representations*, 2018.

Zhengxia Zou, Zhenwei Shi, Yuhong Guo et Jieping Ye : Object detection in 20 years: A survey. *arXiv:1905.05055*, 2019.

# List of Figures

# List of Tables